THE LOGIC OF CATEGORIAL GRAMMARS

Last modified: January 20, 2004

Christian Retoré
Université Bordeaux 1

Projet SIGNES
INRIA-Futurs & LaBRI-C.N.R.S. & Université Bordeaux 3
Université Bordeaux 1
351, cours de la Libération
33405 Talence Cedex
FRANCE

http://www.labri.fr/Recherche/LLA/signes

LECTURE NOTES

[0]A first version was written for a lecture on *The logic of categorial grammars* at ESSLLI 2000. It has then been improved for an ACL 2001 tutorial, a DEA lecture in Bordeaux, a lecture at ESSLLI 2003, another DEA lecture in Bordeaux.

I would like to thank the participants or readers and in particular Roberto Bonato, Pierre Castéran, Dick Crouch, Annie Foret, Nissim Francez, Paul Gochet, Gérard Huet, Dan Klein, Yannick Le Nir, Ralf Matthes, Laurent Miclet, Sylvain Pogodalla, Géraud Sénizergues, Marc Tommasi, who provided comments on the lecture or on the notes.

Together with some other lecture notes by Nissim Francez, these notes will possibly lead to a book.

General References

Lambek's syntactic calculus is the center of this lecture and we strongly recommend Lambek's original article [8] the elegance of which is hard to meet.

For further general references on this logical view of categorial grammars the reader is referred to two chapters of the Handbook *Logic and Language* [1] namely [9,10]; the short survey [11] provides a non technical and up to date state of the art.

Regarding learning algorithms for categorial grammars we recommand [12] or the study [13] which includes a good survey.

If some background is needed we recommend [14] for proof theory and typed λ-calculus and [15] for formal language theory, while [16] provides a general introduction to linear logic .

Finally, let us say that our logical view of categorial grammar is not the only one, and regarding the combinatorial view of categorial grammars we recommend [17].

---

[8] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.

[1] Johan van Benthem and Alice ter Meulen, editors. *Handbook of Logic and Language*. North-Holland Elsevier, Amsterdam, 1997.

[9] Wojciech Buszkowski. Mathematical linguistics and proof theory. In van Benthem and ter Meulen [1], chapter 12, pages 683–736.

[10] Michael Moortgat. Categorial type logic. In van Benthem and ter Meulen [1], chapter 2, pages 93–177.

[11] Christian Retoré. Systèmes déductifs et traitement des langues: un panorama des grammaires catégorielles. *Technique et Science Informatiques*, 20(3):301–336, 2000. Numéro spécial *Traitement Automatique du Langage Naturel* sous la direction de D. Kayser et B. Levrat. Version préliminaire RR-3917 http://www.inria.fr/.

[12] Makoto Kanazawa. *Learnable classes of categorial grammars*. Studies in Logic, Language and Information. FoLLI & CSLI, 1998. distributed by Cambridge University Press.

[13] Roberto Bonato. Uno studio sull'apprendibilità delle grammatiche di Lambek rigide — a study on learnability for rigid Lambek grammars. Tesi di Laurea & Mémoire de D.E.A, Università di Verona & Université Rennes 1, 2000.

[14] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

[15] John E. Hopcroft and Jeffrey Ullman. *Introduction to automata theory, languages and computation*. Addison Wesley, 1979.

[16] Jean-Yves Girard. Linear logic: its syntax and semantics. In Girard et al. [2], pages 1–42.

[2] Jean-Yves Girard, Yves Lafont, and Laurent Regnier, editors. *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Notes*. Cambridge University Press, 1995.

[17] Mark Steedman. *Surface structure and interpretation*. Number 30 in Linguistic Inquiry Monographs. M.I.T. Press, Cambridge, Massachusetts, 1997.

Contents

Chapter 1

Classical categorial grammars: AB grammars

This first chapter deals with material from the late fifties and early sixties, but which nevertheless introduce the design of categorial grammars, which are lexcalized grammars, as opposed to the phrase structure grammars like context-free grammars that were introduced later on.

Although the success of phrase structure grammars went far beyond the one of categorial grammars, their lexicalization was in fact a modern feature, the other one being their connection to logical semantics.

We end with recent results: a learning algorithm for categorial grammars from the nineties, which prove to converge only a few years ago. The possibility to have a learning algorithm for a class of grammar which can describe (small parts of) natural language is, we think, quite an important feature of categorial grammars. It comes from their lexicalisation and logical formulation, which will be further studied in the next chapter.

1.1 Semantic categories and Ajdukiewicz fractions

For the history of categorial grammars, we refer the reader to [18].

In 1935 Ajdukiewicz defined a calculus of fractions to test the correction of logical statements [19]:

> *The discovery of antinomies, and the method of their resolution have made problems of linguistic syntax the most important problems of logic (provided this word is understood in a sense that also includes meta-theoretical considerations). Among these problems that of syntactic connection is of the greatest importance for logic. It is concerned with the specification of the conditions under which a word pattern constituted of meaningful words, forms an expression which itself has a unified meaning (constituted, to be sure, by the meaning of the single words belonging to it). A word pattern of this kind is call syntactically connected.*

His paper deal with both the formal language of logic and natural language, but is actually more concerned with the language of propositional and predicate logic.

> *If one applies this index symbolism to ordinary language, the semantic categories which we have assumed (in accordance with Lesniewski) will not always suffice, since ordinary language are richer in semantic categories.*

Each word (or lexical entry) is provided with an index which is a fraction. Fractions are defined out of two primitive types n (for entities or individuals or first order terms) and s (for propositions or truth values) and whenever N is a fraction and $D_1, \ldots, D_p$ is a sequence or multiset of fractions, $\frac{N}{D_1 \cdots D_p}$ is itself a fraction.

To formalize the definition in his article, syntactically connected expressions and their exponents are recursively defined as follows:

- a word or lexical entry is syntactically connected, and its exponent is its index.
- given
  - n syntactically connected expressions $d_1, \ldots, d_n$ of respective exponents $D_1, \ldots, D_n$
  - an expression f of exponent $\frac{N}{D_1 \cdots D_p}$

[18] Claudia Casadio. Semantic categories and the development of categorial grammars. In R. Oehrle, E. Bach, and D. Wheeler, editors, *Categorial Grammars and Natural Language Structures*, pages 95–124. Reidel, Dordrecht, 1988.

[19] Kazimierz Ajdukiewicz. Die syntaktische konnexität. *Studia Philosophica*, 1:1–27, 1935. (English translation in [3], pages 207–231).

[3] Storrs McCall, editor. *Polish Logic, 1920-1939*. Oxford University Press, 1967.

the expression $f d_1 \cdots d_n$ (or any permutation of it) is syntactically connected and has exponent N.

This in particular entails that the sequences of fraction reduces to a single index using the usual simplifications for fractions. It should be observed that in this "commutative setting" the simplification procedure of the fractions is not that simple : if the bracketing corresponding to subexpressions is not given. As Ajdukiewicz is mainly concerned with the language of logic where one can use the Polish notation, word order is not really a problem for him.

1.2 Classical categorial grammars or AB grammars

In 1953,that is a bit before Chomsky introduced his hierarchy of Phrase Structure Grammars [20], Bar-Hillel defines bidirectional categorial grammars [21], taking into account constituent order in Ajdukiewicz types. Therefore his grammars are more concerned with natural language where word order is crucial.

In the literature, these grammars are either called *AB grammars*, *classical categorial grammars*, or *basic categorial grammars*, *BCG*.

Types or fractions are defined as follows:

$$\mathsf{L} ::= \mathsf{P} \quad | \quad \mathsf{L} \setminus \mathsf{L} \quad | \quad \mathsf{L} / \mathsf{L}$$

where P is the set of primitive types or basic categories which usually contains S (for sentences) np (for noun phrases) and n (for nouns), and may include pp (for prepositional phrase) vp (for verb phrase) etc.

The grammar is defined by a lexicon, that is a function Lex which maps words or terminals to finite sets of types (a set of types is needed, since in natural language a single word may admit various constructions: *eat* may ask for an object or not, for instance).

An expression, that is a sequences of words or terminals $w_1 \cdots w_n$, is of type u whenever there exists for each w_i a type t_i in $\mathrm{Lex}(w_i)$ such that $t_1 \cdots t_n \longrightarrow u$ with the following reduction patterns:

$$\forall u, v \in \mathsf{L} \qquad \begin{array}{ll} u(u \setminus v) \longrightarrow v & (\setminus_e) \\ (v / u)u \longrightarrow v & (/_e) \end{array}$$

These rules are called residuation laws, or simplifications, or modus ponens.

These rules provides the symbols $\setminus$ and $/$ with an intuitive meaning: an expression y is of type $A \setminus B$ whenever it needs an expression a of type A on its left to obtain an

[20] Noam Chomsky. The logical structure of linguistic theory. Revised 1956 version published in part by Plenum Press, 1975; University of Chicago Press, 1985, 1955.

[21] Yehoshua Bar-Hillel. A quasi arithmetical notation for syntactic description. *Language*, 29:47–58, 1953.

expression *ay* of type *B*; symmetrically, an expression *z* is of type B/A whenever it needs an expression *a* of type *A* on its right to obtain an expression *za* of type *B*;

The set of sentences or the language generated by the grammar is the set of word sequences of type *S*.

The derivation tree is simply a binary tree whose leaves are the t_i and whose nodes are labeled by rules $/_e$ and $\backslash_e$.

It should be observed that such a grammar is lexicalized: to generate different languages the rules do not change, but only the lexicon, and this is coherent with modern linguistic theories, like the minimalist program of Chomsky [22] (language variation is only lexical), and with some formalisms for computational linguistics, like (Lexicalized) Tree Adjoining Grammars [23,24] or Head-Driven Phrase Structure Grammars [25,26].

Another observation is that the rules are like modus ponens, but in a logic where contraction and weakening is not allowed, and where the order of the hypothesis is taken into account. We shall come back on these matters.

Let us state one of the first result on categorial grammars known as the Gaifman theorem of [27] which is more or less equivalent to the existence of a Greibach normal form for context-free grammars:

Proposition 1 *Every AB grammar is equivalent to an AB grammar containing only types of the form*

$$p \qquad (p/q) \qquad ((p/q)/r)$$

where p, qr *stand for primitive types.*

PROOF : This theorem is an immediate consequence of propositions 4 and 3 to be proved below using Greibach normal form theorem that is now famous. This enables a simpler proof. ◇

[22] Noam Chomsky. *The minimalist program*. MIT Press, Cambridge, MA, 1995.

[23] Aravind Joshi, Leon Levy, and Masako Takahashi. Tree adjunct grammar. *Journal of Computer and System Sciences*, 10:136–163, 1975.

[24] Aravind Joshi and Yves Schabes. Tree adjoining grammars. In Rozenberg and Salomaa [4], chapter 2.

[4] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*. Springer Verlag, Berlin, 1997.

[25] Fernando C. N. Pereira and Stuart M. Shieber. *Prolog and Natural-Language Analysis*. Number 10 in CSLI Lecture Notes. University of Chicago Press, Chicago, IL, 1987.

[26] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. Center for the Study of Language and Information, Stanford, CA, USA, 1994. (distributed by Cambridge University Press).

[27] Yehoshua Bar-Hillel, Chaim Gaifman, and Eli Shamir. On categorial and phrase-structure grammars. *Bulletin of the research council of Israel*, F(9):1–16, 1963.

1.3 Example: a tiny AB grammar

Consider the following lexicon:

Word	Type(s)
cosa	$(S/(S/np))$
guarda	(S/vp)
passare	(vp/np)
il	(np/n)
treno	n

The sentence *guarda passare il treno* (he/she looks the train passing by) belongs to the generated language:

$$\begin{array}{l}(S/vp) \quad (vp/np) \quad (np/n) \quad n\\ \longrightarrow (S/vp) \quad (vp/np) \quad np\\ \longrightarrow (S/vp) \quad vp\\ \longrightarrow S\end{array}$$

The derivation tree for this analysis can be written as:

$$[_{/_e}(S/vp) \quad [_{/_e}(vp/np) \quad [_{/_e}(np/n) \quad n]]]$$

The sentence *cosa guarda passare* (what is he/she looking passing by?) does not belong to the generated language : indeed the sequence

$$(S/(S/np))(S/vp)(vp/np)$$

does not contain anything that could be reduced.

Exercises: define AB grammars for

1. $a^n b^n$,
2. brackets $(((\,)\,)(\,))$
3. for a small fragment of English with np being proper names or determinants applied to nouns, and (try to) extend it to include relative pronouns.

1.4 AB-grammars and context free grammars

1.4.1 Context-free grammars

Context-Free Grammars (CFGs) were introduced in [20] and a good introduction is provided in [15]; we use the following standard notation:

[20] Noam Chomsky. The logical structure of linguistic theory. Revised 1956 version published in part by Plenum Press, 1975; University of Chicago Press, 1985, 1955.

[15] John E. Hopcroft and Jeffrey Ullman. *Introduction to automata theory, languages and computation*. Addison Wesley, 1979.

- M^* stands for the set of finite sequences over the set M.
- M^+ stands for the set of finite non empty sequences over the set M.
- ε stands for the empty sequence of M^*.

A context free grammar is defined by:

Non Terminals a set NT of symbols called non terminals, one of them, S being the start symbol.

Terminals a disjoint set T of symbols called terminals (or words according to the linguistic viewpoint)

production rules a finite set of production rules of the form $X \longrightarrow W$ with $X \in NT$ and $W \in (T \cup NT)^*$

A sequence $V \in (T \cup NT)^*$ is said to rewrite immediately into a sequence $W \in (T \cup NT)^*$ whenever there exists $W', W'', W''' \in (T \cup NT)^*$ and a non terminal X such that

- $V = W'XW''$
- $X \longrightarrow W''$ is a production rule.
- $W = W'W''W'''$

The relation $\longrightarrow$ is defined over sequences in $(T \cup NT)^*$ as the transitive closure of "rewrites immediately into". The language generated by a CFG is the subset of T^* containing the sequences into which S rewrites.

Two grammars which generates the same languages are said to be weakly equivalent.

Whenever a non-terminal N rewrites into a sequence of terminals and non terminals $X_1, \ldots, X_n$ it is possible (as linguist often do) to denote the derivation tree by an expression in DT :

- a non terminal or a terminal is a DT and its yield is itself.
- if $T_1, \ldots, T_n$ are DT produced by non terminals $X_1, \ldots, X_n$ and if $X \longrightarrow X_1 \cdots X_n$ is a rule of the grammar then $[_X T_1, \ldots, T_n]$ is a DT labeled X and its yield is the concatenation of the yields of $T_1, \ldots, T_n$.

Obviously a sequence of terminals $a_1 \cdots a_n$ is in the language if and only if there exists a derivation tree labeled S the yield of which is $a_1 \cdots a_n$. We denote by ε the empty sequence.

Two grammars which generate the same derivation *trees* are said to be strongly equivalent.

A CFG is said to be in Chomsky normal from whenever its production rules are of the form $X \longrightarrow Y\,Z$ and $X \longrightarrow a$ with $X, Y, Z \in NT$ and $a \in T$. Any CFG can be turned into a weakly equivalent CFG in Chomsky normal form and this transformation can be performed in polynomial time. [28,15]

A CFG is said to be ε free whenever ε does not belong to the generated language. It is easy to decide whether a CFG is ε free or not, and if it is not ε-free, the grammar can be written with production rules of an ε-free CFG, together with the rule: $S \longrightarrow \varepsilon$.

A CFG is said to be in Greibach normal form whenever its production rules are of the form: $X \longrightarrow a\,X_1 \cdots X_n$ with $a \in NT$, $X, X_1, \ldots, X_n \in NT$. It is said to be in strong Greibach normal form whenever $n \leq 2$. Any ε-free CFG can be turned into a CFG in (strong) Greibach normal form, and these transformations can be performed in polynomial time. [29,30] While the derivation trees of a CFG and the ones of its Chomsky normal form are closely related, the derivation trees of the Greibach normal from of a CFG are in general very different from the derivation trees of the original CFG: to lexicalize a CFG while preserving the analyses, one has to move to TAGs [24].

1.4.2 From context-free grammars to AB-grammars

The study of the relationship between CFG and AB-grammars was studied and "finished" long ago, that is in the early sixties in particular in [27].

Proposition 2 *Every ε-free Context-Free Grammar in Greibach normal form is strongly equivalent to an AB categorial grammar.*

PROOF : Let us consider the following AB grammar:

- Its words are the terminals of the CFG.
- Its primitive types are the non terminals of the CFG.

[28] Noam Chomsky. Formal properties of grammars. In *Handbook of Mathematical Psychology*, volume 2, pages 323 – 418. Wiley, New-York, 1963.

[15] John E. Hopcroft and Jeffrey Ullman. *Introduction to automata theory, languages and computation*. Addison Wesley, 1979.

[29] Sheila A. Greibach. A new normal-form theorem for context-free phrase structure grammars. *Journal of the ACM*, 12(1):42–52, 1965.

[30] M. A. Harrison. *Introduction to Formal Language Theory*. Addison Wesley, 1978.

[24] Aravind Joshi and Yves Schabes. Tree adjoining grammars. In Rozenberg and Salomaa [4], chapter 2.

[4] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*. Springer Verlag, Berlin, 1997.

[27] Yehoshua Bar-Hillel, Chaim Gaifman, and Eli Shamir. On categorial and phrase-structure grammars. *Bulletin of the research council of Israel*, F(9):1–16, 1963.

- Lex(a), the finite set of types associated with a terminal a contains the formulae $((\cdots((X \,/\, X_n) \,/\, X_{n-1}) \,/\, \cdots) \,/\, X_2) \,/\, X_1$ such that there are non terminals $X, X_1, \ldots, X_n$ such that $X \longrightarrow aX_1 \cdots X_n$ is a production rule.

It is then easily observed that the derivation trees of both grammars are isomorphic. ◇

Proposition 3 *Each ε-free Context Free Grammar is weakly equivalent to an AB-grammar containing only types of the form X or $X \,/\, Y$ or $(X \,/\, Y) \,/\, Z$.*

PROOF : Here we provide the reader with a simple "modern proof" using the existence of a Greibach normal from: indeed the Gaifman theorem first published in [27] was proved before the existence of Greibach normal form for CFGs [29], and these two theorems are actually more or less equivalent.

As we just said, any CFG can be turned into a weakly equivalent CFG in strong Greibach normal form. As can be observed from the construction of an equivalent AB grammar in the previous proof, if the CFG is in strong Greibach normal form that is if rules are of the form: $X \longrightarrow aX_1 \cdots X_n$ with $0 \leq n \leq 2$, then the corresponding AB grammar only uses types of the form X, $X \,/\, X_1$, $(X \,/\, X_2) \,/\, X_1$. ◇

1.4.3 From AB grammars to context-free grammars

Proposition 4 *Every AB grammar is strongly equivalent to a CFG in Chomsky normal form.*

PROOF : Let G be the CFG defined by:

- Terminals T are the words of the AB grammar.
- Non Terminals NT are all the subtypes of the types appearing in the lexicon of the AB grammar — a type is considered to be a subtype of itself.
- The production rules are of two kinds:
  - $X \longrightarrow a$ whenever $X \in \text{Lex}(a)$
  - $X \longrightarrow (X \,/\, Z)Z$ and $X \longrightarrow Z(Z \setminus X)$ for all $X, Z \in NT$ — beware that from the CFG viewpoint $(Z \setminus X)$ or $(X \,/\, Z)$ is a single non terminal.

This defines a CFG because the lexicon is finite, so there are only finitely many subtypes of types in the lexicon, hence finitely many production rules. The derivation trees in both formalisms are isomorphic. ◇

---

[27] Yehoshua Bar-Hillel, Chaim Gaifman, and Eli Shamir. On categorial and phrase-structure grammars. *Bulletin of the research council of Israel*, F(9):1–16, 1963.

[29] Sheila A. Greibach. A new normal-form theorem for context-free phrase structure grammars. *Journal of the ACM*, 12(1):42–52, 1965.

1.5 Parsing AB grammars

Theorem 5 *A sentence of n word can be analyzed according to an AB-grammar in $O(n^3)$ times using $O(n^2)$ space.*

PROOF : (easy exercice) Following the relation between AB-grammars and CFG in Chomsky normal form, it is not difficult to adpat the Cocke Kasami Younger algorithm (see e.g. [31]) to AB grammars. ◇

1.6 Limitations of AB-grammars

In an AB grammar one is not able to derive $(t \,/\, v)$ from $(t \,/\, u)$ and $(u \,/\, v)$. Consider for instance the Italian sentence *Cosa guarda passare?*. One is not able to derive it with the simple type assignment given above. We would need transitivity of $/$ to obtain it:

$$(S \,/\, (S \,/\, np)) \quad (S \,/\, vp) \quad (vp \,/\, np) \quad \overset{(trans.)}{\longrightarrow} \quad (S \,/\, (S \,/\, np)) \quad (S \,/\, np) \quad \longrightarrow \quad S$$

We would also like to model the behavior of an object relative-pronoun like *that/whom*, by providing it with the type $(n \setminus n) \,/\, (S \,/\, np)$ but unfortunately this too requires transitivity — unless a transitive verb also has the type $np \setminus (S \,/\, np)$ but it is quite unnatural that the verb first combine with its subject and thereafter with its object.

On the mathematical side, one would like to interpret categories by subsets of a free monoid, (the intended one being sequences of words), so that the subset of sequences of type S are precisely the correct sentences. This is indeed impossible. One may view the residuation rules as modus ponens, but then what is lacking are introduction rules to get the completeness of the calculus with respect to this natural monoidal interpretation. This is sorted out by the Lambek calculus that we are to study later on.

1.7 Learning AB grammars

Let us end up our study of AB grammars with an interesting property: they enjoy good learning algorithms from positive examples, at least when example are structured. This learning question is important for the following two reasons:

- It models, although very roughly, the process of language acquisition and more

[31] Klaas Sikkel and Anton Nijholt. Parsing of context-free languages. In Rozenberg and Salomaa [4], chapter 2.

[4] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*. Springer Verlag, Berlin, 1997.

precisely of syntax acquisition [32,33] extensively discussed in generative linguistics; indeed, it is the main justification for the existence of a *universal grammar* see e.g. [22].

  - The similarity with natural language acquisition by human beings, is that we only learn from positive examples, and that structure is needed for the learning device.
  - The main difference is that the sequence of languages which converges to the target language is increasing, while in natural language acquisition the sequence of languages is decreasing.

- This learning algorithm provides a method for the automated construction of a grammar (that is a lexicon) from a corpus, which also can be viewed as an automated method for completing an existing grammar/lexicon.

1.7.1 Grammatical inference for categorial grammars

Learning (that in this case is also called grammatical inference) from positive example is the following problem: define a function Learn from finite sets of positive examples to grammars of a given class $\mathscr{G}$, such that:

- Given a grammar G of the class $\mathscr{G}$ and an enumeration $s_1, s_2, \ldots$ of the sentences G generates, letting $\mathrm{Ex}_i = \{s_1, \ldots, s_i\}$, there exists an N such that for all $n \geq N$ the grammar $\mathrm{Learn}(\mathrm{Ex}_n)$ is constant and exactly generates the sentences produced by G.

- The following is not mandatory, but one usually asks for this extra property: for every set of sentences Ex the grammar $\mathrm{Learn}(\mathrm{Ex})$ generates all the examples in Ex

This definition is the so-called identification in the limit introduced by Gold in 1967 [34]. The grammars we are to consider are of course AB grammars, but what will the positive examples be? In our definition the term ”sentence” is left vague. Actually we shall use this definition not with mere sequences of words, but we will rather consider the derivation trees produced by the grammar, and so our examples will be derivation trees in which the types of the words are absent: this is not absolutely

[32] L.R. Gleitman and E.L. Newport. The invention of language by children: Environmental and biological influences on the acquisition of language. In Gleitman and Liberman [5], chapter 1, pages 1–24.

[5] L.R. Gleitman and M. Liberman, editors. *An invitation to cognitive sciences, Vol. 1: Language*. MIT Press, 1995.

[33] Steven Pinker. Language acquisition. In Gleitman and Liberman [5], chapter 6, pages 135–182.

[22] Noam Chomsky. *The minimalist program*. MIT Press, Cambridge, MA, 1995.

[34] E. Mark Gold. Language identification in the limit. *Information and control*, 10:447–474, 1967.

unrealistic, because the learner of a language has access to some information related to the syntactic structure of the sentences like prosody or semantics; nevertheless it is unrealistic, because the complete syntactic structure is not fully known.

The lexicalization of categorial grammars is extremely helpful for this learning question: indeed we have no rules to learn, but only the types of the words to guess. Observe that it is needed to bound the number of types per word; otherwise each new occurrence of a word may lead to the introduction of a new type for this word, and this process cannot converge.

As this presentation is just meant to give an idea of learning algorithms, we only present here the simplest case of learning from structures: the algorithm RG of Buszkowski and Penn. The AB grammars considered are rigid, that is to say there is exactly one type per word.

1.7.2 Unification and AB grammars

The algorithm makes use of type-unification, and this kind of technique is quite common in grammatical inference see [35], so let us briefly define it and explain its relation to AB grammars. A substitution σ is a function from variables to types which is extended from types to types by

$$\sigma(S) = S \qquad \sigma(A \setminus B) = \sigma(A) \setminus \sigma(B) \qquad \sigma(B / A) = \sigma(B) / \sigma(A)$$

Given a substitution σ, one can apply it to the lexicon of an AB grammar. If a sentence is generated by an AB grammar defined by a lexicon Lex then it is also generated by the AB grammar defined by the lexicon $\sigma(\text{Lex})$.

A substitution is said to unify a set of types T if for all types A, B in T one has $\sigma(A) = \sigma(B)$. For such kinds of formulae, whenever a unifier exists, there exists a most general unifier (mgu) that is a unifier σ_u such for every unifier τ there exist a substitution σ_τ such that $\tau = \sigma_\tau \circ \sigma_u$.

The relation between two rigid AB grammars with respective lexicon Lex and Lex$'$ defined by *there exists a substitution* σ *such that* $\text{Lex}' = \sigma(\text{Lex})$ defines an order which is a complete lattice, and the supremum of a family correspond to the less general grammar generating all the trees of all the grammars in the family.

---

[35] Jacques Nicolas. Grammatical inference as unification. Rapport de Recherche RR-3632, INRIA, 1999. http://www.inria.fr/.

1.7.3 The RG algorithm

We present here the RG algorithm (learning Rigid Grammars) introduced by W. Buszkowski and G. Penn in [36,37] and which has been further studied by M. Kanazawa [12].

To illustrate this algorithm, let us take a small set of positive examples:

(1.1) $[_{\backslash_e}[_{/_e}$ a man] swims]

(1.2) $[_{\backslash_e}[_{/_e}$ a fish]$[_{\backslash_e}$ swims fast]]

Typing As the examples are assumed to be correct sentences we know the root should be labeled by the type S which is the only type fixed in advance, a constant.

Each time there is a $\backslash_e$ (resp. $/_e$) node labeled y, we know the argument node, the one on the left (resp. on the right) should be x while the function node the one on the right (resp. on the left) should be $x \backslash y$ (resp. y / x)

So by assigning a new variable to each argument node we have typed the whole tree, and so words have been provided with a type (involving the added variables and S).

We can do so on our examples; to denote the resulting type, we add it on top of the opening bracket.

(1.3) $[^{S}_{/_e}[^{x_2}_{\backslash_e}$ a:(x_2 / x_1) man:$x_1]$ swims:$(x_2 \backslash S)]$

(1.4) $[^{S}_{\backslash_e}[^{y_2}_{/_e}$ a:(y_2 / y_3) fish:$y_3][^{(y_2 \backslash S)}_{\backslash_e}$ swims:y_1 fast:$(y_1 \backslash (y_2 \backslash S))]]$

Unification We end up from the previous steps with several types per word. For instance the examples above yields:

(1.5)

word	*type*1	*type*2
a:	x_2 / x_1	y_2 / y_3
fast:		$y_1 \backslash (y_2 \backslash S)$
man:	x_1	
fish:		y_3
swims:	$x_2 \backslash S$	y_1

---

[36] Wojciech Buszkowski. Discovery procedures for categorial grammars. In J. van Benthem and E. Klein, editors, *Categories, Polymorphism and Unification*. Universiteit van Amsterdam, 1987.

[37] Wojciech Buszkowski and Gerald Penn. Categorial grammars determined from linguistic data by unification. *Studia Logica*, 49:431–454, 1990.

[12] Makoto Kanazawa. *Learnable classes of categorial grammars*. Studies in Logic, Language and Information. FoLLI & CSLI, 1998. distributed by Cambridge University Press.

One has then to unify the set of the types associated with a single word, and the output of the algorithm is the grammar/lexicon in which every words gets the single type which unifies the original types, collected from each occurrence of a word in each example. If these sets of types can be unified, then the result of this substitution is a rigid grammar which generates all the examples, and can be shown to be the less general grammar to generate these examples.

In our example, unification succeeds and leads the most general unifier σ_u defined as follows:

(1.6)

$$\begin{array}{rcl} \sigma_u(x_1) & = & z_1 \\ \sigma_u(x_2) & = & z_2 \\ \sigma_u(y_1) & = & z_2 \setminus S \\ \sigma_u(y_2) & = & z_2 \\ \sigma_u(y_3) & = & z_1 \end{array}$$

which yields the rigid grammar/lexicon:

(1.7)

$$\begin{array}{rl} \text{a:} & z_2 \,/\, z_1 \\ \text{fast:} & (z_2 \setminus S) \setminus (z_2 \setminus S) \\ \text{man:} & z_1 \\ \text{fish:} & z_1 \\ \text{swims:} & z_2 \setminus S \end{array}$$

1.7.3.1 Convergence of the RG algorithm

This algorithm converges in the sense we defined above, as shown by [12]. The technique also applies for learning rigid Lambek grammars from natural deduction trees [13] and we follow his presentation.

The proof of convergence makes use of the following notions and notations:

$G \subset G'$ This reflexive relation bewteen G and G' holds whenever every assignment $a : T$ in G is in G' as well — in particular when G' is rigid, so is G, and they are equal.

size of a grammar The size of a grammar is simply the sum of types in the lexicon, where the size of a type is its number of occurrences of base categories (variables or S).

---

[13] Roberto Bonato. Uno studio sull'apprendibilità delle grammatiche di Lambek rigide — a study on learnability for rigid Lambek grammars. Tesi di Laurea & Mémoire de D.E.A, Università di Verona & Université Rennes 1, 2000.

$G \sqsubset G'$ This reflexive relation bewteen G and G' holds when there exists a substitution σ such that $\sigma(G) \subset G'$ which dos not identify different types of a given word, but this is always the case when the grammar is rigid.

FA-structure An FA-structure is a binary tree whose leaves are labeled with words (terminals) and internal nodes with names of the rules, namely $/_e$ and $\backslash_e$. An analysis in an AB grammar, once the types are erased, is an FA structure, and, conversely, for every type T, every FA structure can be labeled with types in order to obtain an analysis of the sequence of words as having category T — that's what the typing algorithm does, with $T = S$. The positive examples we are using for the RG learning algorithm, see examples 1.1 and 1.2 are FA-structures.

$FL(G)$ Given a grammar G, $FL(G)$ is the tree language consisting in all the FA-structures with root S derived from G.

$GF(D)$ Given a set of FA-structures D, $GF(D)$ is the lexicon obtained by collecting the types of each word in the various examples of D — as in example 1.5 above.

$RG(D)$ Given a set of examples D, $RG(D)$ is, whenever it exists, the rigid grammar/lexicon obtained by applying the most general unifier to $GF(D)$ — as in example 1.7 above.

Proposition 6 *Given a grammar G there are finitely many grammars H such that $H \sqsubset G$.*

PROOF : There are finitely many grammars which are included in G, since G is a finite set of assignments. Whenever $\sigma(H) = K$ for some substitution σ the size of H is smaller or equal to the size of K, and, up to renaming, there are finitely many grammars smaller than a given grammar.

By definition, if $H \sqsubset G$ then there exist $K \subset G$ and a substitution σ such that $\sigma(H) = K$. Because there are finitely many K such that $K \subset G$, and for every K there are finitely many H for which there could exists σ with $\sigma(H) = K$ we conclude that there are finitely many H such that $H \sqsubset G$. ◇

Proposition 7 *If $G \sqsubset G'$ then $FL(G) \subset FL(G')$.*

PROOF : $G \sqsubset G'$ means that there exists σ such that $\sigma(G) \subset G'$. Let T be an FA-structure in $FL(G)$, hence T comes from an analysis A of a sequence of words $m_1 \cdots m_n$. If we apply σ to A we obtain an analysis of the same sequence of words in G'. Indeed for a word whose assignment is T in G we have the assignment $\sigma(T)$ which is its assignment in G', and the types obtained inside the tree match the sules since $\sigma(A \setminus B) = \sigma(A) \setminus \sigma(B)$ and $\sigma(A / B) = \sigma(A) / \sigma(B)$ So $\sigma(A)$ is an analysis in G' of $m_1 \cdots m_n$. hence, by definition, the FA-structure underlying $\sigma(A)$ is in $FL(G')$, and this underlying FA-structure is F.T ◇

Proposition 8 *If $GF(D) \sqsubset G$ then $D \subset FL(G)$.*

PROOF : By construction of $GF(D)$ we have $D \subset FL(GF(D))$ and we also have $FL(GF(D)) \subset FL(G)$ because of proposition 7. ◇

Proposition 9 *If $RG(D)$ exists then $D \subset FL(RG(D))$.*

PROOF : By definition $RG(D) = \sigma_u(GF(D))$ where σ_u is the most general unifier of all the types of each word. So we have $GF(D) \sqsubset RG(D)$, and applying previous proposition 8 with $G = RG(D)$ we obtain $D \subset FL(RG(D))$. ◇

Proposition 10 *If $D \subset FL(G)$ then $GF(D) \sqsubset G$.*

PROOF : By construction of $GF(D)$, there is exactly one occurrence of a given type variable x in a tree of D typed as we did in the example. Now, viewing the same tree as a tree of $FL(G)$ at the place as x there is a type label, say T. Doing so for every type variable, we can define a substitution by $\sigma(x) = T$ for all type variables x: indeed because x occurs once, such a substitution is well defined. When this substitution is applied to $GF(D)$ it yields a grammar which only contains assignments from G — by applying the substitution on the whole tree, still it is a well typed tree, and in particular the types on the leaves must coincide. ◇

Proposition 11 *When $D \subset FL(G)$ with G a rigid grammar, the grammar $RG(D)$ exists and $RG(D) \sqsubset G$.*

PROOF : By proposition 10 we have $GF(D) \sqsubset G$, so there exists a substitution σ such that $\sigma(GF(D)) \subset G$.

As G is rigid, σ unifies all the types of each word. Hence there exists a unifier of all the types of each word, and $RG(D)$ exists.

$RG(D)$ is defined as the application of most general unifier σ_u to $GF(D)$. By definition of a most general unifier, which works as usual eventhough we unify sets of types, there exists a substitution τ such that $\sigma = \tau \circ \sigma_u$.

Hence $\tau(RG(D)) = \tau(\sigma_u(GF(D))) = \sigma(GF(D)) = \sigma(GF(D)) \subset G$; thus $\tau(RG(D)) \subset G$, hence $RG(D) \sqsubset G$. ◇

Proposition 12 *If $D \subset D' \subset FL(G)$ with G a rigid grammar then $RG(D) \sqsubset RG(D') \sqsubset G$.*

PROOF : Because of proposition 11 both $RG(D)$ and $RG(D')$ exist. We have $D \subset D'$ and $D' \subset FL(RG(D'))$, so $D \subset FL(RG(D'))$; hence, by proposition 11 applied to D and $G = RG(D')$ (a rigid grammar) we have $RG(D) \sqsubset RG(D')$. ◇

Theorem 13 *The algorithm RG for learning rigid AB grammars converges in the sense of Gold (paragraph 1.7.1).*

PROOF : Take $D_i, i \in \omega$ an increasing sequence of sets of examples in $FL(G)$ enumerating $FL(G)$ — $\cup_{i\in\omega} D_i = FL(G)$:

$$D_1 \subset D_2 \subset \cdots D_i \subset D_{i+1} \cdots \subset FL(G)$$

Because of the proposition 11 for every $i \in \omega RG(D_i)$ exist and because of proposition 12 they define an increasing sequence of grammars w.r.t. $\sqsubset$ which by proposition 11 is bounded by G:

$$RG(D_1) \sqsubset RG(D_2) \sqsubset \cdots RG(D_i) \sqsubset RG(D_{i+1}) \cdots \sqsubset G$$

As they are finitely many grammars below G w.r.t. $\sqsubset$ (proposition 6) this sequence is stationary after a certain rank, say N, that is forall $n \geq N$ $RG(D_n) = RG(D_N)$.

We have $FL(RG(D_N)) = FL(G)$:

$FL(RG(D_N)) \supset FL(G)$ Let T be an FA-structure of $FL(G)$. Since $\cup_{i\in\omega} D_i = FL(G)$ there exists p such that $T \in FL(D_p)$.

- If $p < N$, because $D_p \subset D_N$, $T \in D_N$, and by proposition 9 $T \in FL(RG(D_N))$.
- If $p \geq N$, we have $RG(D_p) = RG(D_N)$ since the sequence of grammars is stationary after N. By proposition 9 we have $D_p \subset FL(RG(D_p))$ hence $T \in FL(RG(D_N)) = FL(RG(D_p))$.

In all cases, $T \in FL(RG(D_N))$.

$FL(RG(D_N)) \subset FL(G)$ Since $RG(D_N) \sqsubset G$, by proposition 7 we have

$$FL(RG(D_N)) \subset FL(G)$$

◇

1.7.4 Other cases

The learning problem covered by the RG algorithm is very simple and restricted.

Firstly, the class of grammars we are learning is quite restricted:

1. They are AB-grammars and not richer categorial grammars.
2. They are rigid, that is each word has a single syntactic behavior. This limitation is not too difficult to overcome: different occurrences of the same word corresponding to different syntactic behavior can to be distinguished. This is sound when the occurences correspond to really different words like *that* as a demonstrative and *that* as a complementizer, but it is less convincing when the word is the same like the transitive use of *eat* (*I ate an apple.*) and the absolutive use of *eat* (*I already ate*).

Secondly, we are using input structures which are not so easy to obtain, and which are probably too close to the output that we are looking for:

3. Parse structures or FA structures are much too precise, it would make more sense to have a tree structure, or some information on the tree structure of the sentence, but not the whole tree structure. It seems that dependency structures would be a good compromise between no structure, and a complete structure, and, from a practical viewpoint, it is possible to actually obtain such a corpus of exeamples, by shallow parsing or efficient (partial) parsing by dependency grammars. On the other hand, it is well known that strings are not enough to learn since grammatical rules apply to trees and not to sequences of words, and this has been confirmed by negative results in formal learing theory.

The base algorithm that we presented can be adapted in order to go beyond the limitations enumerated above.

1. Firstly an extension is to learn Lambek grammars from parse structures. It works and it is worth noticing that this class of languages does not have finite elasticity, but is learnable. [13] The same learning mechanism works for minimalist grammars when they are viewed as categorial grammars, also the determinism is lost. [38]. A strong generalization of this kind of results has been proved: reversible regular tree languages (and dependency grammars too) are learnable from positive examples [39].

2. An orthogonal extension is to consider k-valued AB grammars: in this later case, one has to try to unify types in all possible manners in order to have less than k types per word. This has been studied by Kanazawa [12].

3. Regarding the input structures, the simplest generalization is to consider unlabeled trees: then one has to try all possible labeling with $\backslash_e$ and $/_e$. Going even further one can learn from unstructured sentences that are simply sequences of words: once again this is done by considering all possible structures on such sentences. This has been studied by Kanazawa [12].

---

[13] Roberto Bonato. Uno studio sull'apprendibilità delle grammatiche di Lambek rigide — a study on learnability for rigid Lambek grammars. Tesi di Laurea & Mémoire de D.E.A, Università di Verona & Université Rennes 1, 2000.

[38] Roberto Bonato and Christian Retoré. Learning rigid lambek grammars and minimalist grammars from structured sentences. In Popelìnskỳ and Nepil [6], pages 23–34.

[6] Luboš Popelìnskỳ and Miloslev Nepil, editors. *Proceedings of the third workshop on Learning Language in Logic, LLL 01*, number FI-MU-RS-2001-08 in FI MU Report series, Strasbourg, September 2001. Faculty of Informatics – Masaryk University.

[39] Jérôme Besombes and Jean-Yves Marion. Identification of reversible dependency tree languages. In Popelìnskỳ and Nepil [6], pages 11–22.

[12] Makoto Kanazawa. *Learnable classes of categorial grammars*. Studies in Logic, Language and Information. FoLLI & CSLI, 1998. distributed by Cambridge University Press.

All these extensions considerably increase the complexity of the algorithm, as one can imagine, but nevertheless the existence of learning algorithms for categorial grammars is a good property that other formalisms for natural language syntax do not have.

Chapter 2

A logic for categorial grammars: Lambek's syntactic calculus

Our second chapter is a rather complete study of the Lambek calculus, which enables a completely logical treatment of categorial grammar.

We first present its syntax in full details, both with sequent calculus and natural deduction, and explain the relationship between these two presentations. Then we turn our attention to the normal forms for such proofs. Normalization and its dual namely interpolation are not only pleasant mathematical properties; they also are key properties for the correspondence between Lambek grammars and more familiar phrase structure grammars; we prove in detail the weak equivalence between context-free grammars and Lambek grammars.

Next we prove completeness for the Lambek calculus with linguistically natural models: in these models categories are interpreted as subsets of a free monoid; taking words or lexical items as generators of the free monoid really gives sense to the categorial approach.

We end with a description of the simple algorithm for computing the Montague semantics of a sentence from the semantics of the lexical items and the syntactic analysis. The straightforward correspondence between Montague semantics and categorial syntax is in our opinion an important advantage of (Lambek) categorial grammars.

2.1 Lambek syntactic calculus and Lambek grammars

We now turn our attention to Lambek calculus (L) and Lambek grammars (LCG) that were introduced in the seminal paper [8]: we strongly recommend to read it; the subsequent paper [40] is also worth reading in particular it presents a non associative version that we will not study here, but has been intensively used by Moortgat [10] and Morrill [41].

The limitations of AB grammars, and the endless quest of new rules (composition, type raising, Geach laws, etc.) is a way to explain the interest of Lambek calculus. Another is to place AB-grammar into a richer and more natural mathematical formalism.

A controversial but more interesting justification is the following: syntax is driven by resource consumption, which is neatly handled by resource conscious logics — the Lambek calculus being the first such logic. This viewpoint is not that far from Chomsky's minimalist program [22] as discussed in [42].

Lambek (categorial) grammars or LCGs for short proceed exactly as AB grammars do. A lexicon Lex provides each word with one or several types, constructed from the usual primitive types $\mathsf{P} = \{S, np, n, \ldots\}$ — noun phrases, nouns, sentences. Types are more or less the same as the one of AB grammars: the only difference is that Lambek types allow for a (non commutative) product or conjunction denoted by $\bullet$:

$$\mathsf{Lp} ::= \mathsf{P} \quad | \quad \mathsf{Lp} \setminus \mathsf{Lp} \quad | \quad \mathsf{Lp} / \mathsf{Lp} \quad | \quad \mathsf{Lp} \bullet \mathsf{Lp}$$

When introducing AB grammars, we already explained the intuitive meaning of $A \setminus B$ and B / A: an expression is of type $A \setminus B$ (resp. B / A) when it is waiting for an expression of type A on its left (resp. right) to form a compound expression of type B. An expression of type A followed by an expression B is of type $A \bullet B$, and product is related to $\setminus$ and $/$ by the following relations:

$$A \setminus (B \setminus X) = (B \bullet A) \setminus X \qquad\qquad (X / A) / B = X / (B \bullet A)$$

These relations looks like currying, but beware the order, which is required by the behavior of $\setminus$ and $/$: in the left equation both types require a sequence $a\,b$ on their left,

---

[8] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.

[40] Joachim Lambek. On the calculus of syntactic types. In Roman Jakobson, editor, *Structure of language and its mathematical aspects*, pages 166–178. American Mathematical Society, 1961.

[10] Michael Moortgat. Categorial type logic. In van Benthem and ter Meulen [1], chapter 2, pages 93–177.

[1] Johan van Benthem and Alice ter Meulen, editors. *Handbook of Logic and Language*. North-Holland Elsevier, Amsterdam, 1997.

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

[22] Noam Chomsky. *The minimalist program*. MIT Press, Cambridge, MA, 1995.

[42] Christian Retoré and Edward Stabler. Generative grammar in resource logics. 2(1):3–25, 2004.

and in the second equation both types require a sequence $b\,a$ on their right (with a, b of respective types A, B).

Recall that for AB grammars a sequence of words $w_1 \cdots w_n$, is of type u whenever there exists for each w_i a type t_i in $\mathrm{Lex}(w_i)$ such that $t_1 \cdots t_n \longrightarrow u$ with the following reduction patterns:

$$\forall u, v \in \mathsf{Lp} \qquad \begin{array}{ll} u(u \setminus v) \longrightarrow v & (\setminus_e) \\ (v \,/\, u)u \longrightarrow v & (/_e) \end{array}$$

Here the logical aspect of these rules — they look like modus ponens — will be emphasized by allowing for other rules, so that $\setminus$ and $/$ will really by implications (and $\bullet$ will be their associated conjunction). Accordingly $\longrightarrow$ will be written $\vdash$, and our first objective is to define this logical calculus: for the time being we only know the modus ponens of the non commutative implications $\setminus$ and $/$. Therefore we simply replace $\longrightarrow$ with $\vdash$ to obtain the following definition: a sequence of words (or terminals) $w_1 \cdots w_n$ is of type u whenever there exists for each w_i a type t_i in $\mathrm{Lex}(w_i)$ such that $t_1 \cdots t_n \vdash u$, where $\vdash$ is the deductive relation of the Lambek calculus to be define thereafter. The generated language or the set of correct sentences is the set of sequences of type S.

2.2 Natural deduction for Lambek calculus

To the best of my knowledge natural deduction for Lambek has mainly be studied by van Benthem [43] one of the first paper being [44].

2.2.1 In Prawitz style

Maybe the simplest way to define product free Lambek calculus is natural deduction in tree like setting :

this rule requires at least two free hyp.

$$\begin{array}{l} A \text{ left most free hyp.} \\ \quad \ldots [A] \ldots\ldots \\ \qquad \vdots \\ \qquad B \\ \quad \dfrac{}{A \setminus B} \setminus_i \text{ binding } A \end{array} \qquad\qquad \dfrac{\begin{array}{cc} \Delta & \Gamma \\ \vdots & \vdots \\ A & A \setminus B \end{array}}{B} \setminus_e$$

[43] Johan 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.

[44] Johan van Benthem. Categorial grammars and lambda calculus. In D. Skordev, editor, *Mathematical logic and its Applications*. Plenum Press, 1987.

this rule requires at least two free hyp.

$$\dfrac{\begin{array}{c}A \text{ right most free hyp.}\\ \ldots\ldots[A]\ldots\\ \vdots\\ B\end{array}}{B/A}\,/_i \text{ binding } A \qquad \dfrac{\begin{array}{cc}\Gamma & \Delta\\ \vdots & \vdots\\ B/A & A\end{array}}{B}\,/_e$$

These deductions clearly extend the derivation trees of AB grammars. AB simplification or residuation rules are two of the rules of the system, the rules $\backslash_e$ and $/_e$; the other two being the corresponding introduction rules. The fact that these rules are particular cases of the rules for intuitionistic logic confirms that the fraction symbols $\backslash$ and $/$ can be viewed as implications.

It should be observed that as opposed to natural deduction for intuitionistic logic, there is no need to specify which hypothesis A is cancelled by an $/_i$ or $\backslash_i$ introduction rule. Indeed in the first case it is the left most free hypothesis, and in the second case it is the right most free hypothesis. As a consequence the formal structure of a deduction is a plain (binary/unary) tree with leaves labeled with formulae and with nodes labelled by rules : binary nodes are labelled with either $/_e$ or $\backslash_e$ and unary nodes with either $/_i$ or $\backslash_i$. Such a plain tree is enough to reconstruct the deduction, i.e. which hypothesis are free or not and which hypothesis is cancelled by which rule. This remark is the basis of the study of [45] ; parse structure of a Lambek grammar are defined to be natural deduction trees, and they are studied as tree languages.

Product Lambek calculus admits a product which corresponds to the implications by the usual rules of currying given above. The product is often skipped out of natural deduction presentation of the Lambek calculus. There is no need to do so, but it is true that these rules are less natural, because of the order on hypotheses:

$$\dfrac{\begin{array}{cc}\Delta & \Gamma\\ \vdots & \vdots\\ A & B\end{array}}{A\bullet B}\,\bullet_i \qquad \dfrac{\begin{array}{c}\Delta\\ \vdots\\ A\bullet B\end{array} \qquad \begin{array}{l}\text{——} : \text{without free hyp.}\\ \ldots[A]_\alpha \ldots \text{——} \ldots [B]_\alpha \ldots\\ \vdots\\ C\end{array}}{C}\,\bullet_e(\alpha) \text{ binding } A \text{ and } B$$

The main problem is that in order to apply the product elimination rule there should be no free hypothesis in between the two cancelled assumptions, A and B, and that the order of the premises after the rule is not anymore the left right order. Another problem

[45] Hans-Jörg Tiede. Lambek calculus proofs and tree automata. In Michael Moortgat, editor, *Logical Aspects of Computational Linguistics, LACL'98, selected papers*, number 2014 in LNCS/LNAI. Springer-Verlag, 2001.

is that, as we shall see, proof-normalization or rather the sub-formula property is more problematic with the product.

Also observe that there can be several consecutive A and B free hypothesis, so that a labelling of the cancelled hypotheses is needed for this rule: natrual deduction are not anymore plain trees.

This kind of natural deduction rules were first introduced by S. Abramsky in [46] for multiplicative linear logic, but in this commutative case the problem of hypothesis-order vanishes.

2.2.2 In Gentzen style

One can also define natural deduction in a "sequent" presentation that is in specifying at each node what the free hypotheses are; this formulation is possibly clearer in particular when one uses the product. Nevertheless this presentation defines exactly the same logical calculus as the natural deduction in tree like format given above: the proofs of the two systems are isomorphic.

Although we use sequents, that are expressions $A_1,\ldots,A_n \vdash C$, this calculus is by no means a sequent calculus: there are no left rules, no cut rule, and the notion of normal proof (for having the sub-formula property) is completely different.

$$\frac{\Gamma \vdash A \quad \Delta \vdash A\backslash B}{\Gamma,\Delta \vdash B}\backslash_e \qquad \frac{A,\Gamma \vdash C}{\Gamma \vdash A\backslash C}\backslash_i \quad \Gamma \neq \varepsilon$$

$$\frac{\Delta \vdash B/A \quad \Gamma \vdash A}{\Delta,\Gamma \vdash B}/_e \qquad \frac{\Gamma,A \vdash C}{\Gamma \vdash C/A}/_i \quad \Gamma \neq \varepsilon$$

$$\frac{\Delta \vdash A\bullet B \quad \Gamma,A,B,\Gamma' \vdash C}{\Gamma,\Delta,\Gamma' \vdash C}\bullet_e \qquad \frac{\Delta \vdash A \quad \Gamma \vdash B}{\Delta,\Gamma \vdash A\bullet B}\bullet_i$$

$$\frac{}{A \vdash A}\ axiom$$

[46] Samson Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111:3–57, 1993.

2.3 An example

Here we take up again our small example of an Italian lexicon:

Word	Type(s)
cosa	$(S/(S/np))$
guarda	(S/vp)
passare	(vp/np)
il	(np/n)
treno	n

Remember that the sentence *Cosa guarda passare* could not be analyzed in AB grammars, because the transitivity of $/$ was not a rule of AB grammars. Let us show that it can be analyzed with the Lambek calculus (we use Natural Deduction in Gentzen style):

$$\dfrac{(S/(S/np)) \vdash (S/(S/np)) \qquad \dfrac{\dfrac{(S/vp) \vdash (S/vp) \qquad \dfrac{(vp/np) \vdash (vp/np) \quad np \vdash np}{(vp/np), np \vdash vp}\,/_e}{(S/vp), (vp/np), np \vdash S}\,/_i}{(S/vp), (vp/np) \vdash S/np}\,/_e}{(S/(S/np)), (S/vp), (vp/np) \vdash S}$$

This example relies on composition for $/$, which is not provable within AB grammars. Composition is established by using a fake np which is then abstracted by an introduction rule : to make a comparison with Chomsky's theories [47,22] this fake np is corresponds to a trace and the introduction rule to a movement.

Similarly it can be shown that it is possible to only assign the type $(np \setminus S)/np$ to a transitive verb, and to construct object relatives with *whom/that* having the type $(n \setminus n)/(S/np)$ — which could not be done in AB grammars, see paragraph **??**. This results form the possibility to rearrange brackets in the Lambek calculus: $(a \setminus b)/c \vdash a \setminus (b/c)$, etc.

Finally it can be easily seen that one has $x \vdash (z/x) \setminus z$ and $x \vdash z/(x \setminus z)$ for every categories x and z. This is interesting from a semantic viewpoint: an np (an individual) can be viewed as a $(S/np) \setminus S$ or $S/(np \setminus S)$ (a function form one place predicates to truth values) that is the set of all the properties of this individual.

---

[47] Noam Chomsky. *Syntactic structures*. Janua linguarum. Mouton, The Hague, 1957.

[22] Noam Chomsky. *The minimalist program*. MIT Press, Cambridge, MA, 1995.

2.4 Sequent calculus

Here are the rules of the Lambek calculus in Sequent Calculus, as given in the original paper [8]. Although it also handles expressions $A_1,\ldots,A_n \vdash C$, let us insists that it is different from Natural Deduction in sequent style given above: for instance the modus ponens or residuation laws of the AB grammars are not rules of this system (they are just derivable) and the notion of a normal proof is very different.

$$\frac{\Gamma,B,\Gamma' \vdash C \quad \Delta \vdash A}{\Gamma,\Delta,A\setminus B,\Gamma' \vdash C}\setminus_h \qquad \frac{A,\Gamma \vdash C}{\Gamma \vdash A\setminus C}\setminus_i \quad \Gamma \neq \varepsilon$$

$$\frac{\Gamma,B,\Gamma' \vdash C \quad \Delta \vdash A}{\Gamma,B/A,\Delta,\Gamma' \vdash C}/_h \qquad \frac{\Gamma,A \vdash C}{\Gamma \vdash C/A}/_i \quad \Gamma \neq \varepsilon$$

$$\frac{\Gamma,A,B,\Gamma' \vdash C}{\Gamma,A\bullet B,\Gamma' \vdash C}\bullet_h \qquad \frac{\Delta \vdash A \quad \Gamma \vdash B}{\Delta,\Gamma \vdash A\bullet B}\bullet_i$$

$$\frac{\Gamma \vdash A \quad \Delta_1,A,\Delta_2 \vdash B}{\Delta_1,\Gamma,\Delta_2 \vdash B}\,cut \qquad \frac{}{A \vdash A}\,axiom$$

Here is an obvious proposition, known as η-expansion:

Proposition 14 *Each axioms $A \vdash A$ can be derived from axioms $p \vdash p$, with p being a primitive type (and the proof does not use the cut rule).*

The polarity of an occurrence of a propositional variable p in a formula F is defined as usual:

- p is positive in p
- if p is positive in A, then
  - p is positive in $X\bullet A$, $A\bullet X$, $X\setminus A$, A/X
  - p is negative in $A\setminus X$, X/A
- if p is negative in A, then
  - p is negative in $X\bullet A$, $A\bullet X$, $X\setminus A$, A/X

[8] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.

– p is positive in $A \setminus X, \quad X / A$

The polarity of an occurrence of a propositional variable p in a sequent $\Gamma \vdash C$ is:

- if p is in C, the polarity of p in C
- if p is in a formula G of Γ, the opposite of the polarity of p in G.

If a proof only use atomic axioms (this is always possible, as said above) that are $p \vdash p$ with p a primitive type, then one can follow these two occurrences of p, one being negative and the other positive and none of the rules changes the polarity of an occurrence of a primitive type. The two occurrences of p either lead to a cut formula (the one that disappear in the cut rule) or to the conclusion sequent. Now observe that the cut rule cancels a formula in positive position (on the right) with the same formula in negative position (on the left), so that the same number of positive and negative occurrences of p disappear. Consequently:

Proposition 15 *Each propositional variable has exactly the same number of positive and negative occurrences in a provable sequent.*

2.5 An example

Here is an example of a proof in sequent calculus, corresponding to the analysis of *Cosa guarda passare* already given above but in natural deduction format . It is somehow less natural, but has other advantages, like an easier sub-formula property.

$$\dfrac{S \vdash S \qquad \dfrac{\dfrac{S \vdash S \quad vp \vdash vp}{S / vp, vp \vdash S}\,/_h \quad np \vdash np}{S / vp, vp / np, np \vdash S}\,/_h}{\dfrac{S \vdash S \qquad S / vp, vp / np \vdash S / np}{(S / (S / np)), S / vp, vp / np \vdash S}\,/_i}\,/_h$$

2.6 Equivalence of sequent calculus and natural deduction

As we will see, this equivalence is absolutely clear as far as provability is concerned. In fact there is a correspondence for proofs as well, but it is not a straightforward isomorphism [14].

[14] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

As introduction rules are common to both formalisms, we just need to mimic elimination rules $\diamond_e$ in sequent calculus and left rules $\diamond_h$ in natural deduction, and by induction on the height of the proofs the equivalence of both formalisms follows. This section is an easy adaptation of the results in [14] for intuitionistic logic.

2.6.1 From natural deduction to sequent calculus

It is possible to do "better" than the translation we provide here; indeed, when the natural deduction is normal, one can manage to obtain a cut-free proof, and this better translation is implicitely used when one uses proof nets for λ-calculus see e.g. [48,49]

Replace:	with:
$\dfrac{\Delta \vdash A \quad \Gamma \vdash A \setminus B}{\Delta, \Gamma \vdash B}\, \setminus_e$	$\dfrac{\Gamma \vdash A \setminus B \qquad \dfrac{\Delta \vdash A \quad \dfrac{}{B \vdash B}\, ax}{\Delta, A \setminus B \vdash B}\, \setminus_h}{\Delta, \Gamma \vdash B}\, cut$
$\dfrac{\Gamma \vdash B / A \quad \Delta \vdash A}{\Gamma, \Delta \vdash B}\, /_e$	$\dfrac{\Gamma \vdash B / A \qquad \dfrac{\Delta \vdash A \quad \dfrac{}{B \vdash B}\, ax}{B / A, \Delta \vdash B}\, /_h}{\Gamma, \Delta \vdash B}\, cut$
$\dfrac{\Gamma \vdash A \bullet B \quad \Delta, A, B, \Theta \vdash C}{\Delta, \Gamma, \Theta \vdash C}\, \bullet_e$	$\dfrac{\Gamma \vdash A \bullet B \qquad \dfrac{\Delta, A, B, \Theta \vdash C}{\Delta, A \bullet B, \Theta \vdash C}\, \bullet_h}{\Delta, \Gamma, \Theta \vdash C}\, cut$

2.6.2 From sequent calculus to natural deduction

By induction on the height of a sequent calculus proof, let us see that it can be turned into a natural deduction. As above, we will not exhibit a translation from cut free proofs to normal deductions, although it is possible.

- If the proof consists in an axiom, its translation is obvious.

[48] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[49] Philippe de Groote and Christian Retoré. Semantic readings of proof nets. In Geert-Jan Kruijff, Glyn Morrill, and Dick Oehrle, editors, *Formal Grammar*, pages 57–70, Prague, 1996. FoLLI.

- If the proof ends with an introduction rule, $\backslash_i$, $/_i$ or $\bullet_i$ by induction hypothesis we have a deduction of the premise(s) and as these rules also exist in natural deduction the translation follows.

- If the proof ends with an $\backslash_h$ rule:

$$\dfrac{\begin{array}{cc}\vdots\,\gamma & \vdots\,\delta \\ \Gamma,B,\Gamma' \vdash C & \Delta \vdash A\end{array}}{\Gamma,\Delta,A\backslash B,\Gamma' \vdash C}\,\backslash_h$$

  then by induction hypothesis we have two natural deduction proofs, γ^* of $\Gamma,B,\Gamma' \vdash C$ and δ^* of $\Delta \vdash A$ and a translation of the whole proof is:

$$\begin{array}{ccc} & \Delta & \\ & \vdots\,\delta & \\ & A \quad A\backslash B & \\ \Gamma & \overline{\qquad B \qquad}\,\backslash_e & \Gamma' \\ & \vdots\,\gamma & \\ & C & \end{array}$$

- If the proof ends with $/_h$ we proceed symmetrically.

- If the proof ends with $\bullet_h$:

$$\dfrac{\begin{array}{c}\vdots\,\gamma \\ \Gamma,A,B,\Gamma' \vdash C\end{array}}{\Gamma,A\bullet B,\Gamma' \vdash C}\,\bullet_h$$

  by induction hypothesis we have a proof γ^* of $\Gamma,A\bullet B,\Gamma' \vdash C$ and a translation is the following:

$$\dfrac{\begin{array}{ccc}\Gamma & A\,B & \Gamma' \\ & \vdots\,\gamma^* & \\ A\bullet B & C & \end{array}}{C}\,\bullet_e$$

- If the proof ends with a cut:

$$\dfrac{\begin{array}{cc}\vdots\,\gamma & \vdots\,\delta \\ \Gamma \vdash X & \Delta,X,\Delta' \vdash C\end{array}}{C}\,cut$$

  by induction hypothesis we have two natural deductions $\gamma^*$of $\Gamma \vdash X$ and δ^* of $\Delta,X,\Delta' \vdash C$ and a translation is:

$$\begin{array}{c} \Gamma \\ \vdots \\ \gamma^* \\ \Delta \quad X \quad \Delta' \\ \vdots \\ \delta \\ \vdots \\ C \end{array}$$

2.7 The empty sequence

In the introduction rules we have assumed that the context contains at least two formulae: there fore the context afterwards is never empty. By case inspection we see that this guarantees that the context of a sequent (the sequence on the left of $\vdash$) never is empty in a proof.

This is justified by the intended meaning of the connectives. Indeed by assigning the type $A \setminus B$ to a word or an expression e, we mean that an expression a of type A is *required* before e to obtain an expression ae of type B. This would fail without the "no empty sequence" requirement.

To explain this, let L1 be the calculus L without this restriction. Indeed, assume A is a tautology of L1, i.e. $\vdash_{L1} A$ (*); now let Γ be a sequence of type $A \setminus B$, that is $\Gamma \vdash_{L1} A \setminus B$ (**). Then from (*) and (**) we can infer by $\setminus_e$ the sequent $\Gamma \vdash_{L1} B$ without any sequence preceding Γ. This can actually happen in natural language; indeed some expression, including all modifiers do have such a tautology type, like $X \setminus X$.

For instance, a natural type for English adjectives is n / n and thus *very* gets the type $(n / n) / (n / n)$: when applied to an adjective on its right, one obtains an adjective phrase. Without the exclusion of the empty sequence, one is able to analyze in L1 the expression *"a very book"* as a noun phrase: indeed the adjective following *very* can be provided by the empty sequence, since n / n is derivable in L1. Let us give the proof in L1 with a natural deduction in Prawitz style:

$$\dfrac{\text{a} : np / n \qquad \dfrac{\text{very} : (n / n) / (n / n) \qquad \dfrac{[n]_\alpha}{n / n} /_i \alpha}{n / n} /_e \qquad \text{book} : n}{np} /_e$$

One may wonder why such a requirement was not needed in AB grammars. As AB grammars only contains elimination rules, no hypothesis is cancelled during a derivation, and as there are hypotheses at the beginning of every sub-analysis (the types of the words in the analyzed sequence) there always is at least one hypothesis.

2.8 Normalization of natural deduction

This section is also an easy adaptation of similar results presented in [14].

2.8.1 Normalization for product free Lambek calculus

A natural deduction is said to be normal whenever it does not contain an introduction rule followed by an elimination rule. There are two such possible configurations:

$$\dfrac{\dfrac{\dfrac{\ldots\ldots[A]_\alpha\ldots}{\vdots\,\delta'}}{B}\,/_i\alpha \qquad \begin{matrix}\Delta\\ \vdots\,\delta\\ A\end{matrix}}{B}\,/_e \qquad\qquad \dfrac{\begin{matrix}\Delta\\ \vdots\,\delta\\ A\end{matrix} \qquad \dfrac{\ldots[A]_\alpha\ldots\ldots}{\vdots\,\delta'}}{B}\,\backslash_i\alpha}{B}\,\backslash_e$$

Whenever such a configuration appears, it can be reduced as follows:

1. find the hypothesis A which has been cancelled in the proof δ' of B under some hypotheses including A
2. replace this hypothesis with the proof δ of A

So the configurations above reduce to:

$$\begin{matrix}\Delta\\ \vdots\,\delta\\ \ldots\ldots A\ldots\\ \vdots\,\delta'\\ B\end{matrix} \qquad\qquad \begin{matrix}\Delta\\ \vdots\,\delta\\ \ldots A\ldots\\ \vdots\,\delta'\\ B\end{matrix}$$

Proposition 16 *Natural deduction for L without product enjoys strong normalization, that is there are no infinite reduction sequences.*

PROOF : Observe that the size of the proof decreases in each reduction step. ◇

Proposition 17 *Normalization is a locally confluent process.*

PROOF : If a proof d contains two redexes, they correspond to two elimination rules e' and e'' between sub-proofs corresponding to a function f' applied to an argument a' and to a function f'' applied to an argument a''. One of the following case applies:

- e'' is in a'

[14] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

- e'' is in f'
- e' is in a''
- e' is in f''
- e' and e'' can not be compared.

Assume we reduce e'. The redex e'' which is not reduced possesses a unique trace $\vec{e}''$ in the reduced proof d'. Symmetrically if we reduce e'' the redex e' which is not reduced possesses a unique trace $\vec{e}$" in d''. If in d' we reduce $\vec{e}''$ we obtain a proof d''' but if in d'' we reduce $\vec{e}'$ we also obtain d'''. $\diamond$

It is easily shown by induction on the proofs that whenever a natural deduction is normal (that is without such configuration) each formula is a sub-formula of a free hypothesis or of the conclusion. More precisely. In order to establish this, let us introduce the notion of principal branch.

Let us call a *principal branch* leading to F a sequence $H_0, \ldots, H_n = F$ of formulae of a natural deduction tree such that:

- H_0 is a free hypothesis
- H_i is the principal premise — the one carrying the eliminated symbol — of an elimination rule whose conclusion is H_{i+1}
- H_n is F

Proposition 18 *Let d be a normal natural deduction (without product), then:*

1. *if d ends with an elimination then there is a principal branch leading to its conclusion*
2. *each formula in d is the sub-formula of a free hypothesis or of the conclusion*

PROOF : By induction on d.

axiom If d is an axiom, (1) and (2) hold.

$\backslash_i$ **introduction** (1) holds by vacuity. Assume d is made out of d' by the introduction $\backslash_i$ rule: by induction hypothesis each formula in d' is a sub-formula of A, Γ (the free hypotheses under which B is proved) or a sub-formula of B; so it is true that each formula in d is a sub-formula of $\Gamma, A \setminus B$, since A and B are sub-formulae of $A \setminus B$.

$\backslash_e$ **elimination** Assume d is an elimination rule $\backslash_e$ applied to:

- d' with conclusion A and free hypotheses Γ
- d'' with conclusion $A \setminus B$ and free hypotheses Δ

(1) Since d is normal the last rule of d'' is an elimination: indeed, if it were an introduction rule then it would be a $\backslash_i$ introduction making a redex with the final elimination in d. As d'' ends with an elimination, by induction hypothesis, there is a principal branch leading from H_0 in Δ to $A \backslash B$, so d contains a principal branch leading to its conclusion B.

(2) By induction hypothesis

- all formulae in d' are sub-formula of A or Γ (the free hypotheses under which A is proved)
- all formulae in d'' are sub-formulae of $\Delta, A \backslash B$.

Because of the principal branch of d' leading to $A \backslash B$, the conclusion $A \backslash B$ of d' is a sub-formula of some H_0 in Δ. Thus each formulae in d is a sub-formula of Γ, Δ hence of Γ, Δ, B

$/_i$ **introduction** as $\backslash_i$ introduction.

$/_e$ **elimination** as $\backslash_e$ elimination

◇

Here is a proposition of [50] that we shall use to prove that every context free grammar is weakly equivalent to a Lambek grammar. Let us call the order $o(A)$ of a formula A the number of alternating implications:

- $o(p) = 0$ when p is a primitive type
- $o(A \backslash B) = \max(o(A) + 1, o(B))$
- $o(B / A) = \max(o(A) + 1, o(B))$

Proposition 19 *A provable sequent $A_1, \ldots, A_n \vdash p$ of the product free Lambek calculus with $o(A_i) \leq 1$ and p a primitive type is provable with $\backslash_e$ and $/_e$ only — in other words AB derivations and L derivations coincide when types are of order at most one.*

PROOF : We proceed by contradiction, so we assume that the normal deduction contains an introduction rule, and so there is a lowest introduction rule — one without any introduction rule below.

Let us consider an arbitrary lowest introduction I.

- If the chosen lowest introduction I is an $\backslash_i$ introduction leading from y to $b \backslash y$. This introduction cannot be the last rule, because the conclusion is a primitive type p. So this rule is followed by a an elimination rule E, and there are three possibilities:

[50] Joel M. Cohen. The equivalence of two concepts of categorial grammars. *Information and Control*, 1967.

- If $b \backslash y$ is the principal premisse of the elimination rule E, then the rule E is an $\backslash_e$ elimination rule other premisse b; we then have a redex I, E and this conflicts with the deduction being normal.
- If $b \backslash y$ is not the principal premisse of the elimination rule E, then E is either an $\backslash_e$ elimination rule with principal premisse being $(b \backslash y) \backslash z$ or an $/_e$ elimination rule with principal premisse $z / (b \backslash y)$. In both cases the principal premisse is of order at least two. This conflicts with d enjoying the subformula property which is forced by d being normal (previous proposition 18).

• If the chosen lowest introduction I is an $/_i$ rule, the argument is symmetrical.

Therefore there is no lowest introduction, hence no introduction at all. ◇

2.8.2 Normalization and Lambek calculus with product

We have to introduce commutative reductions for the product, otherwise is is possible that a normal proof does not satisfy the sub-formula property:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A \vdash A \qquad B \vdash B}{A, B \vdash A \bullet B}\bullet_i \quad D \vdash D}{A, B, D \vdash (A \bullet B) \bullet D}\bullet_i}{A, B \vdash (A \bullet B) \bullet D / D}/_i \quad A \bullet B \vdash A \bullet B}{A \bullet B \vdash (A \bullet B) \bullet D / D}\bullet_e \quad D \vdash D}{A \bullet B, D \vdash (A \bullet B) \bullet D}/_e$$

Let us mention that this can be achieved by adding some "commutative contractions" which basically consists in putting product elimination rules as high as possible (just after the cancelled hypotheses A and B have met), and then rearranging the sub-trees made of product elimination rules with a kind of associativity so that the eliminated product never is the conclusion of another product elimination. This is too lengthy for what it is since this kind of result can also be deduced from the correspondence with sequent calculus.

2.9 Cut-elimination for sequent calculus

Cut elimination is the process under which a proof is turned into a proof of the same sequent *without any cut rule* — in other words, the cut rule is redundant.

This property is famous for classical or intuitionistic logic see e.g. [14], and regarding L, it was originally proved in [8].

Cut elimination has an important consequence, that we state before proving cut elimination:

Proposition 20 *In a cut-free proof of $A_1,\ldots,A_n \vdash A_{n+1}$ every formula of every sequent is a sub-formula of some formula A_i ($1 \leq i \leq n+1$).*

PROOF : By case inspection it is easily observed that every rule of the sequent calculus but the cut rule, satisfies the property that every formula in its premise sequent(s) is a sub-formula of some formula in its conclusion sequent. ◇

We give a syntactic proof of cut elimination (while models could be used as well): it is lengthy, tedious and without surprise, but one has to see this kind of proof at least once.

We proceed by induction on (d,r) with $(d,r) < (d',r')$ if $d < d'$ or $d = d' \wedge r < r'$ where r is the number of rules of the proof, and d the maximal degree of a cut, assumed to be 0 when there is no cut.

The degree of a formula is the height of the sub-formula tree, and the degree of a cut is the degree of the cut-formula, the one which disappears during the cut rule.

First, let us see that we can assume that the last rule R is a cut rule and the only cut. Indeed, otherwise we can transform all the subproofs above this rule R (because they contain less rules) into cut-free proof and then apply R, obtaining a cut-free proof.

Therefore we can assume that R is the only cut hence of maximal degree d.

$$\dfrac{\dfrac{\vdots\,\gamma}{\Gamma \vdash X}\,R^a \qquad \dfrac{\vdots\,\delta}{\Delta, X, \Delta' \vdash C}\,R^f}{\Delta,\Gamma,\Delta' \vdash C}\,cut\ d$$

Notice that because the last rule is the only cut, neither R^a nor R^f is a cut rule.

We are going to explore all possible values for R^a and R^f, and whatever these rules are, at least one of the following cases apply:

1. One of R^a or R^f is an axiom: both the cut and the axiom are suppressed.

2. R^a does not create the cut-formula, — so $R^a \neq \bullet_i, \backslash_i, /_i$. In this case it is possible to apply R^a after the cut. We can apply the induction hypothesis to the proof(s) minus R^a since its (their) number of rules is smaller: it can be turned into a cut-free proof. Reapplying R^a we obtain a cut free proof.

[14] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

[8] Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.

3. If R^f does not create the cut formula, we proceed symmetrically.

4. If both R^a and R^f create the cut formula, then this cut of degree d is replaced with two cut of degree strictly smaller. Hence, the maximal degree of a cut is strictly smaller (as the last rule was the only cut) and by induction hypothesis we are done.

We only describe the cases for $\backslash$ because the ones for $/$ are strictly symmetrical.

1 R^a or R^f is an axiom The final cut can be supressed.

$$\dfrac{X \vdash X \qquad \begin{array}{c}\vdots\,\delta\\ \Gamma, X, \Delta \vdash C\end{array}}{\Gamma, X, \Delta \vdash C}\ cut \qquad \text{reduces to} \qquad \begin{array}{c}\vdots\,\delta\\ \Gamma, X, \Delta \vdash C\end{array}$$

2 R^a does not create X, the cut formula		
R^a	Before reduction	After reduction
$\bullet_h$	$\dfrac{\dfrac{\begin{array}{c}\vdots\,\gamma\\ \Gamma, A, B, \Gamma' \vdash X\end{array}}{\Gamma, A \bullet B, \Gamma' \vdash X}\bullet_h \qquad \begin{array}{c}\vdots\,\delta\\ \Delta, X, \Delta' \vdash C\end{array}}{\Delta, \Gamma, A \bullet B, \Gamma', \Delta' \vdash C}\ cut\ d$	$\dfrac{\dfrac{\begin{array}{c}\vdots\,\gamma\\ \Gamma, A, B, \Gamma' \vdash X\end{array} \qquad \begin{array}{c}\vdots\,\delta\\ \Delta, X, \Delta' \vdash C\end{array}}{\Delta, \Gamma, A, B, \Gamma', \Delta' \vdash C}\ cut\ d}{\Delta, \Gamma, A \bullet B, \Gamma', \Delta' \vdash C}\bullet_h$
$\backslash_h$	$\dfrac{\dfrac{\begin{array}{c}\vdots\,\delta\\ \Delta, B, \Delta'' \vdash X\end{array} \qquad \begin{array}{c}\vdots\,\delta'\\ \Delta' \vdash A\end{array}}{\Delta, \Delta', A \backslash B, \Delta'' \vdash X}\backslash_h \qquad \begin{array}{c}\vdots\,\gamma\\ \Gamma, X, \Gamma' \vdash C\end{array}}{\Gamma, \Delta, \Delta', A \backslash B, \Delta'', \Gamma' \vdash C}\ cut\ d$	$\dfrac{\dfrac{\begin{array}{c}\vdots\,\delta\\ \Delta, B, \Delta'' \vdash X\end{array} \qquad \begin{array}{c}\vdots\,\gamma\\ \Gamma, X, \Gamma' \vdash C\end{array}}{\Gamma, \Delta, B, \Delta'', \Gamma' \vdash C}\ cut\ d \qquad \begin{array}{c}\vdots\,\delta'\\ \Delta' \vdash A\end{array}}{\Gamma, \Delta, \Delta', A \backslash B, \Delta'', \Gamma' \vdash C}\backslash_h$

3 R^f **does not create** X**, the cut formula**		
R^f	Before reduction	After reduction
$\bullet_h$	$\dfrac{\dfrac{\vdots\,\delta}{\Delta\vdash X} \quad \dfrac{\vdots\,\gamma}{\Gamma,X,\Gamma',A,B,\Gamma''\vdash C}\,\bullet_h}{\Gamma,\Delta,\Gamma',A\bullet B,\Gamma''\vdash C}\,cut\ d$	$\dfrac{\dfrac{\vdots\,\delta}{\Delta\vdash X} \quad \dfrac{\vdots\,\gamma}{\Gamma,X,\Gamma',A,B,\Gamma''\vdash C}\,cut\ d}{\Gamma,\Delta,\Gamma',A,B,\Gamma''\vdash C}\,\bullet_h$
$\backslash_h$	$\dfrac{\dfrac{\vdots\,\delta}{\Delta\vdash X} \quad \dfrac{\vdots\,\gamma}{\Gamma,B,\Gamma',X,\Gamma''\vdash C} \quad \dfrac{\vdots\,\theta}{\Theta\vdash A}\backslash_h}{\Gamma,\Theta,A\backslash B,\Gamma',X,\Gamma''\vdash C}\,cut\ d}{\Gamma,\Theta,A\backslash B,\Gamma',\Delta,\Gamma''\vdash C}$	$\dfrac{\dfrac{\vdots\,\delta}{\Delta\vdash X} \quad \dfrac{\vdots\,\gamma}{\Gamma,B,\Gamma',X,\Gamma''\vdash C}\,cut\ d \quad \dfrac{\vdots\,\theta}{\Theta\vdash A}}{\Gamma,B,\Gamma',\Delta,\Gamma''\vdash C}\backslash_h}{\Gamma,\Theta,A\backslash B,\Gamma',\Delta,\Gamma''\vdash C}$
$\backslash_h$	$\dfrac{\dfrac{\vdots\,\delta}{\Delta\vdash X} \quad \dfrac{\vdots\,\gamma}{\Gamma,B,\Gamma'''\vdash C} \quad \dfrac{\vdots\,\gamma'}{\Gamma',X,\Gamma''\vdash A}\backslash_h}{\Gamma,\Gamma',X,\Gamma'',A\backslash B,\Gamma'''\vdash C}\,cut\ d}{\Gamma,\Gamma',\Delta,\Gamma'',A\backslash B,\Gamma'''\vdash C}$	$\dfrac{\dfrac{\vdots\,\gamma}{\Gamma,B,\Gamma'''\vdash C} \quad \dfrac{\dfrac{\vdots\,\delta}{\Delta\vdash X} \quad \dfrac{\vdots\,\gamma'}{\Gamma',X,\Gamma''\vdash A}\,cut d}{\Gamma',\Delta,\Gamma''\vdash A}\backslash_h}{\Gamma,\Gamma',\Delta,\Gamma'',A\backslash B,\Gamma'''\vdash C}$
$\bullet_i$	$\dfrac{\dfrac{\vdots\,\delta}{\Delta\vdash X} \quad \dfrac{\vdots\,\gamma}{\Gamma,X,\Gamma'\vdash A} \quad \dfrac{\vdots\,\theta}{\Theta\vdash B}\bullet_i}{\Gamma,X,\Gamma',\Theta\vdash A\bullet B}\,cut\ d}{\Gamma,\Delta,\Gamma',\Theta\vdash A\bullet B}$	$\dfrac{\dfrac{\vdots\,\delta}{\Delta\vdash X} \quad \dfrac{\vdots\,\gamma}{\Gamma,X,\Gamma'\vdash A}\,cut\ d \quad \dfrac{\vdots\,\theta}{\Theta\vdash B}}{\Gamma,\Delta,\Gamma'\vdash A}\bullet_i}{\Gamma,\Delta,\Gamma',\Theta\vdash A\bullet B}$
$\backslash_i$	$\dfrac{\dfrac{\vdots\,\delta}{\Delta\vdash X} \quad \dfrac{\vdots\,\gamma}{A,\Gamma,X,\Gamma'\vdash B}\backslash_i}{\Gamma,X,\Gamma'\vdash A\backslash B}\,cut\ d}{\Gamma,\Delta,\Gamma'\vdash A\backslash B}$	$\dfrac{\dfrac{\vdots\,\delta}{\Delta\vdash X} \quad \dfrac{\vdots\,\gamma}{A,\Gamma,X,\Gamma'\vdash B}\,cut\ d}{A,\Gamma,\Delta,\Gamma'\vdash B}\backslash_i}{\Gamma,\Delta,\Gamma'\vdash A\backslash B}$

4 Both R^a and R^f create the cut-formula		
	Before reduction	After reduction
$\bullet$	$\dfrac{\dfrac{\begin{array}{c}\vdots\,\delta\\ \Delta\vdash U\end{array}\quad\begin{array}{c}\vdots\,\theta\\ \Theta\vdash V\end{array}}{\Delta,\Theta\vdash U\bullet V}\bullet_i\quad\dfrac{\begin{array}{c}\vdots\,\gamma\\ \Gamma,U,V,\Gamma'\vdash C\end{array}}{\Gamma,U\bullet V,\Gamma'\vdash C}\bullet_h}{\Gamma,\Delta,\Theta,\Gamma'\vdash C}\,cut\ d$	$\dfrac{\begin{array}{c}\vdots\,\delta\\ \Delta\vdash U\end{array}\quad\dfrac{\begin{array}{c}\vdots\,\theta\\ \Theta\vdash V\end{array}\quad\begin{array}{c}\vdots\,\gamma\\ \Gamma,U,V,\Gamma'\vdash C\end{array}}{\Gamma,U,\Theta,\Gamma'\vdash C}\,cut<d}{\Gamma,\Delta,\Theta,\Gamma'\vdash C}\,cut<d$
$\backslash$	$\dfrac{\dfrac{\begin{array}{c}\vdots\,\delta\\ U,\Delta\vdash V\end{array}}{\Delta\vdash U\backslash V}\backslash_i\quad\dfrac{\begin{array}{c}\vdots\,\gamma\\ \Gamma,V,\Gamma'\vdash C\end{array}\quad\begin{array}{c}\vdots\,\theta\\ \Theta\vdash U\end{array}}{\Gamma,\Theta,U\backslash V,\Gamma'\vdash C}\backslash_h}{\Gamma,\Theta,\Delta,\Gamma'\vdash C}\,cut\ d$	$\dfrac{\dfrac{\begin{array}{c}\vdots\,\theta\\ \Theta\vdash U\end{array}\quad\begin{array}{c}\vdots\,\delta\\ U,\Delta\vdash V\end{array}}{\Theta,\Delta\vdash V}\,cut<d\quad\begin{array}{c}\vdots\,\gamma\\ \Gamma,V,\Gamma'\vdash C\end{array}}{\Gamma,\Theta,\Delta,\Gamma'\vdash C}\,cut<d$

To be fully complete one should check that whenever the original proof contains no sequent with an empty antecedent, so does the cut free proof we inductively defined.

Now let us summarize what we have proved in this section:

Proposition 21 *Every proof of a given sequent $\Gamma\vdash A$ can be turned into a cut free proof of the same sequent — all formulae in the cut-free proof being sub-formulae of the sequent $\Gamma\vdash C$.*

2.10 Decidability

One way wonder why we wanted to have normal or cut free proof since the computational process of cut elimination or normalization is of little interest for categorial grammars.

What is nevertheless very interesting in such a result is that instead of looking for any proof when we want, for instance to parse and analyze a sentence, we can restrict our search space to these canonical proofs, either normal deductions or cut-free proofs. As we have seen, from cut elimination (or natural deduction normalization) entails the sub-formula property and then it is quite easy to have the decidability of the calculus:

Proposition 22 *There is an algorithm which decides whether a sequent is derivable in L.*

PROOF : Assume we want to prove a sequent. Since the cut rule is not needed, we have finitely many rules to try, each of these rules leading to prove one or two smaller sequents which are also in finite number. ◇

2.11 Models for the Lambek calculus and completeness

We now turn our attention towards models for the Lambek calculus. As we have seen that as far as provability is concerned, cut-free sequent calculus, sequent calculus and natural deduction are equivalent, we are going to use the most adequate formalism to establish properties of models with respect to the deductive system.

These models have been first investigated in [51] and our presentation follows [9].

As we have said Lambek calculus prohibits the empty sequence, and we will present models for L with this restriction. Let us nevertheless say that all these results can be adapted by adding a unit to residuated semi-groups and to semi-groups — replacing the word "semi-group" with the word "monoid".

2.11.1 Residuated semi-groups and the free group model

Let us call a *residuated semi-group*, a structure $(M, \circ, \backslash\backslash, /\!/, \sqsubset)$ where

- M is a set.
- $\circ$ is an associative composition over M — $(M, \circ)$ is a semi-group.
- $\backslash\backslash$ and $/\!/$ are binary composition law on M.
- $\sqsubset$ is an order on M.

which satisfies the following property:

(RSG) The following order relations are either all true or all false:

$$\begin{array}{rcl} a & \sqsubset & (c /\!/ b) \\ (a \circ b) & \sqsubset & c \\ b & \sqsubset & (a \backslash\backslash c) \end{array}$$

Proposition 23 *In a residuated semi-group $(M, \circ, \backslash\backslash, /\!/, \sqsubset)$, for all $a, b, x, y \in M$ one has:*

1. $a \sqsubset b \quad \Rightarrow \quad (a \circ x) \sqsubset (b \circ x)$
2. $a \sqsubset b \quad \Rightarrow \quad (x \circ a) \sqsubset (x \circ b)$

---

[51] Wojciech Buszkowski. Compatibility of a categorial grammar with an asssociated category system. *Zeitschrift für matematische Logik und Grundlagen der Mathematik*, 28:229–238, 1982.

[9] Wojciech Buszkowski. Mathematical linguistics and proof theory. In van Benthem and ter Meulen [1], chapter 12, pages 683–736.

[1] Johan van Benthem and Alice ter Meulen, editors. *Handbook of Logic and Language*. North-Holland Elsevier, Amsterdam, 1997.

3. $\left(\begin{array}{c} a \sqsubset b \\ \textit{and} \\ x \sqsubset y \end{array}\right) \quad \Rightarrow \quad (a \circ x) \sqsubset (b \circ y)$

In other words, a residuated semi-group is in particular an ordered semi-group.

PROOF : (1) From $(b \circ x) \sqsubset (b \circ x)$ ($\sqsubset$ is an order) (RSG) yields $b \sqsubset ((b \circ x) /\!/ x)$; if we assume $a \sqsubset b$ by transitivity of $\sqsubset$ we have $a \sqsubset ((b \circ x) /\!/ x)$ which by (RSG) yields $(a \circ x) \sqsubset (b \circ x)$.

(2) From $(x \circ b) \sqsubset (x \circ b)$ ($\sqsubset$ is an order) (RSG) yields $b \sqsubset (x \backslash\!\backslash (x \circ b))$; if we assume $a \sqsubset b$ by transitivity of $\sqsubset$ we have $a \sqsubset (x \backslash\!\backslash (x \circ b))$ which by (RSG) yields $(x \circ a) \sqsubset (x \circ b)$.

(3) The assumption $a \sqsubset b$ yields $(a \circ x) \sqsubset (b \circ x)$ (*) by (1). The assumption $x \sqsubset y$ yields $(b \circ x) \sqsubset (b \circ y)$ (**) by (2). By transitivity of $\sqsubset$, (*) and (**) yields $(a \circ x) \sqsubset (b \circ y)$.

$\diamond$

Given a residuated semi-group, an *interpretation* $[\ldots]$ is a map from primitive types to elements in M, which extends to types and sequences of types in the obvious way:

$$\begin{array}{rclcrcl} [A,B] & = & [A] \circ [B] & & [A \setminus B] & = & [A] \backslash\!\backslash [B] \\ [A \bullet B] & = & [A] \circ [B] & & [B / A] & = & [B] /\!/ [A] \end{array}$$

A sequent $\Gamma \vdash C$ is said to be *valid* in a residuated semi-group whenever $[\Gamma] \sqsubset [C]$.

2.11.2 The free group model

A particular case of residuated semi-group is the free group over primitive types. It will be especially important in the section 2.13. The free group interpretation for L is

- a particular residuated semi-group where
  - $(M, \cdot)$ is the free group over the propositional variables,
  - $a \backslash\!\backslash b$ is $a^{-1}b$
  - $b /\!/ a$ is ba^{-1}
  - $a \sqsubset b$ is $a = b$ (the discrete order)

  One easily observes that the three equalities

  $$ab = c \qquad a = cb^{-1} \qquad b = a^{-1}c$$

  are equivalent — so (RSG) holds.
- a standard interpretation defined by $[p] = p$

Because of the soundness of L w.r.t. residuated semi-groups (next proposition) whenever a sequent $\Gamma \vdash C$ is provable one has $[\Gamma] = [C]$ in the free group. The free group model is of course not complete: indeed it interprets $\vdash$ by a symmetrical relation ($=$) while $\vdash$ is not symmetrical: $n \vdash s \,/\, (n \setminus s)$ is provable but not $s \,/\, (n \setminus s) \vdash n$.

2.11.3 L is sound and complete w.r.t. residuated semi-groups

Proposition 24 *A provable sequent is valid in every residuated semi-group, for every interpretation of the primitive types.*

PROOF : We proceed by induction on the proof in natural deduction.

If the proof consists in an axiom $X \vdash X$ then the result is true: $[X] \sqsubset [X]$ whatever the semi-group or the interpretation is.

If the last rule is the introduction rule $\setminus_i$:

$$\frac{A, \Gamma \vdash C}{\Gamma \vdash A \setminus C} \setminus_i \quad \Gamma \neq \varepsilon$$

by induction hypothesis we have $[A] \circ [\Gamma] \sqsubset [C]$, thus, by (RSG) we have $[\Gamma] \sqsubset ([A] \setminus\setminus [C])$, so the sequent $\Gamma \vdash A \setminus C$ is valid as well.

If the last rule is the introduction rule $/_i$ we proceed as for $\setminus_i$.

If the last rule is the elimination rule $\setminus_e$:

$$\frac{\Gamma \vdash A \quad \Delta \vdash A \setminus B}{\Gamma, \Delta \vdash B} \setminus_e$$

then by induction hypothesis we know that $[\Gamma] \sqsubset [A]$, and using proposition 23 we have $[\Gamma] \circ [\Delta] \quad \sqsubset \quad [A] \circ [\Delta]$ (1); we also have $[\Delta] \sqsubset [A] \setminus\setminus [B]$ — hence by (RSG) $([A] \circ [\Delta]) \sqsubset [B]$ (2). Therefore from (1) and (2) we obtain,

$$[\Gamma, \Delta] \quad = \quad [\Gamma] \circ [\Delta] \quad \sqsubset (1) \quad [A] \circ [\Delta] \quad \sqsubset (2) \quad [B]$$

If the last rule is the elimination rule $/_e$ we proceed as for $/_i$.

If the last rule is the product elimination rule $\bullet_e$

$$\frac{\Gamma \vdash A \bullet B \quad \Delta, A, B, \Delta' \vdash C}{\Delta, \Gamma, \Delta' \vdash C} \bullet_e$$

By induction hypothesis we know that $[\Gamma] \sqsubset [A \bullet B] = [A] \circ [B]$, and, using proposition 23 we obtain $[\Delta] \circ [\Gamma] \circ [\Delta'] \sqsubset [\Delta] \circ [A] \circ [B] \circ [\Delta']$. We also know that $[\Delta, A, B, \Delta'] = [\Delta] \circ [A] \circ [B] \circ [\Delta'] \sqsubset [C]$. We therefore have

$$[\Delta, \Gamma, \Delta'] = [\Delta] \circ [\Gamma] \circ [\Delta'] \sqsubset [\Delta] \circ [A] \circ [B] \circ [\Delta'] \sqsubset [C]$$

If the last rule is the product introduction rule $\bullet_i$ by induction hypothesis we know that $[\Delta] \sqsubset [A]$ and that $[\Delta'] \sqsubset [B]$; consequently

$$[\Delta, \Delta'] = [\Delta] \circ [\Delta'] \sqsubset [A] \circ [B] = [A \bullet B]$$

$\diamond$

Proposition 25 *A sequent which is valid in every residuated semi-group is derivable.*

PROOF : Let F be the set of formulae and let $M = F/\!\vdash\!\dashv$ be the quotient of formulae by the equivalence relation $\vdash\!\dashv$; this relation $\vdash\!\dashv$ is defined by $A \vdash\!\dashv B$ whenever $A \vdash B$ and $B \vdash A$; it is symmetrical, axioms shows it is reflexive and the cut rule makes sure it is transitive.

Is is easily observed that $\backslash$, $/$, $\bullet$ and $\vdash$ can be defined over equivalence classes, that is: whenever $A \vdash\!\dashv A'$ and $B \vdash\!\dashv B'$ one has $(A \diamond B) \vdash\!\dashv (A' \diamond B')$ So let us define $\circ, \backslash\backslash, /\!/$ as the similar operations over equivalence classes of $\vdash\!\dashv$: $A^{\vdash\!\dashv} \circ B^{\vdash\!\dashv} = (A \bullet B)^{\vdash\!\dashv}$, $A^{\vdash\!\dashv} \backslash\backslash B^{\vdash\!\dashv} = (A \backslash B)^{\vdash\!\dashv}$ and $B^{\vdash\!\dashv} /\!/ A^{\vdash\!\dashv} = (B/A)^{\vdash\!\dashv}$ Finally let $\sqsubset$ be $\vdash$ which can also be defined for equivalence classes: if $A \vdash\!\dashv A'$ and $B \vdash\!\dashv B'$ then $A \vdash B$ is equivalent to $A' \vdash B'$.

The property (RSG) is satisfied i.e. $(A^{\vdash\!\dashv} \circ B^{\vdash\!\dashv}) \sqsubset C^{\vdash\!\dashv}$ is equivalent to $A^{\vdash\!\dashv} \sqsubset (C^{\vdash\!\dashv} /\!/ B^{\vdash\!\dashv})$ and to $A^{\vdash\!\dashv} \sqsubset (B^{\vdash\!\dashv} \backslash\backslash C^{\vdash\!\dashv})$. Indeed $A \vdash A$ and $B \vdash B$ lead to $A, B \vdash A \bullet B$; thus from $A \bullet B \vdash C$ one obtains $A, B \vdash C$ which yields $A \vdash C/B$ by $/_i$ and $B \vdash A \backslash C$ by $\backslash_i$; from $B \vdash A \backslash C$ (resp. $A \vdash B/C$) using $A \vdash A$ (resp. $B \vdash B$) one obtains $A, B \vdash C$ by $\backslash_e$ (resp. by $/_e$) and $A \bullet B \backslash C$ by $\bullet_h$.

Now let us consider the interpretation $[p] = p^{\vdash\!\dashv}$ for every primitive type. Then for every formula $[A] = A^{\vdash\!\dashv}$.

To say that a sequent $H_1, \ldots, H_n \vdash A$ is valid in this model under this interpretation is to say that $[H_1, \ldots, H_n] \sqsubset [A]$. Therefore $H_1 \bullet \cdots \bullet H_n \vdash A$ is provable which entails that $H_1, \ldots, H_n \vdash A$ is provable as well — indeed from $H_1 \bullet \cdots \bullet H_n \vdash A$ one obtains $\vdash H_1 \bullet \cdots \bullet H_n \backslash A$ (*); then by n rules $\bullet_i$ on the axioms $H_i \vdash H_i$ one obtains $H_1, \ldots, H_n \vdash H_1 \bullet \cdots \bullet H_n$ (**) and an application of $\bullet_e$ to (*) and (**) yields $H_1, \ldots, H_n \vdash A$. $\diamond$

2.11.4 L is sound and complete w.r.t. (free) semi-group models

A more interesting class of models is provided by semi-groups. Indeed, the interpretation of a category should be the set of the words and expressions of this category, shouldn't it?

So, given a semi-group $(W, .)$ that is a set W endowed with an associative composition "." one can define a residuated semi-group as follows:

- $M = 2^W$

- $A \circ B = \{ab \quad | \quad a \in A \text{ and } b \in B\}$
- $A \backslash\backslash B = \{z \quad | \quad \forall a \in A \quad az \in B\}$
- $B // A = \{z \quad | \quad \forall a \in A \quad za \in B\}$
- $A \sqsubset B$ whenever $A \subset B$ (as sets).

It is easily seen that this structure really is a residuated semi-group:

- $\circ$ is associative:

$$(A \circ B) \circ C = \{abc \quad | \quad a \in A \text{ and } b \in B \text{ and } c \in C\} = A \circ (B \circ C)$$

- $\subset$ is an order on 2^W

(RSG) The following statements are clearly equivalent:

$$\begin{array}{lcl} (A \circ B) \subset C & : & \forall a \in A \, \forall b \in B \quad ab \in C \\ A \subset (C // B) & : & \forall a \in A \quad a \in (C // B) \\ B \subset (A \backslash\backslash C) & : & \forall b \in B \quad b \in (A \backslash\backslash C) \end{array}$$

What is of special interest are free semi-group models, since there are no equations between sequences of words. The following result may be understood as L is the logic of free semi-groups:

Proposition 26 *Product free L is complete over free semi-group models.*

PROOF : Take as semi-group the finite non empty sequences of formulae F^+, endowed with concatenation $(A_1, \ldots, A_n) \cdot (B_1, \ldots, B_p) = A_1, \ldots, A_n, B_1, \ldots, B_p$.

For a primitive type p define $[p]$ by $\{\Gamma \quad | \quad \Gamma \vdash p\}$.

Let us firstly see that for every formula F, the set of finite sequences $[F]$ defined inductively from the $[p]$'s by the definition of $\backslash\backslash$ and $//$ is precisely $Ctx(F) = \{\Delta \quad | \quad \Delta \vdash F\}$. We proceed by induction on F. Is F if some primitive type, it is the definition. Now assume that $[G] = Ctx(G)$ and $[H] = Ctx(H)$ and let us see that $[G \backslash H] = Ctx(G \backslash H)$ — the case H / G being symmetrical.

$Ctx(G \backslash H) \subset [G \backslash H]$ Let Δ be a sequence such that $\Delta \in Ctx(G \backslash H)$ that is $\Delta \vdash G \backslash H$ (1) and let us see that for every $\Theta \in [G]$ we have $\Theta, \Delta \in [H]$ — which entails $\Delta \in [G \backslash H]$. By induction hypothesis we have $Ctx(G) = [G]$ so $\Theta \vdash G$ (2). From (1) and (2) we obtain $\Theta, \Delta \vdash H$, so $\Theta, \Delta \in Ctx(H)$. Since by induction hypothesis $Ctx(H) = [H]$ we have $\Theta, \Delta \in [H]$. As this holds for every Θ we have $\Delta \in [G \backslash H]$.

$[G \setminus H] \subset Ctx(G \setminus H)$ Let Δ be a sequence such that $\Delta \in [G \setminus H]$. Let us show that $\Delta \vdash G \setminus H$. Since $G \vdash G$ we have $G \in Ctx(G)$ and by induction hypothesis $G \in [G]$. By definition of $[G \setminus H]$ we thus have $G, \Delta \in [H]$ and, since by induction hypothesis we have $[H] = Ctx(H)$ we obtain $G, \Delta \vdash H$. Now, by the $\setminus_i$ introduction rule we obtain $\Delta \vdash G \setminus H$, that is $\Delta \in Ctx(G \setminus H)$.

If a sequent$A_1, \ldots, A_n \vdash C$ is valid in this model under this interpretation, what does it means? We have $[A_1] \circ \cdots \circ [A_n] \subset [C]$ and as $A_i \in [A_i]$ we have $A_1, \ldots, A_n \in [C]$ that is $A_1, \ldots, A_n \vdash C$. $\diamond$

Next follows a very difficult result due to Pentus [52], that we give without proof:

Proposition 27 *L with product is also complete w.r.t. free semi-groups models.*

2.12 Interpolation

This section presents the interpolation theorem for Lambek calculus which appeared in the thesis of Roorda [53].

Interpolation is somehow the converse of cut elimination. The interest of cut free proofs is that they obey the sub-formula property. The usual interest of interpolation, say for classical or intuitionistic logic is to be able to factor equal sub-proofs in a given proof. In the Lambek calculus where contraction is prohibited, nothing like this can happen. So the interest is very different, let us explain it shortly.

Assume we are able to formulate the calculus with a set of axioms, and only the cut rule: viewing $\vdash$ as $\longrightarrow$ (in th oposite direction) the calculus is nothing but a set of context free production rules — the cut rule is the substitution rule often left implicit in phrase structure grammars.

Indeed a production rule $X \longrightarrow X_1 \cdots X_n$ corresponds to an axiom $X_1, \ldots, X_n \vdash X$ and the cut rule simply state that is we have been able to derive

$$W \longrightarrow V_1 \cdots V_k \, T \, U_1 \cdots U_l$$
$$T \longrightarrow Z_1 \cdots Z_j$$

then we are able to derive

$$W \longrightarrow V_1 \cdots V_k \, Z_1 \cdots Z_j \, U_1 \cdots U_l.$$

Now observe that for a given Lambek grammar because of cut elimination we know that the types appearing in any syntactic analysis are all sub-formulae of the conclusion sequent: indeed a syntactic analysis is a proof of $t_1, \ldots, t_n \vdash S$ with all t_i in the lexicon.

[52] Mati Pentus. Lambek calculus is L-complete. Technical Report LP-93-14, Institute for Logic, Language and Computation, Universiteit van Amsterdam, 1993.

[53] Dirk Roorda. *Resource logic: proof theoretical investigations*. PhD thesis, FWI, Universiteit van Amsterdam, 1991.

Can we derive every any syntactic analysis from a *finite* number of provable sequents by means of the cut rule only?

As we shall see in the next section, it is possible and consequently Lambek grammars are weakly equivalent to context free grammars.

Given a formula or a sequence of formulae Δ and a primitive type p we denote by $\rho_p(\Delta)$ *the number of occurrences of* p *in* Δ.

Proposition 28 *Let* $\Gamma,\Delta,\Theta \vdash C$ *be a provable sequent in L, with* $\Delta \neq \varepsilon$. *There exists an interpolant of* Δ *that is a formula* I *such that:*

1. $\Delta \vdash I$
2. $\Gamma, I, \Theta \vdash C$
3. $\rho_p(I) \leq \rho_p(\Delta)$ *for every primitive type* p
4. $\rho_p(I) \leq \rho_p(\Gamma,\Theta,C)$ *for every primitive type* p

PROOF : We proceed by induction on the size of a cut free proof of $\Gamma,\Delta,\Theta \vdash C$ — there are many cases in this proof, according to the nature of the last rule, and to the respective position of the created formula and Δ.

$$\boxed{\text{axiom } X \vdash X}$$

If the proof is an axiom, then Δ is a formula X and $I = X$ obviously works:

1. $X \vdash X$
2. $X \vdash X$
3. $\rho_p(X) = \rho_p(X)$
4. $\rho_p(X) = \rho_p(\varepsilon,\varepsilon,X)$

$$\boxed{\frac{\Pi \vdash X \quad \Phi \vdash Y}{\Pi,\Phi \vdash X \bullet Y}\ \bullet_h}$$

- $\Pi = \Pi',\Delta,\Pi''$ — so $\Gamma = \Pi'$ and $\Theta = \Pi'',\Phi$.
  By induction hypothesis we have an interpolant I for Δ in $\Pi',\Delta,\Pi'' \vdash X$, let us see it is an interpolant for Δ in $\Pi',\Delta,\Pi'',\Phi \vdash X \bullet Y$.
  1. We already have $\Delta \vdash I$
  2. From $\Pi',I,\Pi'' \vdash X$ and $\Phi \vdash Y$, we have $\Pi',I,\Pi'',\Phi \vdash X \bullet Y$.
  3. We already have $\rho_p(I) \leq \rho_p(\Delta)$.
  4. From $\rho_p(I) \leq \rho_p(\Pi',\Pi'',X)$ we obtain $\rho_p(I) \leq \rho_p(\Pi',\Pi'',\Phi,X,Y)$.

- $\Phi = \Phi', \Delta, \Phi''$ — so $\Gamma = \Pi, \Phi'$ and $\Theta = \Phi''$.
  Symmetrical to the previous case.
- $\Pi = \Pi', \Delta'$ $\Phi = \Delta'', \Phi''$ and $\Delta = \Delta', \Delta''$ — so $\Gamma = \Pi'$ and $\Theta = \Phi''$.
  By induction hypothesis we have an interpolant I' for Δ' in $\Pi', \Delta' \vdash X$ and an interpolant I'' for Δ'' in $\Delta'', \Phi'' \vdash X$. Then $I = I' \bullet I''$ is an interpolant for $\Delta = \Delta', \Delta''$ in $\Pi', \Delta', \Delta'', \Phi'' \vdash X \bullet Y$.
  1. From $\Delta' \vdash I'$ and $\Delta'' \vdash I''$ we obtain $\Delta', \Delta'' \vdash X \bullet Y$ by $\bullet_i$.
  2. From $\Pi', I' \vdash X$ and $I'', \Phi'' \vdash Y$ we have $\Pi', I', I'', \Phi'' \vdash X \bullet Y$ by $\bullet_i$ and finally $\Pi', I' \bullet I'', \Phi'' \vdash X \bullet Y$ by $\bullet_h$
  3. From $\rho_p(I') \leq \rho_p(\Pi', X)$ and $\rho_p(I'') \leq \rho_p(\Phi'', Y)$ we get $\rho_p(I' \bullet I'') = \rho_p(I') + \rho_p(I'') \leq \rho_p(\Pi', X) + \rho_p(\Phi'', Y) = \rho_p(\Pi', \Phi'', X, Y) = \rho_p(\Pi', \Phi'', X \bullet Y)$.
  4. From $\rho_p(I') \leq \rho_p(\Delta')$ and $\rho_p(I'') \leq \rho_p(\Delta'')$ we get $\rho_p(I' \bullet I'') = \rho_p(I') + \rho_p(I'') \leq \rho_p(\Delta', \Delta'') = \rho_p(\Delta)$.

$$\boxed{\frac{\Pi, X, Y, \Phi \vdash C}{\Pi, X \bullet Y, \Phi \vdash C} \bullet_h}$$

Let Δ' be defined as follows: if Δ contains $X \bullet Y$ then $\Delta' = \Delta[X \bullet Y := X, Y]$, otherwise $\Delta' = \Delta$. Let I be an interpolant for Δ' in $\Pi, X, Y, \Phi \vdash C$. Then I is itself an interpolant for Δ in $\Pi, X \bullet Y, \Phi \vdash C$.

1. From $\Delta' \vdash I$ we have $\Delta \vdash I$ (possibly using $\bullet_h$).
2. From $\Pi, X \bullet Y, \Phi[\Delta' := I] \vdash C$ we get $\Pi, X \bullet Y, \Phi[\Delta := I] \vdash C$.
3. From $\rho_p(\Delta) = \rho_p(\Delta')$ we obtain $\rho_p(I) \leq \rho_p(\Delta)$
4. Since $\rho_p((\Pi, X \bullet Y, \Phi)[\Delta' := \varepsilon], C) = \rho_p((\Pi, X \bullet Y, \Phi)[\Delta := \varepsilon], C)$ we have $\rho_p(I) \leq \rho_p((\Pi, X \bullet Y, \Phi)[\Delta := \varepsilon], C)$.

$$\boxed{\frac{X, \Gamma, \Delta, \Theta \vdash Y}{\Gamma, \Delta, \Theta \vdash X \setminus Y} \setminus_i}$$

By induction hypothesis we have an interpolant I for Δ in $A, \Gamma, \Delta, \Theta \vdash B$. It is an interpolant for Δ in $\Gamma, \Delta, \Theta \vdash X \setminus Y$ as well.

1. We already have $\Delta \vdash I$.
2. From $X, \Gamma, I, \Theta \vdash Y$ we obtain $\Gamma, I, \Theta \vdash X \setminus Y$ by $\setminus_i$.
3. We already have $\rho_p(I) \leq \rho_p(\Delta)$.
4. We have: $\rho_p(I) \leq \rho_p(X, \Gamma, \Theta, Y) = \rho_p(\Gamma, \Theta, X \setminus Y)$.

$$\boxed{\dfrac{\Pi \vdash X \quad \Phi, Y, \Psi \vdash C}{\Phi, \Pi, X \setminus Y, \Psi \vdash C} \setminus_h}$$

- Δ is included into Π Let I be an interpolant for Δ in the premise containing it. Then I is an interpolant for Δ in $\Phi, \Pi, X \setminus Y, \Psi \vdash C$.
  1. We already have $\Delta \vdash I$
  2. From $\Pi[\Delta := I] \vdash X$ and $\Phi, Y, \Psi \vdash C$, by $\setminus_h$ we obtain $\Phi, \Pi[\Delta := I], X \setminus Y, \Psi \vdash C$
  3. We already have $\rho_p(I) \leq \rho_p(\Delta)$.
  4. From $\rho_p(I) \leq \rho_p(\Pi[\Delta := \varepsilon], X)$ we have $\rho_p(I) \leq \rho_p(\Phi, \Pi[\Delta := \varepsilon], X \setminus Y, \Psi, C)$
- Δ is included in Φ (resp. Ψ) Let I be an interpolant for Δ in the premise containing it. Then I is an interpolant for Δ in $\Phi, \Pi, X \setminus Y, \Psi \vdash C$.
  1. We already have $\Delta \vdash I$
  2. From $\Phi[\Delta := I], Y, \Psi \vdash C$ (resp. $\Phi, Y, \Psi[\Delta := I] \vdash C$) and $\Pi \vdash X$, by $\setminus_h$ we obtain $\Phi[\Delta := I], \Pi, X \setminus Y, \Psi \vdash C$ (resp. $\Phi, \Pi, X \setminus Y, \Psi[\Delta := I] \vdash C$)
  3. We already have $\rho_p(I) \leq \rho_p(\Delta)$.
  4. From $\rho_p(I) \leq \rho_p(\Phi[\Delta := \varepsilon], Y, \Psi, C)$ (resp. $\rho_p(I) \leq \rho_p(\Phi, Y, \Psi[\Delta := \varepsilon], C)$) we have $\rho_p(I) \leq \rho_p(\Phi[\Delta := \varepsilon], \Pi, X \setminus Y, \Psi, C)$ (resp. $\rho_p(I) \leq \rho_p(\Phi, \Pi, X \setminus Y, \Psi[\Delta := \varepsilon], C)$).
- $\Delta = \Delta', \Delta''$ and $\Phi = \Phi', \Delta'$ and $\Pi = \Delta'', \Pi''$.

  Let I' be an interpolant for Δ' in $\Phi', \Delta', Y, \Psi \vdash C$, and let I'' be an interpolant for Δ'' in $\Delta'', \Pi'' \vdash X$. Then $I = I' \bullet I''$ is an interpolant for Δ', Δ'' in $\Phi', \Delta', \Delta'', \Pi'', X \setminus Y, \Psi \vdash C$.
  1. From $\Delta' \vdash I'$ and $\Delta'' \vdash I''$ we have $\Delta', \Delta'' \vdash I' \bullet I''$ by $\bullet_i$.
  2. From $I'', \Pi'' \vdash X$ and $\Phi', I', Y, \Psi \vdash C$ we have $\Phi', I', I'', X \setminus Y, \Psi \vdash C$ by $\setminus_h$ and $\Phi', I' \bullet I'', X \setminus Y, \Psi \vdash C$ by $\bullet_i$.
  3. We have $\rho_p(I' \bullet I'') = \rho_p(I') + \rho_p(I'') \leq \rho_p(\Delta') + \rho_p(\Delta'') = \rho_p(\Delta)$.
  4. We have $\rho_p(I' \bullet I'') = \rho_p(I') + \rho_p(I'') \leq \rho_p(\Phi', Y, \Psi, C) + \rho_p(\Pi'', X) = \rho_p(\Phi', \Pi'', X \setminus Y, \Psi, C)$.
- $\Delta = \Phi'', \Pi, X \setminus Y, \Psi'$ with $\Phi = \Phi', \Phi''$ and $\Psi = \Psi', \Psi''$.

  Let I be an interpolant for Φ'', Y, Ψ' in $\Phi', \Phi'', Y, \Psi', \Psi'' \vdash C$. Then I is itself interpolant for $\Phi'', \Pi, X \setminus Y, \Psi'$ in $\Phi', \Phi'', \Pi, X \setminus Y, \Psi', \Phi'' \vdash C$.
  1. From $\Phi'', Y, \Psi' \vdash I$ and $\Pi \vdash X$ we have $\Phi'', \Pi, X \setminus Y, \Psi' \vdash I$ by $\setminus_h$.
  2. We already have $\Phi', I, \Psi'' \vdash C$.
  3. We already have $\rho_p(I) \leq \rho_p(\Phi', \Psi'', C)$
  4. We have $\rho_p(I) \leq \rho_p(\Phi'', Y, \Psi') \leq \rho_p(\Phi'', \Pi, X \setminus Y, \Psi')$.

- $\Delta = \Pi'', X \setminus Y, \Psi'$ with $\Pi = \Pi', \Pi''$ and $\Psi = \Psi', \Psi''$.
  Let I' be and interpolant for Π' in $\Pi', \Pi'' \vdash X$ and let I'' be an interpolant for Y, Ψ' in $\Phi, Y, \Psi', \Psi'' \vdash C$. Then $I' \setminus I''$ is an interpolant for $\Delta = \Pi'', X \setminus Y, \Psi'$ in $\Phi, \Pi', \Pi'', X \setminus Y, \Psi', \Psi'' \vdash C$.
  1. From $I', \Pi'' \vdash X$ and $Y, \Psi' \vdash I''$ we have $I', \Pi'', X \setminus Y, \Psi' \vdash I''$ by $\setminus_h$ and $\Pi'', X \setminus Y, \Psi' \vdash I' \setminus I''$ by $\setminus_i$.
  2. From $\Phi, I'', \Psi'' \vdash C$ and $\Pi' \vdash I'$ we have $\Phi, \Pi', I' \setminus I'', \Psi'' \vdash C$.
  3. We have $\rho_p(I' \setminus I'') \leq \rho_p(\Pi'', X) + \rho_p(Y, \Psi') = \rho_p(\Pi'', X \setminus Y, \Psi')$
  4. We have $\rho_p(I' \setminus I'') \leq \rho_p(\Pi') + \rho_p(\Phi, \Psi'', C) = \rho_p(\Phi, \Pi', \Psi'', C)$

This ends the proof because $/_i$ and $/_e$ are symmetrical to $\setminus_i$ and $\setminus_e$. ◇

2.13 Lambek grammars and context-free grammars

At the beginning of this section we shall see that context free grammars translate into weakly equivalent Lambek grammars [50]: this is non trivial but unsurprising, and this section is in fact devoted to prove the converse, known as Chomsky conjecture stated in 1963 [28, p. 413] and proved by Pentus in 1992 [54]: *Languages generated by Lambek grammars are context free languages*.

This result was already suggested in the previous section on interpolation: if we are able to derive all sequents corresponding to syntactic analyses from a *finite* set of sequents by the cut rule only, then Lambek grammars are context free.

Let us define the *size* $|A|$ of a formula A by its number of primitive types. We are going to show that given an integer m there exists a *finite* set $AX(m)$ of provable sequents such that all provable sequent containing only formulae of size smaller than m are derivable from sequents in $AX(m)$ by means of the cut rule only. This easily entails that Lambek grammars are context-free.

This does not means that they should be left out: they are lexicalized, they offer a pleasant interface with semantics, and even for syntactic considerations, let us say that while the derivation trees of a context-free grammars constitute a regular tree language [55,56] the derivation trees (natural deduction trees) of a Lambek grammar

[50] Joel M. Cohen. The equivalence of two concepts of categorial grammars. *Information and Control*, 1967.

[28] Noam Chomsky. Formal properties of grammars. In *Handbook of Mathematical Psychology*, volume 2, pages 323 – 418. Wiley, New-York, 1963.

[54] Mati Pentus. Lambek grammars are context-free. In *Logic in Computer Science*. IEEE Computer Society Press, 1993.

[55] J. W. Thatcher. Characterizing derivation trees of context free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences*, 1:317–322, 1967.

[56] Ferenc Gécseg and Magnus Steinby. Tree languages. In Rozenberg and Salomaa [4], chapter 1.

[4] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*. Springer Verlag, Berlin, 1997.

can constitute a non regular tree language — but always consitute a context-free tree language. [45]

There are basically two ingredients for the Pentus proof that Lambek grammars are context free. One is interpolation and we already explained its relevance to this question. The other is a property of the free group to be applied to the free group model of section 2.11.2 page 43. This property is needed to find, in a sequent where all formulae have sizes lower than m, two (or more) consecutive formulae whose interpolant also has a size less than m — this is of course to be used for the final induction.

We mainly follow [54], and borrow a few things from [57,9].

2.13.1 From context-free grammars to Lambek grammars

It is natural to think that every AB grammar corresponds to a Lambek grammar because Lambek calculus includes the residuation laws and is even richer. In fact this result although not as difficult as the one needed in the previous section is not fully straight forward.

Using proposition 1 from chapter 1, any AB grammar is weakly equivalent to an AB grammar only containing types of order at most 1.

Now, by proposition 19 a sequent $A_1, \ldots, A_n \vdash S$ with $o(A_i) \leq 1$ is provable with AB residuation rules if and only if it is provable in L. Consequently the language generated by an AB grammar with types of order at most 1 coincide with the language generated by the Lambek grammar with the same lexicon.

Using the weak equivalence between AB grammars and context-free grammars (propositions 4 and 3) we have the result of [50]:

Proposition 29 *Every ε-free context-free grammar is weakly equivalent to a Lambek grammar.*

---

[45] Hans-Jörg Tiede. Lambek calculus proofs and tree automata. In Michael Moortgat, editor, *Logical Aspects of Computational Linguistics, LACL'98, selected papers*, number 2014 in LNCS/LNAI. Springer-Verlag, 2001.

[54] Mati Pentus. Lambek grammars are context-free. In *Logic in Computer Science*. IEEE Computer Society Press, 1993.

[57] Mati Pentus. Product-free lambek calculus and context-free grammars. *Journal of Symbolic Logic*, 62(2):648–660, 1997.

[9] Wojciech Buszkowski. Mathematical linguistics and proof theory. In van Benthem and ter Meulen [1], chapter 12, pages 683–736.

[1] Johan van Benthem and Alice ter Meulen, editors. *Handbook of Logic and Language*. North-Holland Elsevier, Amsterdam, 1997.

[50] Joel M. Cohen. The equivalence of two concepts of categorial grammars. *Information and Control*, 1967.

2.13.2 A property of the free group

Let w be an element of the free group; then $\|w\|$ stands for the length of the reduced word corresponding w — e.g. $\|cb^{-1}a^{-1}abc\| = 2$.

This lemma which is needed for a refinement of interpolation, only concern the free-group. It has actually been proved earlier on in [58] and reproved in [59].

Proposition 30 *1. Let u,v,w be elements of the free group; if $\|u\| < \|uv\|$ and $\|uv\| \geq \|uvw\|$ then $\|vw\| \leq \max(\|v\|,\|w\|)$.*

2. Let u_i $i = 1,\ldots,n+1$ be elements of the free group with $u_1 \cdot \cdots \cdot u_{n+1} = 1$. Then there exists $k \leq n$ such that

$$\|u_k u_{k+1}\| \leq \max(\|u_k\|,\|u_{k+1}\|)$$

PROOF : The first part is actually a lemma for the second part.

1 We proceed by reductio ad absurdum, so we assume that

a. $\|u\| < \|uv\|$
b. $\|uv\| \geq \|uvw\|$
c. $\|vw\| > \|v\|$
d. $\|vw\| > \|w\|$

There exists three reduced words x,y,z such that

- $u = xy^{-1}$ $v = yz$ $uv = xz$
- xy^{-1} yz xz are reduced.

From (a) we have $\|x\| + \|y\| < \|x\| + \|z\|$ so $\|y\| < \|z\|$ and therefore $\|y\| < \frac{1}{2}\|v\|$ (*).

Similarly there exists three reduced words x',y',z' such that

- $v = x'y'$ $w = y'^{-1}z'$ $vw = x'z'$
- $x'y'$ $y'^{-1}z'$ $x'z'$ are reduced.

From (c) we have $\|y'\| + \|z'\| < \|x'\| + \|z'\|$ so $\|y'\| < \|x'\|$ and therefore $\|y'\| < \frac{1}{2}\|v\|$ (**)

From $v = yz = x'y'$ with $\|y\| < \frac{1}{2}\|v\|$ (*) and $\|y'\| < \frac{1}{2}\|v\|$ (**), there exists a non empty a such that

- $z = ay'$ $x' = ya$ $v = yay'$
- ay' ya yay' are reduced

---

[58] Maurice Nivat. Congruences de thue et t-langages. *Studia scientiarum mathematicarum hungarica*, 6:243–249, 1971.

[59] Jean-Michel Autebert, Luc Boasson, and Géraud Sénizergues. Langages de parenthèses, langages n.t.s. et homomorphismes inverses. *R.A.I.R.O. Informatique Théorique*, 18(4):327–344, 1984.

So we have $uvw = xy^{-1}yay'y'^{-1}z' = xaz'$ — as xa and az' are reduced, xaz' is reduced as well. From (b) we have

$$\|uvw\| = \|xaz'\| \leq \|xay'\| = \|xz\| = \|uv\|$$

and therefore $\|z'\| \leq \|y'\|$.
Since from d we have $\|x'z'\| > \|x'y'\|$ so $\|z'\| > \|y'\|$, there is a contradiction.

2 Let k be the first index such that $\|u_1 \cdots u_k\| \geq \|u_1 \cdots u_k u_{k+1}\|$.
If $k = 1$ $\|u_1\| \geq \|u_1 u_2\|$ then $\max(\|u_1\|, \|u_2\|) \geq \|u_1\| \geq \|u_1 u_2\|$.
Otherwise, let

$$u = u_1 \cdots u_{k-1} \qquad v = u_k \qquad w = u_{k+1}$$

we have

$$\|u\| = \|u_1 \cdots u_{k-1}\| < \|uv\| = \|u_1 \cdots u_{k-1} u_k\|$$

and

$$\|uv\| = \|u_1 \cdots u_{k-1} u_k\| \geq \|uvw\| = \|u_1 \cdots u_k u_{k+1}\|$$

so applying the first part (1) of this proposition we obtain

$$\|u_k u_{k+1}\| \leq \max(\|u_k\|, \|u_{k+1}\|)$$

◇

2.13.3 Interpolation for thin sequents

A sequent $\Gamma \vdash C$ is said to be *thin* whenever it is provable and $\rho_p(\Gamma, C)$ is at most 2 — where $\rho_p(\Theta)$ is the number of occurrences of a primitive type p in Θ. Notice that by proposition 15 which says that a provable sequent contains as many positive and negative occurrences of a primitive type, $\rho_p(\Gamma, A)$ is either 0 or 2.

Here is a proposition which is very representative of multiplicative calculi, in which a formula is neither contracted or weakened:

Proposition 31 *Each provable sequent may be obtained from a thin sequent by substituting primitive types with primitive types.*

PROOF : Given a cut free proof d with only primitive axioms of a sequent $\Gamma \vdash C$, number the axioms and replace each axiom $p \vdash p$ by $p_i \vdash p_i$ where i is the number of the axiom, and also replace all the traces of this occurrence of p in the proof with p_i. Clearly the result is itself a proof of a sequent $\Gamma' \vdash C'$, which contains exactly two or zero occurrences of each primitive type, and which gives back $\Gamma \vdash C$ when each p_i is substituted with p. ◇

Proposition 32 *Let $\Gamma,\Delta,\Theta \vdash C$ be a thin sequent. Then there exists a formula B such that:*

1. *$\Delta \vdash B$ is thin*
2. *$\Gamma,B,\Theta \vdash C$ is thin*
3. *$|B| = \|[\Delta]\|$ — the number of primitive types in B is the size of the interpretation of Δ in the free group (see pragraph 2.11.2 page 43).*

PROOF : p stands for any primitive type,

Let B be an interpolant of Δ which exists by theorem 28. We then have:

a. $\Delta \vdash B$ is provable

b. $\Gamma,B,\Theta \vdash C$ is provable

c. $\rho_p(B) \leq \min(\rho_p(\Gamma,\Theta,C),\rho_p(\Delta))$

Let us prove 1. As the sequent $\Gamma,\Delta,\Theta \vdash C$ is thin,

$$\rho_p(\Gamma,\Delta,\Theta,C) = \rho_p(\Gamma,\Theta,C) + \rho_p(\Delta)$$

is either 0 or 2; so by c $\rho_p(B)$ is either 0 or 1, and we have

$$\rho_p(\Delta,B) = \rho_p(\Delta) + \rho_p(B) \leq \rho_p(\Gamma,\Delta,\Theta,C) + \rho_p(B) \leq 2+1$$

Since $\Delta \vdash B$ is provable (a), $\rho_p(\Delta,B)$ is even, and thus $\rho_p(\Delta,B) \leq 2$. So, being provable, $\Delta \vdash B$ is thin.

Now let us prove 2 Similarly,

$$\rho_p(\Gamma,B,\Theta,C) = \rho_p(\Gamma,\Theta,C) + \rho_p(B) \leq \rho_p(\Gamma,\Delta,\Theta,C) + \rho_p(B) \leq 2+1$$

Since $\Gamma,B,\Theta \vdash C$ is provable (b)

$$\rho_p(\Gamma,B,\Theta,C)$$

is even, so

$$\rho_p(\Gamma,B,\Theta,C) \leq 2$$

So, being provable, $\Gamma,B,\Theta \vdash C$ is thin.

Finally let us show 3.

- if p does not occur in Δ then p does neither occur in $[\Delta]$ nor in B, by c.
- if p occurs once in Δ then it occurs once in $[\Delta]$ too — it cannot cancel with another occurrence of p; as $\Delta \vdash B$ is thin it also occurs once in B — it occurs twice in $\Delta \vdash B$ and once in Δ so it occurs once in B.

- if p occurs twice in Δ then it does not occur in Γ,Θ,C; therefore it does not occur in B by (c). The soundness of the interpretation in the free group entails $[\Gamma,\Delta,\Theta] = [C]$ that is

$$[\Delta] = [\Gamma]^{-1}[C][\Theta]^{-1}$$

As p does not occur in Γ,Θ,C, there is no occurrence of p in $[\Gamma]^{-1}[C][\Theta]^{-1}$ and therefore no occurrence of p in $[\Delta]$

So for every primitive type, and whatever its number of occurrences in Δ is, there are exactly as many occurrences of p in B and in $[\Delta]$, so the number of primitive types in B and in $[\Delta]$ are equal: $|B| = \|[\Delta]\|$. ◇

Proposition 33 *Let $A_1,\ldots,A_n \vdash A_{n+1}$ be a thin sequent with $|A_i| \leq m$; then either:*

- *there exists an index k and a type B with $|B| \leq m$ such that the following sequents are thin:*

$$A_1,\ldots,A_{k-1},B,A_{k+2},\ldots,A_n \vdash A_{n+1}$$
$$A_k,A_{k+1} \vdash B$$

- *there exist a type B with $|B| \leq m$ such that the following sequents are thin:*

$$B,A_n \vdash A_{n+1}$$
$$A_1,\ldots,A_{n-1} \vdash B$$

PROOF : Let $u_i = [A_i]$ for $1 \leq i \leq n$ and $u_{n+1} = [C]^{-1}$. Interpreting the provability in the free group we obtain: $u_1 \cdots u_n u_{n+1} = 1$ By lemma 30 there exist an index $k \leq n$ for which $\|u_k u_{k+1}\| \leq \max(\|u_k\|,\|u_{k+1}\|) \leq m$.

If $k < n$, we apply proposition 32 with

$$\Delta = A_k,A_{k+1},$$
$$\Gamma = A_1,\ldots,A_{k-1}$$
$$\Theta = A_{k+2},\ldots,A_n.$$

So the sequents

$$A_1,\ldots,A_{k-1},B,A_{k+2},\ldots,A_n \vdash A_{n+1}$$
$$A_k,A_{k+1} \vdash B$$

are thin, and

$$|B| = \|[A_k,A_{k+1}]\| = \|u_k u_{k+1}\| \leq m$$

If $k = n$, we apply proposition 32 with

$$\Gamma = \varepsilon,$$
$$\Delta = A_1,\ldots,A_{n-1}$$
$$\Theta = A_n.$$

So the sequents $A_1,\ldots,A_n \vdash B$ and $B,A_n \vdash B$ are thin.

Since $[A_1,\ldots,A_{n-1},A_n] = [C]$
we have $|B| = \|[A_1,\ldots,A_{n-1}]\| = \|[C][A_n]^{-1}\|$

therefore

$$|B| = \|[C][A_n]^{-1}\| = \|([A_n][C]^{-1})^{-1}\| = \|(u_n u_{n+1})^{-1}\| = \|u_n u_{n+1}\| \leq m$$

$\diamond$

2.13.4 From Lambek grammars to context-free grammars

Proposition 34 *If a sequent $A_1,\ldots,A_n \vdash A_{n+1}$ with each $|A_i| \leq m$ is provable in L, then it is provable from provable sequents $U,V \vdash X$ or $U \vdash X$ with $|U|,|V|,|X| \leq m$ by means of the cut rule only.*

PROOF : We proceed by induction on n. If $n \leq 2$ then there is nothing to prove. Otherwise, let $A'_1,\ldots,A'_n \vdash A'_{n+1}$ be a corresponding thin sequent obtained as in proposition 31 — using a different primitive type for each axiom in the proof of $A_1,\ldots,A_n \vdash A_{n+1}$. Thus there exists a substitution σ replacing primitive types with primitive types and preserving provability such that $\sigma(A') = A$.

As the substitution replaces primitive types with primitive types, we also have $|A'_i| \leq m$. By proposition 33 there exists a formula B' with $|B| \leq m$ such that either:

- $A'_1,\ldots,A'_{k-1},B',A'_{k+2},\ldots,A'_n \vdash A'_{n+1}$
  $A'_k,A'_{k+2} \vdash B'$
  are thin, and therefore provable.
  Let $B = \sigma(B')$, so B has at most m primitive types as well; applying the substitution we obtain two provable sequents

  $A_1,\ldots,A_{k-1},B,A_{k+2},\ldots,A_n \vdash A_{n+1}$
  $A_k,A_{k+1} \vdash B$.

  By induction hypothesis

  $$A_1,\ldots,A_{k-1},B,A_{k+2},\ldots,A_n \vdash A_{n+1} \qquad (*)$$

  is provable from provable sequents $U,V \vdash X$ or $U \vdash X$ with with $|U|,|V|,|X| \leq m$ by means of the cut rule only.
  Notice that $A_k,A_{k+1} \vdash B \quad (**)$ is of the form $U,V \vdash X$ with $|U|,|V|,|X| \leq m$.
  A cut rule between the proof of $(*)$ and $(**)$ yields a proof of $A_1,\ldots,A_n \vdash A_{n+1}$ from provable sequents $U,V \vdash X$ or $U \vdash X$ with $|U|,|V|,|X| \leq m$ by means of the cut rule only.

- $B', A'_n \vdash A'_{n+1}$ and $A_1, \ldots, A_{n-1} \vdash B$ are thin and therefore provable.
  Let $B = \sigma(B')$, so $|B| \leq m$; applying the substitution we obtain two provable sequents

  $$B, A_n \vdash A_{n+1}$$
  $$A_1, \ldots, A_{n-1} \vdash B.$$

  By induction hypothesis

  $$A_1, \ldots, A_{n-1}, B \vdash A_{n+1} \qquad (+)$$

  is provable from provable sequents $U, V \vdash X$ or $U \vdash X$ with U, V, X having at most m primitive types by means of the cut rule only.
  Notice that $B, A_n \vdash A_{n+1} \quad (++)$ is of the form $U, V \vdash X$ with $|U|, |V|, |X| \leq m$.
  A cut rule between the proof of $(+)$ and $(++)$ yields a proof of $A_1, \ldots, A_n \vdash A_{n+1}$ from provable sequents $U, V \vdash X$ or $U \vdash X$ with $|U|, |V|, |X| \leq m$ by means of the cut rule only.

$\diamond$

Theorem 35 *Let* Lex *be the lexicon of a Lambek grammar G_L, and let and let m the maximal number of primitive types in a formula of the lexicon. Then the language $L(G_L)$ generated by G_L is the same as the language $L(G_C)$ generated by the following context-free grammar G_C:*

- *Terminals: terminals (words) of G_L*
- *Non-Terminals: all formulae A with $|A| \leq m$*
- *Start symbol S, the one of G_L*
- *$X \longrightarrow a$ whenever $X \in \text{Lex}(a)$*
- *$X \longrightarrow A$ whenever $A \vdash X$ is provable in L*
- *$X \longrightarrow AB$ whenever $A, B \vdash X$ is provable in L*

Observe that the rules are in finite number, because there are finitely many sequents $U, V \vdash X$ or $U \vdash X$ when U, V, X contains at most m primitive types — hence there are only finitely many provable such sequents.

PROOF : Assume $a_1 \cdots a_n \in L(G_C)$. Hence there exist types $X_i \in \text{Lex}(a_i)$ such that $S \longrightarrow X_1 \cdots X_n$. The derivation in the CFG G_C can be turned into a derivation in L using only the cut rule (reversing $\longrightarrow$ and $\vdash$), therefore $a_1 \cdots a_n \in L(G_L)$.

Assume now that $a_1 \cdots a_n \in L(G_L)$. Hence there exist types $X_i \in \text{Lex}(a_i)$ such that $X_1, \ldots, X_n \vdash S$. By proposition 34 such a sequent is provable by means of the

sequents corresponding to production rules, and of the cut rule only. By induction on the size of the cut-only proof, it is easily seen that the proof corresponds to a derivation in the CFG G_C. If the proof is reduced to a proper axiom, than this axiom is itself a production rule. If the last rule is a cut, say between $\Gamma, B, \Theta \vdash C$ and $\Delta \vdash B$, then by induction hypothesis we have $B \longrightarrow \Delta$ and $C \longrightarrow \Gamma B \Theta$ hence $C \longrightarrow \Gamma \Delta \Theta$. Thus, if $a_1 \cdots a_n \in L(G_L)$, we have $S \longrightarrow X_1 \cdots X_n$ with $X_i \in \mathrm{Lex}(A_i)$; as $X_i \in \mathrm{Lex}(a_i)$ we have $S \longrightarrow a_1 \cdots a_n$. ◇

2.14 Lambek calculus and Montague semantics

So far the main interest of categorial grammars are that they are lexicalized. Now we will turn our attention to their relation to Montague semantics, introduced in [60] which is one very important feature of categorial grammars.

We do not give a lecture on Montague semantics, which is a wide area and the reader interested in this topic is referred to [61,62]. Montague semantics is also a controversial view of semantics. Indeed this semantics contains nothing fancy about mental representation or the organization of concepts as for instance in [63] or [64]: semantics is depicted by formulae of predicate calculus, possibly of intentional logic, and the notions are represented by logical constants. Nevertheless it enables a neat and computational treatment of (co)reference and of quantifiers and this is very important — although according to generative grammar, these questions belong to syntax.

After this warning, let us come back to the relation between Montague semantics and categorial grammars. This is due to the following fact, studied in particular by van Benthem (see [43]): simply typed λ-terms which represent formulae of predicate calculus and neatly handle substitution are very close to proofs in the Lambek calculus, that are syntactic analyses. Indeed, via the Curry-Howard isomorphism (see e.g. [14]) simply typed λ-terms are proofs in intuitionistic logic which embeds Lambek calculus. Indeed, reading $a \setminus b$ and b / a as $a \to b$ (intuitionistic implication) each rule of the Lambek calculus is a rule of intuitionistic logic. Assume our Lambek grammar uses

[60] Richard 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.

[61] L. T. F. Gamut. *Logic, Language and Meaning*, volume 2. The University of Chicago Press, 1991.

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

[63] Ray Jackendoff. *The Architecture of the Language Faculty*. Number 28 in Linguistic Inquiry Monographs. M.I.T. Press, Cambridge, Massachusetts, 1995.

[64] James Pustejovsky. *The generative lexicon*. M.I.T. Press, 1995.

[43] Johan 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.

[14] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

the primitive types: np, n, S. First let us define a morphism from syntactic types to semantic types : these semantic types are formulae are define from two types e (entities) and t (truth values or propositions) with the intuitionistic implication $\rightarrow$ as their only connective:

$$types ::= e \quad | \quad t \quad | \quad types \rightarrow types$$

Thus a common noun like *chair* or an intransitive verb like *sleep* have the type $e \rightarrow t$ (the set of entities which are chairs or who sleep) a transitive verb like *takes* is a two place predicate of type $e \rightarrow (e \rightarrow t)$ (the pairs of entities such that the first one takes the second one) etc.

Thus we can define a morphism from syntactic types to semantic types:

$(\text{Syntactic type})^*$	$=$	Semantic type	
S^*	$=$	t	a sentence is a proposition
np^*	$=$	e	a noun phrase is an entity
n^*	$=$	$e \rightarrow t$	a noun is a subset of the set of entities
$(a \setminus b)^* = (b / a)^*$	$=$	$a^* \rightarrow b^*$	extends $(\_)^*$ to all syntactic types

The lexicon associates to each syntactic type $t_k \in \text{Lex}(m)$ of a word m a λ-term τ_k *whose type is precisely* t_k^*, the semantic counter part of the syntactic type t_k. We use implicit right bracketing for types: $a \rightarrow b \rightarrow c = a \rightarrow (b \rightarrow c)$, which goes with implicit left bracketing for λ-terms: $w\ v\ u = (w\ v)\ u$. We need constants for usual logical operations like quantification, conjunction etc. :

Constant	Type
$\exists$	$(e \rightarrow t) \rightarrow t$
$\forall$	$(e \rightarrow t) \rightarrow t$
$\wedge$	$t \rightarrow (t \rightarrow t)$
$\vee$	$t \rightarrow (t \rightarrow t)$
$\supset$	$t \rightarrow (t \rightarrow t)$

and proper constants for the denotation of the words in the lexicon:

likes	$\lambda x \lambda y\ (\text{likes}\ x)\ y$	$x : e,\ y : e, \text{likes} : e \rightarrow (e \rightarrow t)$
`<< likes >> is a two-place predicate`		
Pierre	$\lambda P\ (P\ \text{Pierre})$	$P : e \rightarrow t,\ \text{Pierre} : e$
`<< Pierre >> is viewed as the properties that << Pierre >> holds`		

These constants can include intentionality operators $\hat{}$ and $\check{}$ but we do not present them : indeed they are only introduced in the lexicon (and not in the computations) , they are not modified during the computations, and do not modify the algorithm we

are to present. On the other hand a presentation of intentionality would take us too far for this lecture, we refer the reader to [61].

Given

- *a syntactic analysis of* $m_1 \dots m_n$ in Lambek calculus, that is a proof $\mathscr{D}$ of $t_1, \dots, t_m \vdash S$ and
- *the semantics of each word* m_1,*... et* m_n, that are λ-terms $\tau_i : t_i^*$,

we obtain the semantics of the sentence by the following algorithm,

1. Replace every syntactic type in $\mathscr{D}$ with its semantic counterpart; since intuitionistic logic extends the Lambek calculus the result $\mathscr{D}^*$ of this operation is a proof in intuitionistic logic of $t_1^*, \dots, t_n^* \vdash t = S^*$.
2. Via the Curry-Howard isomorphism, this proof in intuitionistic logic can be viewed as a simply typed λ-term $\mathscr{D}_\lambda^*$ which contains one free variable x_i of type t_i^* per word m_i.
3. Replace in $\mathscr{D}_\lambda^*$. each variable x_i by the λ-term τ_i — whose type is also type t_i^*, so this is a correct substitution.
4. Reduce the resulting λ-term: this provides the semantics of the sentence (another syntactic analysis of the same sentence can lead to a different semantics).

We used natural deduction, because natural deduction is closer to λ-terms, but, if one prefers, one can or use sequent calculus and cut-elimination.

Why does the final λ-term corresponds to a proposition, or a closed predicate calculus formula? It is because in the semantic λ-terms all constants which are not logical connectives have types $e \to e \to \dots \to e$ (functions) or $e \to e \to \dots \to t$ (predicates) ; it is easily observed that every normal λ-term of type t with only constants of such types correspond to a formula of predicate calculus.

2.14.1 An example

Consider the following lexicon:

---

[61] L. T. F. Gamut. *Logic, Language and Meaning*, volume 2. The University of Chicago Press, 1991.

word	***syntactic type*** u ***semantic type*** u^* ***semantics :*** λ***-term of type*** u^* x_v ***means that the variable or constant*** x ***is of type*** v
some	$(S \,/\, (np \setminus S)) \,/\, n$ $(e \to t) \to ((e \to t) \to t)$ $\lambda P_{e\to t}\ \lambda Q_{e\to t}\ (\exists_{(e\to t)\to t}\ (\lambda x_e (\wedge_{t\to(t\to t)} (P\ x)(Q\ x))))$
statements	n $e \to t$ $\lambda x_e (\texttt{statement}_{e\to t}\ x)$
speak_about	$(np \setminus S) \,/\, np$ $e \to (e \to t)$ $\lambda y_e\ \lambda x_e\ ((\texttt{speak\_about}_{e\to(e\to t)}\ x) y)$
themselves	$((np \setminus S) \,/\, np) \setminus (np \setminus S)$ $(e \to (e \to t)) \to (e \to t)$ $\lambda P_{e\to(e\to t)}\ \lambda x_e\ ((P\ x) x)$

Let us first show that *Some statements speak about themselves.* belongs to the language generated by this lexicon. So let us prove (in natural deduction) the following :

$$(S \,/\, (np \setminus S)) \,/\, n\ ,\ n\ ,\ (np \setminus S) \,/\, np\ ,\ ((np \setminus S) \,/\, np) \setminus (np \setminus S) \vdash S$$

using the abbreviations So (some) Sta (statements) SpA (speak about) $Refl$ (themselves) for the syntactic types :

$$\dfrac{\dfrac{So \vdash (S/(np\backslash S))/n \quad Sta \vdash n}{So, Sta \vdash (S/(np\backslash S))}\,/_e \quad \dfrac{SpA \vdash (np\backslash S)/np \quad Refl \vdash ((np\backslash S)/np)\backslash(np\backslash S)}{SpA, Refl \vdash (np\backslash S)}\,\backslash_e}{So, Sta, SpA, Refl \vdash S}\,/_e$$

Using the homomorphism from syntactic types to semantic types we obtain the following intuitionistic deduction, where $So^*, Sta^*, SpA^*, Refl^*$ are abbreviations for the semantic types respectively associated with the syntactic types: $So, Sta, SpA, Refl$:

$$\dfrac{\dfrac{So^* \vdash (e\to t)\to(e\to t)\to t \quad Sta^* \vdash e\to t}{So^*, Sta^* \vdash (e\to t)\to t}\to_e \quad \dfrac{SpA^* \vdash e\to e\to t \quad Refl^* \vdash (e\to e\to t)\to e\to t}{SpA^*, Refl^* \vdash e\to t}\to_e}{So^*, Sta^*, SpA^*, Refl^* \vdash t}\to_e$$

The λ-term representing this deduction simply is

((some statements) (themsleves speak_about)) of type t

where *some,statements,themselves,speak_about* are variables with respective types $So^*, Sta^*, Refl^*, SpA^*$. Let us replace these variables with the semantic λ-terms (of the

same type) which are given in the lexicon. We obtain the following λ-term of type t (written on two lines) that we reduce:

$$\Big(\big(\lambda P_{e\to t}\ \lambda Q_{e\to t}\ (\exists_{(e\to t)\to t}\ (\lambda x_e(\wedge(P\ x)(Q\ x))))\big)\big(\lambda x_e(\texttt{statement}_{e\to t}\ x)\big)\Big)$$
$$\Big(\big(\lambda P_{e\to(e\to t)}\ \lambda x_e\ ((P\ x)x)\big)\big(\lambda y_e\ \lambda x_e\ ((\texttt{speak\_about}_{e\to(e\to t)}\ x)y)\big)\Big)$$

$$\downarrow \beta$$

$$\big(\lambda Q_{e\to t}\ (\exists_{(e\to t)\to t}\ (\lambda x_e(\wedge_{t\to(t\to t)}(\texttt{statement}_{e\to t}\ x)(Q\ x)))))$$
$$\big(\lambda x_e\ ((\texttt{speak\_about}_{e\to(e\to t)}\ x)x)\big)$$

$$\downarrow \beta$$

$$\big(\exists_{(e\to t)\to t}\ (\lambda x_e(\wedge(\texttt{statement}_{e\to t}\ x)((\texttt{speak\_about}_{e\to(e\to t)}\ x)x)))\big)$$

This term represent the following formula of predicate calculus (in a more pleasant format) :

$$\exists x : e\ (\texttt{statement}(x) \wedge \texttt{speak\_about}(x,x))$$

This is the semantics of the analyzed sentence.

2.14.2 An exercice

As an exercice on Lambek calculus and Montague semantics, one can verify that there are two readings of the sentence:

A number corresponds_to every student.

Firstly, one has to provide types for ”a” and ”every” in subject and object position.

Then construct two different syntactic analyses of the sentence.

Finally observe that one analysis gives to the existential ”a” the widest scope while the other gives to the universal quantifier ”every” the widest scope.

Chapter 3

Lambek calculus and linear logic: proof-nets as parse structures

This chapter, a large part of which is a translation of [65]*, deals with the connection between Lambek categorial grammar and linear logic, the main objective being the presentation of proof-nets which are excellent parse structures, because they identify equivalent analyses of a given sentence.*

This graphical notation for proofs that are parse structures in categorial grammar is a not a mere variation for convenience. On a technical ground, it avoids the so called spurious ambiguities of categorial grammars (many different proof/parse structures for a single analysis). Conceptually this proof syntax gives sense to the expression parsing as deduction *often associated with categorial grammar. Indeed the proof-nets only make distinction between proofs which do correspond to different syntactic analyses.*

We first give a rather complete presentation of the correspondance between Lambek calculus and variants of multiplicative linear logic, since Lambek calculus can be defined as non-commutative intuitionnistic multiplicative linear logic without empty antecedents.

Next we define proof-nets and establish their correspondence with the more traditional sequent calculus, and present parsing as proof-net construction and present some recent description of non commutative proof-nets.

Finally, as an evidence of their linguistic relevance, we explain how they provide a formal account of some performance questions, like the complexity of the processing of several intricate syntactic constructs, like center embedded relatives, garden pathing, or preferred readings.

---

[65] Christian Retoré. Calcul de Lambek et logique linéaire. *Traitement Automatique des Langues*, 37(2):39–70, 1996.

3.1 Categorial language and linear logic language

3.1.1 Multiplicative linear logic language

Let us recall the language of the Lambek calculus:

$$\mathsf{Lp} ::= \mathsf{P} \mid \mathsf{Lp} \bullet \mathsf{Lp} \mid \mathsf{Lp} \,/\, \mathsf{Lp} \mid \mathsf{Lp} \setminus \mathsf{Lp}$$

As we have seen in the previous chapter $\setminus$ and $/$ are implications, and the product $\bullet$ is a conjunction. All these connectives are linear logic connectives, but are rather denoted by: $\circ\!\!-$, $-\!\!\circ$, $\otimes$ in the linear logic community.

Lambek calculus	$\setminus$	$/$	$\bullet$
Linear logic	$-\!\!\circ$	$\circ\!\!-$	$\otimes$

Multiplicative linear logic is a classical calculus which extends Lambek calculus by a negation denoted by $(\ldots)^{\perp}$ (*the orthogonal of* $\ldots$) together with the symmetries induced by a classical negation, that are the familiar De Morgan identities of classical logic.

To be precise, usual Multiplicative Linear Logic extends Lambek calculus without the non empty antecedent requirement, and allows for permutation (hypotheses can be permuted). One must restrict permutations to cyclic permutations for having a single involutive negation and two distinct implications $\circ\!\!-$ et $-\!\!\circ$, and in the absence of any form of permutation, there has to be two negations [66,67].

Because of the De Morgan identities, there will be a disjunction $\wp$ (*par*, standing for *in parallel with*) corresponding to the conjunction $\otimes$. As we are especially interested in having a non commutative conjunction so will be, by duality, the disjunction.

Such a disjunction and a classical negation allows for the implication $A \setminus B$ to be defined as $A^{\perp} \wp B$ and for the implication $B \,/\, A$ to be defined as $B \wp A^{\perp}$ — as one can defined $A \Rightarrow B$ as $\neg A \vee B$ in classical logic. Noticed that the non commutativity of the disjunction is needed for having two distinct implications.

In the Lambek calculus, one has the following equivalence: $(C \,/\, B) \,/\, A \equiv C \,/\, (A \otimes B)$: indeed $(C \,/\, B) \,/\, A$ is a formula which requires an A and then a B to obtain C, and $C \,/\, (A \otimes B)$ is a formula which requires an *A_followed_by_a_B*, to obtain a C. The formula $(C \,/\, B) \,/\, A$ can be written as $C \wp B^{\perp} \wp A^{\perp}$ using the (associative) disjunction and the formula $C \,/\, (A \otimes B)$ as $C \wp (A \otimes B)^{\perp}$. Therefore if there is a classical extension of the Lambek calculus then negation has to swap the components of a disjunction (and of a conjunction, by duality).

---

[66] Vito Michele Abrusci. Phase semantics and sequent calculus for pure non-commutative classical linear logic. *Journal of Symbolic Logic*, 56(4):1403–1451, 1991.

[67] Vito Michele Abrusci. Non- commutative proof nets. In Girard et al. [2], pages 271–296.

[2] Jean-Yves Girard, Yves Lafont, and Laurent Regnier, editors. *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Notes*. Cambridge University Press, 1995.

Linear logic, a classical extension of the Lambek calculus as the following language:

$$\mathsf{Li}_+ ::= \mathsf{P} \mid \mathsf{Li}_+^\perp \mid \mathsf{Li}_+ \parr \mathsf{Li}_+ \mid \mathsf{Li}_+ \otimes \mathsf{Li}_+ \mid \mathsf{Li}_+ \setminus \mathsf{Li}_+ \mid \mathsf{Li}_+ / \mathsf{Li}_+$$

and enjoys the following De Morgan identities:

$$(A^\perp)^\perp \equiv A \qquad (A \parr B)^\perp \equiv B^\perp \otimes A^\perp \qquad (A \otimes B)^\perp \equiv B^\perp \parr A^\perp$$

3.1.2 Reduced linear language (negative normal form)

For every formula X in Li_+ there exists a unique equivalent formula $+X$ such that negation only applies to propostional variables, and its only connectives are conjunction and disjunction. In some books, the analogous of $+X$ for classical logic is called its negative normal form. The fromula $+X$ is obtained by replacing its implication by its definition as a disjunction, and then apllying De morgan identities as rewriting rules from left to right, and, finally by concelling double negations. Notice that this form does not require distibutivity of $\parr$ w.r.t. $\otimes$ or $\otimes$ w.r.t. $\parr$ — these distributivity identities do not hold in linear logic.

So every formula in Li_+ is equivalent to a formula $+X$ in Li, where Li is:

$$\mathsf{Li} ::= \mathsf{N} \mid \mathsf{Li} \parr \mathsf{Li} \mid \mathsf{Li} \otimes \mathsf{Li} \qquad \text{where } \mathsf{N} = \mathsf{P} \cup \mathsf{P}^\perp \text{ is the set of } \textit{atoms}.$$

Observe that if $F \in \mathsf{Li}$ then $+F = F$.

Let us denote by $-F$ the unique formula in Li equivalent to $(F)^\perp \in \mathsf{Li}_+$ — $-F = +(F^\perp)$. Given $+F$, $-F$ is obtained be replacing every propositional variable in $+F$ with its negation, every conjunction by a disjunction, every disjunction by a conjunction, and finally by *reverting the left right order of the result*.

Given $F = (\alpha^\perp \parr \beta) \otimes \gamma^\perp$ one first obtains $F' = (\alpha \otimes \beta^\perp) \parr \gamma$, which yields $F^\perp \equiv -F = \gamma \parr (\beta^\perp \otimes \alpha)$ by rewriting F' from right to left.

3.1.3 Relating categories and linear logic formulae : polarities

Since Lp is a sublanguage of Li_+ every formula L in Lp there exists a unique formula $+L$ in Li which is equivalent to L and a unique formula $-L$ which is equivalent to $L^\perp$. These two maps from Lp to Li can be inductively defined as follows:

L	$\alpha \in \mathsf{P}$	$L = M \bullet N$	$L = M \setminus N$	$L = N / M$
$+L$	α	$+M \otimes +N$	$-M \parr +N$	$+N \parr -M$
$-L$	$\alpha^\perp$	$-N \parr -M$	$-N \otimes +M$	$+M \otimes -N$

EXAMPLE **36**

L	$+L$	$-L$	
np	np	$np^\perp$	noun phrase
$np\,/\,n$	$np^\perp \parr n$	$n^\perp \otimes np$	determiner
n	n	$n^\perp$	common noun
$n\setminus n$	$n^\perp \parr n$	$n^\perp \otimes n$	right adjective
$(n\setminus n)\,/\,(n\setminus n)$	$(n^\perp \parr n)\parr(n^\perp\otimes n)$	$(n^\perp \parr n)\otimes(n^\perp\otimes n)$	left modifier for right adjectives
$\beta\setminus((\alpha\,/\,\beta)\setminus\alpha)$	$\beta^\perp \parr((\beta\otimes\alpha^\perp)\parr\alpha)$	$(\alpha^\perp\otimes(\alpha\parr\beta^\perp))\otimes\beta)$	type raising

Let us consider the following sets of formulae, which enables to recognize, among linear formulae the ones which are Lambek formulae or the negation of Lambek formulae.

$\mathsf{Li}^\circ = \{F \in \mathsf{Li} / \exists L \in \mathsf{Lp} \quad +L = F\}$: positive linear formulae
$\mathsf{Li}^\bullet = \{F \in \mathsf{Li} / \exists L \in \mathsf{Lp} \quad -L = F\}$: negative linear formulae
$\mathsf{Li}^\circ \cup \mathsf{Li}^\bullet$: intuitionistic or polarized linear formulae

We then have:

$F \in \mathsf{Li}^\bullet \Leftrightarrow -F \in \mathsf{Li}^\circ$ et $F \in \mathsf{Li}^\circ \Leftrightarrow -F \in \mathsf{Li}^\bullet$
$\mathsf{Li}^\circ \cup \mathsf{Li}^\bullet \neq \mathsf{Li}$ — for instance $\alpha \parr \beta \notin \mathsf{Li}^\circ \cup \mathsf{Li}^\bullet$
$\mathsf{Li}^\bullet \cap \mathsf{Li}^\circ = \emptyset$ — because of the following proposition :

Proposition 37 *The sets of formulae* Li° *and* $\mathsf{Li}^\bullet$ *are inductively defined by:*

$$\left[\begin{array}{lcl} \mathsf{Li}^\circ & ::= & \mathsf{P} \mid \mathsf{Li}^\circ\otimes\mathsf{Li}^\circ \mid \mathsf{Li}^\bullet \parr \mathsf{Li}^\circ \mid \mathsf{Li}^\circ \parr \mathsf{Li}^\bullet \\ \mathsf{Li}^\bullet & ::= & \mathsf{P}^\perp \mid \mathsf{Li}^\bullet \parr \mathsf{Li}^\bullet \mid \mathsf{Li}^\circ\otimes\mathsf{Li}^\bullet \mid \mathsf{Li}^\bullet\otimes\mathsf{Li}^\circ \end{array}\right.$$

The maps $+$ *et* $-$ *are bijections from* Lp *to* Li° *and* $\mathsf{Li}^\bullet$ *respectively.*

If $(\ldots)^\circ_{\mathsf{Lp}}$ denotes the inverse bijection of $+$, from Li° to Lp and if $(\ldots)^\bullet_{\mathsf{Lp}}$ denotes the inverse bijection of $-$ from $\mathsf{Li}^\bullet$ to Lp. then theses two maps are inductively defined as follows:

$F\in\mathsf{Li}^\circ$	$\alpha\in\mathsf{P}$	$(G\in\mathsf{Li}^\circ)\otimes(H\in\mathsf{Li}^\circ)$	$(G\in\mathsf{Li}^\bullet)\parr(H\in\mathsf{Li}^\circ)$	$(G\in\mathsf{Li}^\circ)\parr(H\in\mathsf{Li}^\bullet)$
F°_{Lp}	α	$G^\circ_{\mathsf{Lp}}\otimes H^\circ_{\mathsf{Lp}}$	$G^\bullet_{\mathsf{Lp}}\setminus H^\circ_{\mathsf{Lp}}$	$G^\circ_{\mathsf{Lp}}\,/\,H^\bullet_{\mathsf{Lp}}$
$F\in\mathsf{Li}^\bullet$	$\alpha^\perp\in\mathsf{P}^\perp$	$(G\in\mathsf{Li}^\bullet)\parr(H\in\mathsf{Li}^\bullet)$	$(G\in\mathsf{Li}^\circ)\otimes(H\in\mathsf{Li}^\bullet)$	$(G\in\mathsf{Li}^\bullet)\otimes(H\in\mathsf{Li}^\circ)$
$F^\bullet_{\mathsf{Lp}}$	α	$H^\bullet_{\mathsf{Lp}}\otimes G^\bullet_{\mathsf{Lp}}$	$H^\bullet_{\mathsf{Lp}}\,/\,G^\circ_{\mathsf{Lp}}$	$H^\circ_{\mathsf{Lp}}\setminus G^\bullet_{\mathsf{Lp}}$

The inductive definition of Li° and $\mathsf{Li}^{\bullet}$ yields an easy decision procedure to check whether a formula F is in Li° or $\mathsf{Li}^{\bullet}$ — if so, all sub-formulae of F are in Li° or in $\mathsf{Li}^{\bullet}$: replace every propositional variable with $\circ$ and every negation of a propositional variable with $\bullet$ and compute using $\wp$ et $\otimes$ as the following operations on $\star, \circ, \bullet$:

$\wp$	$\star$	$\circ$	$\bullet$
$\star$	$\star$	$\star$	$\star$
$\circ$	$\star$	$\star$	$\circ$
$\bullet$	$\star$	$\circ$	$\bullet$

$\otimes$	$\star$	$\circ$	$\bullet$
$\star$	$\star$	$\star$	$\star$
$\circ$	$\star$	$\circ$	$\bullet$
$\bullet$	$\star$	$\bullet$	$\star$

The result of this simple computation is used as follows:

- $\star$ whenever the formula is neither in Li° nor in $\mathsf{Li}^{\bullet}$
- $\circ$ whenever the formula is in Li°
- $\bullet$ whenever the formula is in $\mathsf{Li}^{\bullet}$

EXAMPLE **38**

F	computation	conclusion	F°_{Lp}	$F^{\bullet}_{\mathsf{Lp}}$
$(\alpha^{\perp} \wp \beta) \wp \alpha$	$(\bullet \wp \circ) \wp \circ = \circ \wp \circ = \star$	$F \notin \mathsf{Li}^{\circ} \cup \mathsf{Li}^{\bullet}$	undefined	undefined
$(\alpha^{\perp} \wp \beta) \wp \alpha^{\perp}$	$(\bullet \wp \circ) \wp \bullet = \circ \wp \bullet = \circ$	$F \in \mathsf{Li}^{\circ}$	$(\alpha \setminus \beta) / \alpha$	undefined
$(\alpha^{\perp} \wp \beta) \otimes \alpha^{\perp}$	$(\bullet \wp \circ) \otimes \bullet = \circ \otimes \bullet = \bullet$	$F \in \mathsf{Li}^{\bullet}$	undefined	$\alpha / (\alpha \setminus \beta)$

3.2 Two sided calculi

Here is the both sided linear calculus MLL_{+} fro all connectives of the language Li_{+}. In the next section, we shall see how it embeds the Lambek calculus.

Exchange	$\dfrac{\Gamma,A,B,\Delta\vdash\Psi}{\Gamma,B,A,\Delta\vdash\Psi}\,(\mathbf{x})_h$ $\qquad$ $\dfrac{\Theta\vdash\Gamma,A,B,\Delta}{\Theta\vdash\Gamma,B,A,\Delta}\,(\mathbf{x})_i$
Axiom	$\dfrac{}{A\vdash A}\,ax$ $\qquad A\in \mathsf{Li}_+$
Logical rules	$\dfrac{\Gamma\vdash A,\Delta}{A^\perp,\Gamma\vdash\Delta}\,{}^\perp{}_h$ Negation $\dfrac{A,\Gamma\vdash\Delta}{\Gamma\vdash A^\perp,\Delta}\,{}^\perp{}_i$
	$\dfrac{\Gamma,A\vdash\Theta \qquad B,\Gamma'\vdash\Theta'}{\Gamma,A\parr B,\Gamma'\vdash\Theta,\Theta'}\,\parr_h$ Disjunction $\dfrac{\Theta\vdash\Gamma,A,B,\Delta}{\Theta\vdash\Gamma,A\parr B,\Delta}\,\parr_h$
	$\dfrac{\Gamma,A,B,\Delta\vdash\Psi}{\Gamma,A\otimes B,\Delta\vdash\Psi}\,\otimes_h$ Conjunction $\dfrac{\Theta\vdash\Phi,A \qquad \Theta'\vdash B,\Phi'}{\Theta,\Theta'\vdash\Phi,A\otimes B,\Phi'}\,\otimes_i$
	$\dfrac{\Gamma\vdash\Phi,A \qquad \Gamma',B,\Delta'\vdash\Psi'}{\Gamma',\Gamma,A\setminus B,\Delta'\vdash\Phi,\Psi'}\,\setminus_h$ Implications $\dfrac{A,\Gamma\vdash C,\Phi}{\Gamma\vdash A\setminus C,\Phi}\,\setminus_i$
	$\dfrac{\Gamma\vdash\Phi,A \qquad \Gamma',B,\Delta'\vdash\Psi'}{\Gamma',B/A,\Gamma,\Delta'\vdash\Phi,\Psi'}\,/_h$ $\qquad$ $\dfrac{\Gamma,A\vdash\Phi,C}{\Gamma\vdash\Phi,C/A}\,/_i$

3.2.1 Properties of the linear two sided sequent calculus

3.2.1.1 Cut elimination

We left out the cut rule on purpose. There are two ways to formulate the cut rule in a classical calculus:

$$\frac{\Theta\vdash\Phi,A \qquad A,\Theta'\vdash\Psi'}{\Theta,\Theta'\vdash\Phi,\Psi'}\,cut \qquad \frac{\Theta\vdash\Phi,A \qquad \Theta'\vdash A^\perp,\Phi'}{\Theta,\Theta'\vdash\Phi,\Phi'}\,cut$$

As in the Lambek calculus, this rule is redundant, and the proof is more or less the same. As a consequence, the subformula property also holds for this calculus.

3.2.1.2 De Morgan identities

As we earlier claimed, these identities hold for linear logic. For instance:

$$\dfrac{\dfrac{A\vdash A}{A^\perp,A\vdash}\,{}^\perp{}_i}{A\vdash(A^\perp)^\perp}\,{}^\perp{}_i \qquad \dfrac{\dfrac{A\vdash A}{\vdash A^\perp,A}\,{}^\perp{}_i}{(A^\perp)^\perp\vdash A}\,{}^\perp{}_i$$

3.2.1.3 Eta expansion

As for the Lambek calculus, an easy induction on A, shows that every axiom $A \vdash A$ can be derived from axioms $\alpha \vdash \alpha$, where α is a propositional variable, without using the exchange rule. For instance let us show that $A \vdash A$ with $A = \alpha \parr \beta^{\perp}$ can be derived from the axioms $\alpha \vdash \alpha$ and $\beta \vdash \beta$:

$$\dfrac{\dfrac{\dfrac{}{\alpha \vdash \alpha}\, ax \qquad \dfrac{\dfrac{\dfrac{}{\beta \vdash \beta}\, ax}{\beta, \beta^{\perp} \vdash}\, \perp_h}{\beta^{\perp} \vdash \beta^{\perp}}\, \perp_i}{\alpha \parr \beta^{\perp} \vdash \alpha, \beta^{\perp}}\, \parr_h}{\alpha \parr \beta^{\perp} \vdash \alpha \parr \beta^{\perp}}\, \parr_i$$

3.2.1.4 Equality of the two implications

In this calculus, the implication $A \setminus B$ can be viewed as a shorthand for $A^{\perp} \parr B$, while A / B is a short hand for $B \parr A^{\perp}$. Indeed the implications rules can be derived when implications are defined this way. Furthermore, in the presence of a full exchange rule, one has: $A \setminus B \equiv B / A$.

$$\dfrac{\dfrac{\Gamma \vdash A}{\Gamma, A^{\perp} \vdash}\, \perp_h \qquad \Delta', B, \Gamma' \vdash \Theta'}{\Delta', \Gamma, A^{\perp} \parr B, \Gamma' \vdash \Theta'}\, \setminus_i \qquad \dfrac{\dfrac{\dfrac{A^{\perp} \vdash A \quad B \vdash B}{[A \setminus B \equiv A^{\perp} \parr B] \vdash A^{\perp}, B}\, \parr_h}{A \setminus B \vdash B, A^{\perp}}\, (\mathbf{x})_i}{A \setminus B \vdash [B \parr A^{\perp} \equiv B / A]}\, \parr_i$$

3.2.1.5 Negation and symmetrical rules

If one consider formulae up to De Morgan identities, then right rules are enough. For instance the rule $\parr_h$ can be simulated by the rule $\otimes_i$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, A \vdash \Theta}{\Gamma \vdash A^{\perp}, \Theta}\, \perp_i \qquad \dfrac{B, \Gamma' \vdash \Theta'}{\Gamma' \vdash B^{\perp}, \Theta'}\, \perp_i}{\Gamma, \Gamma' \vdash \Theta, A^{\perp} \otimes B^{\perp}, \Theta'}\, \otimes_i}{[A \parr B \equiv (A^{\perp} \otimes B^{\perp})^{\perp}], \Gamma', \Gamma \vdash \Theta', \Theta}\, (\mathbf{x})_h \text{ and } \perp_h}{\Gamma, [B \parr A \equiv (A^{\perp} \otimes B^{\perp})^{\perp}], \Gamma' \vdash \Theta', \Theta}\, (\mathbf{x})_h$$

In order to avoid the exchange rule, one has to consider a more subtle sequent

calculus like the one of [66, p. 1415] but identifying the two negations — this actually forces a restricted form of the exchange rule known as cyclic exchange, that we shall present later on.

3.2.2 The intuitionistic two sided calculus LP_ε

The calculus LP_ε, that is Lambek calculus with permutation and empty antecedents is exactly intuitionistic multiplicative linear logic. This calculus is obtained from MLL_+ by forcing sequents to always have exactly one formula on the right hand side.

It is easily seen that only negation and disjunction need at least two formulae on the right hand side. Therefore the natural language for LP_ε is Lp. The rules are obtained from the ones of MLL_+ in section 3.2, by replacing the sequences of formulae denoted by Φ and Φ' by the empty sequence, and the sequences of of formulae denoted by Ψ and Ψ' by a single formula F or F'. This yields the following rules:

Exchange	$\dfrac{\Gamma, A, B, \Delta \vdash F}{\Gamma, B, A, \Delta \vdash F}\,(\mathbf{x})_h$
Axiom	$\dfrac{}{A \vdash A}\,ax \qquad A \in \mathsf{Lp}$
Règles logiques	$\dfrac{\Gamma, A, B, \Delta \vdash F}{\Gamma, A \otimes B, \Delta \vdash F}\otimes_h$ Conjunction $\qquad \dfrac{\Theta \vdash A \qquad \Theta' \vdash B}{\Theta, \Theta' \vdash A \otimes B}\otimes_h$
	$\dfrac{\Gamma \vdash A \qquad \Gamma', B, \Delta' \vdash F'}{\Gamma', \Gamma, A \setminus B, \Delta' \vdash F'}\setminus_h$ $\qquad$ Implications $\qquad \dfrac{A, \Gamma \vdash C}{\Gamma \vdash A \setminus C}\setminus_h$
	$\dfrac{\Gamma \vdash A \qquad \Gamma', B, \Delta' \vdash F'}{\Gamma', B/A, \Gamma, \Delta' \vdash F'}/_h \qquad \dfrac{\Gamma, A \vdash C}{\Gamma \vdash C/A}/_i$

This calculus LP_ε and its variants are studied in a slightly different perspective in [43], and is also the basis of works on the semantics of LFG in a series of articles like [68].

This calculus allows for several variants according to the presence or absence of the exchange rule, or the allowance or prohibition of sequents with an empty antecedent, that is: the sequence of formulae Π is not empty when the rule $\setminus_i$ or $/_i$ is applied or, equivalently, every sequent in a proof has a non empty antecedent.

---

[66] Vito Michele Abrusci. Phase semantics and sequent calculus for pure non-commutative classical linear logic. *Journal of Symbolic Logic*, 56(4):1403–1451, 1991.

[43] Johan 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.

[68] Mary Dalrymple, John Lamping, Fernando Pereira, and Vijay Saraswat. Linear logic for meaning assembly. In Glyn Morrill and Richard Oehrle, editors, *Formal Grammar*, pages 75–93, Barcelona, 1995. FoLLI.

This last restriction is harmless from a logical viewpoint, i.e. preserves cut-elimination, but is essential for a grammatical use of the Lambek calculus, as we have seen in the previous chapter. let us give one more example of an uncorrect analysis due to empty antecedents:

EXAMPLE **39**

WORD	exemple	simple	très	un
TYPE(S)	n	$n \backslash n$	$(n \backslash n) / (n \backslash n)$	np / n

EXAMPLE **40**

$$\dfrac{\dfrac{}{n \vdash n}\,ax \qquad \dfrac{\dfrac{\dfrac{}{n \vdash n}\,ax \quad \dfrac{}{sn \vdash sn}\,ax}{sn/n, n \vdash sn}\,g/ \quad \dfrac{}{n \vdash n}\,ax}{sn / n, n, n \backslash n \vdash sn}\,g\backslash}{\boxed{\varepsilon} \vdash n / n}\,d/ \qquad\qquad}{\underset{\texttt{un}}{sn / n,} \quad \underset{\texttt{exemple}}{n,} \quad \underset{\texttt{très}}{(n \backslash n) / (n \backslash n)} \vdash sn}\,g/$$

3.2.3 Proof as parse structures: too many of them

Given a Lambek grammar, the parse structure should be proofs. Nevertheless it is easy to find several proofs which should correspond to the same parse structure, but which nevertheless are distinct. For instance, with the previous grammar lexicon, the following sequent calculus proofs are not equal although they are equivalent in the sense that they express the same valence consumption.

EXAMPLE **41**

$$\dfrac{\dfrac{\dfrac{\dfrac{}{n \vdash n}\,ax \quad \dfrac{}{n \vdash n}\,ax}{n, n \backslash n \vdash n}\,\backslash_h}{n \backslash n \vdash n \backslash n}\,\backslash_i \qquad \dfrac{\dfrac{\dfrac{}{n \vdash n}\,ax \quad \dfrac{}{n \vdash n}\,ax}{n, n \backslash n \vdash n}\,\backslash_h \quad \dfrac{}{np \vdash np}\,ax}{np / n, n, n \backslash n \vdash np}\,/_h}{\underset{\texttt{un}}{np \backslash n,} \quad \underset{\texttt{exemple}}{n,} \quad \underset{\texttt{très}}{(n \backslash n) / (n \backslash n),} \quad \underset{\texttt{simple}}{n \backslash n} \vdash np}\,/_h$$

EXAMPLE **42**

$$\dfrac{\dfrac{\dfrac{\dfrac{}{n \vdash n}\,ax \quad \dfrac{}{n \vdash n}\,ax}{n, n \backslash n \vdash n}\,\backslash_h}{n \backslash n \vdash n \backslash n}\,\backslash_i \qquad \dfrac{\dfrac{\dfrac{}{n \vdash n}\,ax \quad \dfrac{}{np \vdash np}\,ax}{np/n, n \vdash np}\,/_h \quad \dfrac{}{n \vdash n}\,ax}{np / n, n, n \backslash n \vdash np}\,\backslash_h}{\underset{\texttt{un}}{np \backslash n,} \quad \underset{\texttt{exemple}}{n,} \quad \underset{\texttt{très}}{(n \backslash n) / (n \backslash n),} \quad \underset{\texttt{simple}}{n \backslash n} \vdash np}\,/_h$$

One of the main objective of this chapter is to find a notion of proof that yields one proof per parse structure; this is a key motivation for proof-nets, to be introduced in section 3.4.

3.3 A one sided calculus for linear logic: MLL

As we have seen in the paragraph 3.1.2 fro every formula X of Li_+ there exists a unique formula $+X$ of Li which is equivalent to it by De Morgan identities, and as explained in paragraph 3.2.1.5, right rules can be simulated by left rules. Therefore, if one consider formulae up to De Morgan identities then the following one sided sequent calculus, defined as follows, is enough:

Exchange	$\dfrac{\vdash \Gamma, A}{\vdash A, \Gamma}\,(\mathsf{cx})$ $\qquad$ $\dfrac{\vdash \Gamma, A, B}{\vdash \Gamma, B, A}\,(\mathsf{tx})$
Axiom	$\dfrac{}{\vdash \alpha, \alpha^{\perp}}\,ax$ $\qquad \alpha \in \mathsf{P}$
Logical rules	$\dfrac{\vdash \Gamma, A, B, \Delta}{\vdash \Gamma, A \wp B, \Delta}\,\wp$ $\qquad$ $\dfrac{\vdash \Gamma, A \qquad \vdash B, \Gamma'}{\vdash \Gamma, A \otimes B, \Gamma'}\,\otimes$

The exchange rule $(\mathbf{x})_h$ of MLL_+ has been split into two rules (tx) (transposition exchange) and (cx) (cyclic exchange). Therefore $(\mathbf{x})_h$ is derivable but, this formulation allows to consider the calculus $\mathsf{NC\text{-}MLL}$ of [69], which only has the (cx) exchange, but not the (tx) exchange.

The simple calculus MLL whose language is Li,proves exactly the same sequents as the bigger two sided calculus MLL_+ :

Proposition 43 *Let $A_1, \ldots, A_n, B_1, \ldots, B_p$ be formulae in Li_+; then one has:*

$$\mathit{MLL}_+ \vdash (A_1, \ldots, A_n \vdash B_1, \ldots, B_p) \;\Leftrightarrow\; \mathit{MLL} \vdash (\vdash -A_n, \ldots, -A_1, +B_1, \ldots, +B_p)$$

For the converse implication, notice that given a formula $F \in \mathsf{Li}$ there usually exists several formulae $X \in \mathsf{Li}_+$ such that $+X = F$ or $-X = F$.

[69] David N. Yetter. Quantales and (non-commutative) linear logic. *Journal of Symbolic Logic*, 55:41–64, 1990.

3.3.1 Variants

We are about to introduce several variants of MLL according to the following restrictions:

INTUI **intuitionistic calculi**

in two sided presentation: one formula in the right hand side of every sequent

in one sided presentation: only polarized formulae (formulae of $\mathsf{Li}^{\circ\bullet}$)

NC **non commutative calculi**

in two sided presentation: no exchange at all

in one sided presentation: cyclic exchange (cx) only (no transposition exchange (tx))

ε-FREE **no empty antecedent**

in two sided presentation: no empty antecedent, at least one formula on the left hand side of every sequent

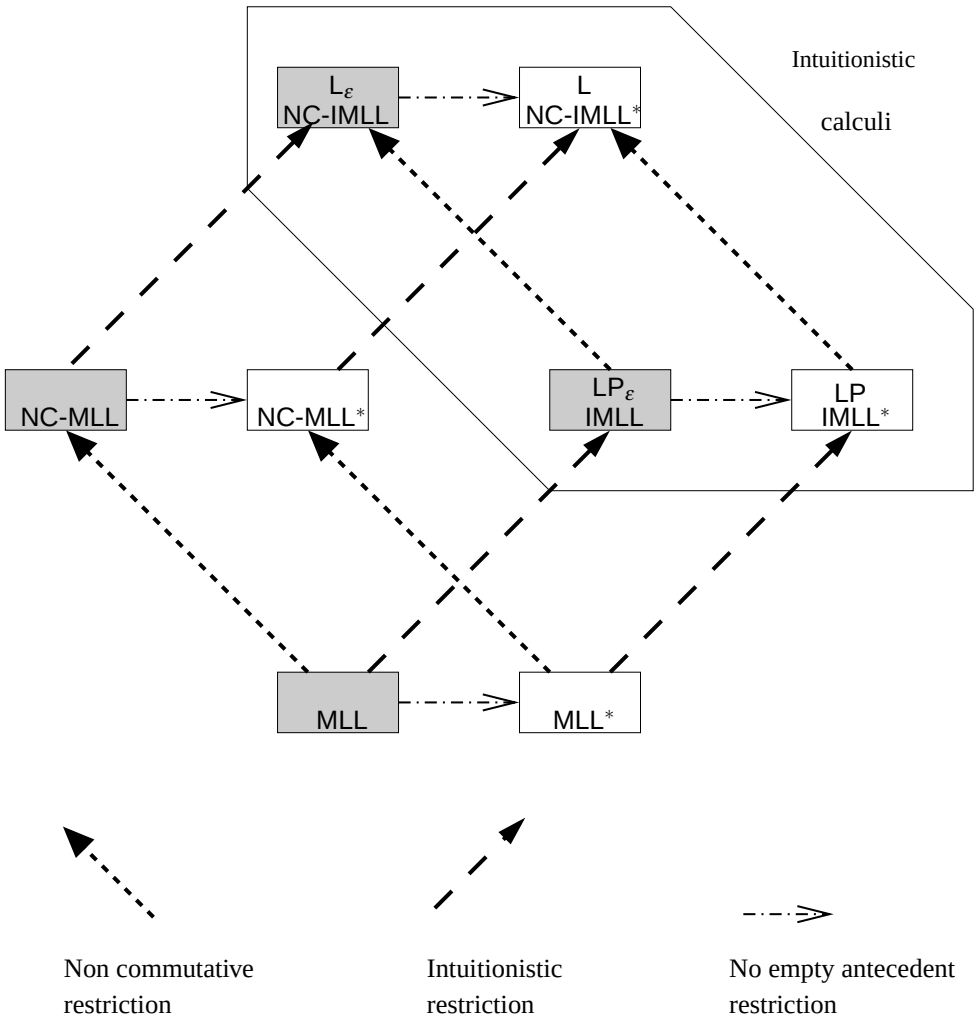
in one sided presentation: at least two formulae in every sequent

The names for these calculi somehow differ in the categorial tradition and in the linear logic community, for instance classical calculi are never considered in the categorial tradition, and calculus without empty antecedent are neverconsidered in linear logic. For linear calculi, the restriction corresponding to forbid empty antecedents will be denoted by $(\cdots)^*$. Conversely, for categorial grammar and Lambek calculus, allowing for empty antecedents will be denoted by $(\cdots)^\varepsilon$. The non commutative restriction of a linear calculus restriction will be denoted by a prefix NC, and the commutative extension of a Lambek style calculus will be denoted by a suffix P

Because of these two communities, we have two names for the intuitionistic calculi, and we hope it will not confuse the reader. To help him, here is a commutative diagram of these restriction. All these restrictions will appear again for describing the proof-nets corresponding to each calculus.

Although this might be surprising we are able to provide a one sided formulation for intuitionistic calculi. So we will use the *linear name* $\cdots$*MLL for one sided caculi* and the *categorial name L*$\cdots$ *for two sided calculi*.

INTUI	NC	ε-FREE	Linear name	Categorial name
yes	yes	yes	NC-IMLL*	L
yes	yes	no	NC-IMLL	L_ε
yes	no	yes	IMLL*	LP
yes	no	no	IMLL	LP_ε
no	yes	yes	NC-MLL*	
no	yes	no	NC-MLL	
no	no	yes	MLL*	
no	no	no	MLL	
			one sided	two sided



3.3.2 The intuitionistic restriction in one sided calculi

The two sided intuitionistic calculus LP_ε is a proper restriction of its classical counterpart MLL. For instance letting $F = (\beta \parr \alpha) \parr (\alpha^\perp \otimes \beta^\perp)$ one has $\mathsf{MLL} \vdash (\vdash F)$ but there is no formula G equivalent to F such that $\mathsf{LP}_\varepsilon \vdash (\vdash G)$. Actually, this restriction is only a *language restriction*, that we already studied in paragraph 3.1.3. Indeed, it is only because there is no formula in Lp equivalent to F, i.e. because $F \notin \mathsf{Li}^\bullet \cup \mathsf{Li}^\circ$ that F is not a theorem of IMLL. More precisely we have the following result:

The fact that intuitionistic restriction for two sided calculi does correspond to the possibility to formulate a proof of an equivalent sequent in a usual two sided intuitionistic calculi (with one conclusion and with only the connectives $\backslash, /, \bullet$) relies on the following property, first studied by by van de Wiele in the typed case [70, taken up again] and by Danos and Regnier [71,72] in the untyped case. This property has lead Lamarche to interesting theory of intuitionistic proof-nets [73] somehow orthogonal to our presentation.

Proposition 44 *If* $\forall i \in [1,n]\ A_i \in \mathsf{Li}^\bullet \cup \mathsf{Li}^\circ$ *then*

$$\mathsf{MLL} \vdash (\vdash A_1, \ldots, A_n) \ \Leftrightarrow\ \mathsf{IMLL} \vdash (\vdash A_1, \ldots, A_n)$$

and whenever these properties hold, then exactly one formula of the sequent is in Li°*, all others being in* $\mathsf{Li}^\bullet$*. This also holds for the variants NC-MLL et NC-IMLL.*

PROOF : Easy induction on the proofs. ◇

From the previous proposition we easily deduce the correspondence between one sided intuitionistic calculi and the two sided intuitionistic calculi:

Proposition 45 *If* $\mathsf{MLL} \vdash (\vdash F_1, \ldots, F_n)$*, with* $\forall i \in [1,n]\ F_i \in \mathsf{Li}^\bullet \cup \mathsf{Li}^\circ$*, then :*

- *there exists a unique index* $i_0 \in [1,n]$ *such that* $F_{i_0} \in \mathsf{Li}^\circ$ *and for every other index* $i \in [1,n]$ *we have* $F_i \in \mathsf{Li}^\bullet$ *because of the proposition 44*
- *because of paragraph 3.1.3, every formula* $F_i^\perp$ *with* $i \neq i_0$ *is equivalent to unique formula* $(F_i)^\bullet_{\mathsf{Lp}} \in \mathsf{Lp}$*, while* F_{i_0} *is equivalent to a unique formula* $(F_{i_0})^\circ_{\mathsf{Lp}}$
- $\mathsf{LP}_\varepsilon \vdash \left((F_{i_0-1})^\bullet_{\mathsf{Lp}}, (F_{i_0-2})^\bullet_{\mathsf{Lp}}, \ldots, (F_1)^\bullet_{\mathsf{Lp}}, (F_n)^\bullet_{\mathsf{Lp}}, \ldots, (F_{i_0+1})^\bullet_L \vdash (F_{i_0})^\circ_{\mathsf{Lp}}\right)$

[70] G. Bellin and P. J. Scott. On the π-calculus and linear logic. *Theoretical Computer Science*, 135:11–65, December 1994.

[71] Vincent Danos. *La logique linéaire appliquée à l'étude de divers processus de normalisation et principalement du λ-calcul*. Thèse de Doctorat, spécialité Mathématiques, Université Paris 7, juin 1990.

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

[73] François Lamarche. Proof nets for intuitionistic linear logic: Essential nets. 35 page technical report available by FTP from the Imperial College archives, April 1994.

Conversely, $LP_\varepsilon \vdash (X_1,\ldots,X_n \vdash Y) \;\Rightarrow\; MLL, IMLL \vdash (\vdash -X_n,\ldots,-X_1,+Y)$.

If one replaces MLL with NC-MLL (resp. NC-MLL*) and LP_ε with L_ε (resp. L) the result also holds. (As announced in the commutative diagram, the restriction INTUI,NC *and* ε-FREE *commute).*

For these non commutative variants NC-MLL, NC-MLL*, L_ε et L, with a restricted exchange rule, one has to abide by the order between formulae: this order is reversed when formulae move form one hand-side of the sequent to the other.

PROOF : The "conversely" is obvious.

The direct implication is shown by induction on the proof. For the proof to work in the non commutative case, the rule (tx) is only used for the translation of the $(\mathbf{x})_h$ rule of IMLL. Here is, for instance, the translation of the $/_h$.

Assume that the sequences of formulae involved in $/_h$ are $\Gamma = G_1,\ldots,G_n$, $\Gamma' = G'_1,\ldots,G'_k$, $\Delta' = D'_1,\ldots,D'_l$. Here is the NC-MLL proof which simulates the rule $/_h$ of L_ε — remember that $+A\otimes -B = -(A/B)$ (c.f. paragraph 3.1.3) :

$$\dfrac{\dfrac{\vdash -G_n,\ldots,-G_1,+A \qquad \dfrac{\vdash -D'_l,\ldots,-D'_1,-B,-G'_k,\ldots,-G'_1,+C'}{\vdash -B,-G'_k,\ldots,-G'_1,+C',-D'_l,\ldots,-D'_1}\,l(EC)}{\vdash -G_n,\ldots,-G_1,+A\otimes -B,-G'_k,\ldots,-G'_1,+C',-D'_l,\ldots,-D'_1}\,\otimes}{\vdash -D'_l,\ldots,-D'_1,-G_n,\ldots,-G_1,+A\otimes -B,-G'_k,\ldots,-G'_1,+C'}\,l(EC)}{\vdash -D'_l,\ldots,-D'_1,-G_n,\ldots,-G_1,-(A/B),-G'_k,\ldots,-G'_1,+C'}\,i.e.$$

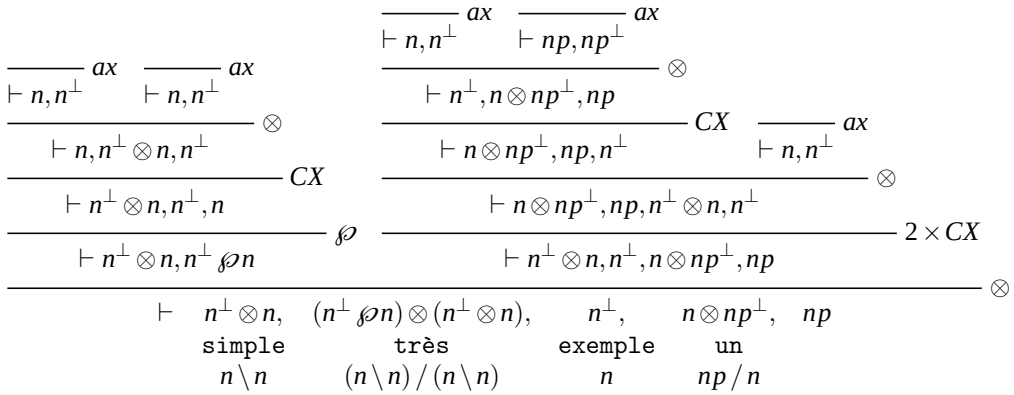
$\diamond$

Let us provide the NC-MLL translation of the proofs or parse structures given in examples 41 and 42 :

EXAMPLE **46**

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash n,n^\perp}\,ax \quad \dfrac{}{\vdash n,n^\perp}\,ax}{\vdash n,n^\perp\otimes n,n^\perp}\,\otimes}{\vdash n^\perp\otimes n,n^\perp,n}\,CX}{\vdash n^\perp\otimes n,n^\perp \parr n}\,\parr \qquad \dfrac{\dfrac{\dfrac{}{\vdash n,n^\perp}\,ax \quad \dfrac{}{\vdash n,n^\perp}\,ax}{\vdash n,n^\perp\otimes n,n^\perp}\,\otimes}{\vdash n^\perp\otimes n,n^\perp,n}\,CX}{\vdash n^\perp\otimes n,(n^\perp \parr n)\otimes(n^\perp\otimes n),n^\perp,n}\,\otimes \qquad \dfrac{}{\vdash np^\perp,np}\,ax}{\vdash \underset{\substack{\texttt{simple}\\ n\backslash n}}{n^\perp\otimes n},\ \underset{\substack{\texttt{très}\\ (n\backslash n)/(n\backslash n)}}{(n^\perp\parr n)\otimes(n^\perp\otimes n)},\ \underset{\substack{\texttt{exemple}\\ n}}{n^\perp},\ \underset{\substack{\texttt{un}\\ np/n}}{n\otimes np^\perp},\ np}\,\otimes$$

EXAMPLE **47**



3.4 Proof-nets : concise and expressive proofs

We now turn our attention to proof-nets; they are for linear logic what natural deductions (or typed lambda terms) are for intuitionistic logic, in the sense that the contexts are not copied at each step of the proof.

From a logical viewpoint they are much more compact than sequent calculus proofs, they well-formedness is a global condition but quick to check, and cut-elimination is a local and efficient process. But their main advantage, is a better representation of proofs. Indeed, many sequent calculus proofs which only differ in the order of application of the rules convert into the same proof-net. For example, the two proofs given in the examples 46 and 47 will yield the same proof-net. It should be notice that when these proofs are viewed as parse structures (they correspond to parse structures 41 and 42. in a Lambek grammar), they both describe the same linguistic analysis, so it is really a good outcome of proof-nets that we are able to describe this analysis by a single object.

3.4.1 Proof-nets for MLL

3.4.1.1 R&B Graphs

A *matching* in a graph is a subset of the set of edges such that no two edges of the matching are adjacent. The matching is said to be *perfect* whenever each vertex of the graph is incident to an edge of the matching – because it is a matching, each vertex is incident to exactly one edge of the matching.

Definition 48 (R&B graphs) *A R&B graph is a edge colored graph, whose edges either are of colour B (blue or bold), or R(red or regular), such that the B edges defines a perfect matching of the graph.*

B edges correspond to formulae and R edges to connectives. The recognition, among all such graphs, of the ones which are proofs, will involved the notion of alternate elementary path.

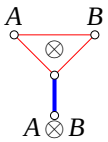
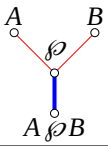
Definition 49 (æ paths and cycles) *An æ path in a R&B graph is an alternating elementary path, that is a path the edges of which are alternatively in B and in R which does not use twice the same edge — as B edges are a matching, this is equivalent to the path does not contain twice the same vertex (except, possibly the first and last vertices that might be the same). More precisely, an æ path is a finite sequence of edges* $(a_i)_{i\in[1,n]}$ *such that :*

$$i \neq j \Longrightarrow a_i \neq a_j \qquad \#(a_i \cap a_{i+1}) = 1$$
$$a_i \in B \Longrightarrow a_{i+1} \in R \qquad a_i \in R \Longrightarrow a_{i+1} \in B$$

An æ cycle is an æ path of even length, whose end vertices are equal.

3.4.1.2 Prenets

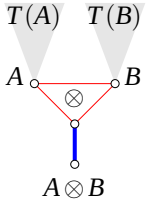
Definition 50 (Prenets or proof structures, links) *Prenets are R&B graphs built from basic R&B graphs called links:*

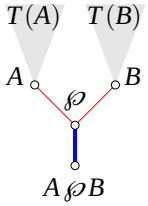
Liens			
Name	*Graph*	*Premises*	*Conclusions*
Axiom		*none*	α *et* $\alpha^\perp$
Times		A *et* B	$A\otimes B$
Par		A *et* B	$A\wp B$

in such a way that each formula is the conclusion of exactly one link and the premise of at most one link. Formulae that are not the premise of a link are called conclusions of the prenet.

Definition 51 (R&B subformula tree) *Given a formula C, its* R&B subformula tree $T(C)$ *is a R&B graph defined inductively as follows:*

- *If* $C = \alpha$ *is a propositional variable then* $T(C)$ *is :* 
- $T(A\otimes B)$ *et* $T(A\wp B)$ *are définis à partir de* $T(A)$ *et* $T(B)$ *ainsi :*

$T(A\otimes B)$: 

$T(A\wp B)$: 

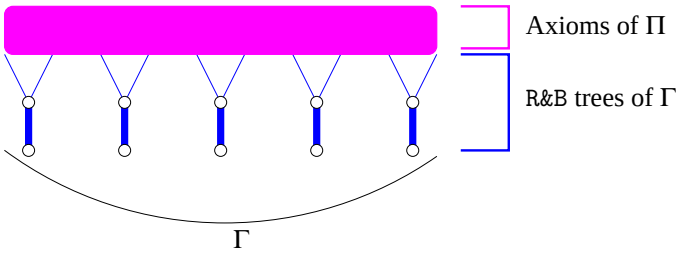
Beware that the R&B subformula tree of a formula C is not, from a graph theoretical point of view, a tree: indeed, every *Times* link contains a cycle. We nevertheless chose this name because it is very similar to the subformula tree, and that w.r.t. the æ paths, the only paths we shall use, the R&B subformulae trees are acyclic.

The vertices corresponding to propositional variables in a subformula tree will be called leaves of the subformula tree.

Definition 52 (prenet with conclusions Γ) *Given a sequence of formulae* Γ*, a prenet with conclusions* Γ *consists in:*

- *the R&B subformula trees of the formulae in* Γ
- *a set of non incident B edges joining dual leaves, called axioms, such that each leave is incident to exactly one axiom.*

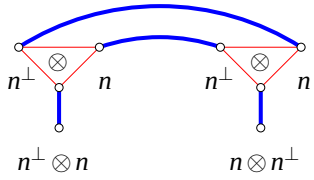
The structure of a prenet is the following:



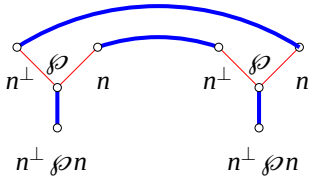
Notice that the order between formulae of Γ or their subformula trees is not part of the structure, but because of the labeling of the vertices, R&B subformula trees make a distinction between their right and left subtrees.

Here are some examples of prenets:

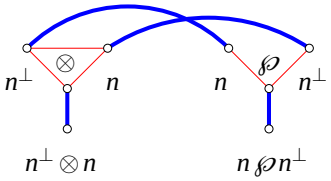
EXAMPLE **53**



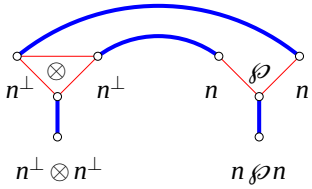
EXAMPLE **54**



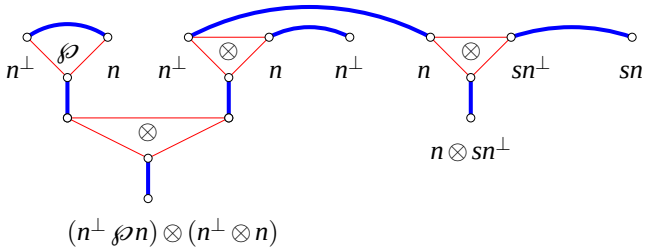
EXAMPLE **55**



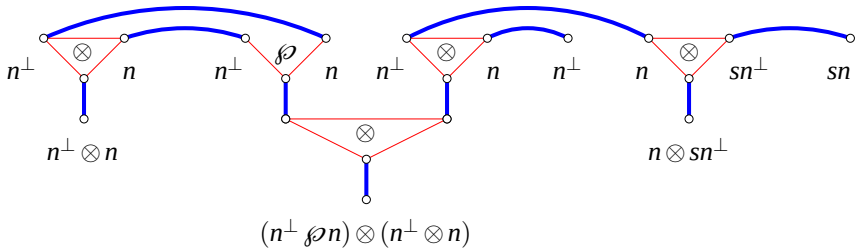
EXAMPLE **56**



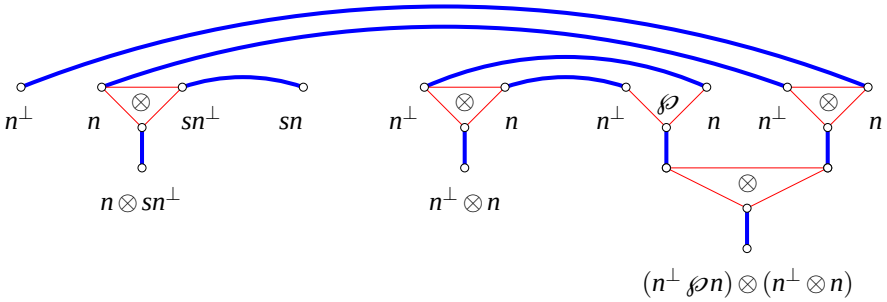
EXAMPLE **57**



EXAMPLE **58**



EXAMPLE **59**



3.4.1.3 Proof-Nets

Definition 60 (proof-net) *A proof-net is a prenet satisfying the following properties:*

ØÆ : *there is no æ cycle.*

SAT : *there exists an æ path between any two vertices.*

Regarding older presentation of proof-nets, like [74,16], where one consider all correction graphs obtained by suppressing one of every pair of *Par* edges in the sub-formula trees, the first property correspond to the acyclicity of all these graphs and the second property to their connectedness. [75,76].

The following result of [76,77] shows that the correctness of proof-nets is not too bad from an algorithmic point of view — recently some linear algorithms have been provided on some other presentation of proof-nets, and they certainly can be adapted to our formalism [78,79].

Proposition 61 *Given a prenet with n vertices, their exist an algorithm working in n^2 steps which decides whether the prenet is a proof-net.*

Among the examples of prenets given above, only 55, 56, 57, 58 and 59 are proof-nets. The prenet 53 contains an æ cycle, and the prenet 54 does not contain any æ path between the left most leaves $n^{\perp}$ and n.

3.4.2 Sequent calculus and proof-nets

The following proposition gives a precise account of the correspondence between proof-nets and sequent calculus proofs, and its proofs shows how sequent calculus proofs are mapped onto proof-nets. The converse correspondence relies on graph theoretical properties, and we refer the reader to [76,77].

Theorem 62 *Every sequent calculus proof in MLL of a sequent $\vdash A_1,\ldots,A_n$ translates into a proof-net with conclusions $A_1,\ldots,A_n$. Conversely, every proof-net with conclusions $A_1,\ldots,A_n$ corresponds to at least one sequent calculus proof in MLL of $\vdash A_1,\ldots,A_n$ in NC-MLL — every such proof is called a sequentialisation of the proof-net.*

[74] Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.

[16] Jean-Yves Girard. Linear logic: its syntax and semantics. In Girard et al. [2], pages 1–42.

[2] Jean-Yves Girard, Yves Lafont, and Laurent Regnier, editors. *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Notes*. Cambridge University Press, 1995.

[75] Arnaud Fleury and Christian Retoré. The mix rule. *Mathematical Structures in Computer Science*, 4(2):273–285, 1994.

[76] Christian Retoré. Perfect matchings and series-parallel graphs: multiplicative proof nets as R&B-graphs. In J.-Y. Girard, M. Okada, and A. Scedrov, editors, *Linear'96*, volume 3 of *Electronic Notes in Theoretical Science*. Elsevier, 1996. http://www.elsevier.nl/.

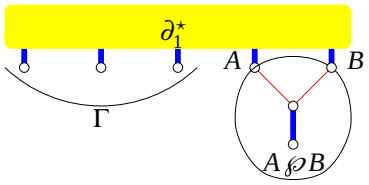
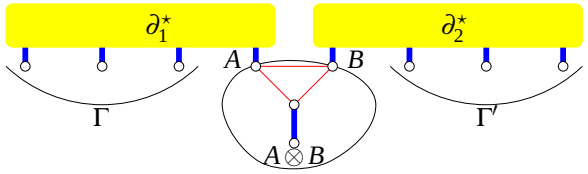
[77] Christian Retoré. Handsome proof-nets: perfect matchings and cographs. *Theoretical Computer Science*, 294(3):473–488, 2003. Complete version RR-3652 http://www.inria.fr.

[78] S. Guerrini. Correctness of multiplicative proof nets is linear. In *14th Symposium on Logic in Computer Science (LICS'99)*, pages 454–463. IEEE, July 1999.

[79] A. Murawski and C.-H. Ong. Dominator trees and fast verification of proof nets. In *15th Symposium on Logic in Computer Science (LICS' 00)*, pages 181–191. IEEE, June 2000.

PROOF : As said above, we limit ourselves to the first part of this statement.

The translation from sequent calculus proofs to proof-nets is defined inductively. As the exchange rule has no effect on proof-nets, since for the time being we have no order on the conclusions, we simply skip it. The effect of this rule would be to produce crossings of axiom links, but up to now this is not part of our description of a proof-net. For instance the examples, 58 and 59 are considered as the same proof-net.

Proof ∂ in MLL	Corresponding proof-net $\partial^\star$
$\dfrac{}{\vdash \alpha^\perp, \alpha}\, ax$	
$\dfrac{\begin{array}{c}\vdots\,\partial_1 \\ \vdash \Gamma, A, B\end{array}}{\vdash \Gamma, A \parr B}\, \parr$	
$\dfrac{\begin{array}{cc}\vdots\,\partial_1 & \vdots\,\partial_2 \\ \vdash \Gamma, A & \vdash B, \Gamma'\end{array}}{\vdash \Gamma, A \otimes B, \Gamma'}\, \otimes$	

It is easily checked by induction that the prenet corresponding to a sequent calculus proof are proof-nets : no æ cycle can appear during the construction, and the fact that there always exists an æ path between any two vertices is also preserved during the construction. ◇

Using this inductive definition, the proofs 46 and 47, both yield to the proof-net 58, so a single proof-net correspond to a single parse structure.

Rules and links are in a one-to-one correspondence (ax/*Axiome*, $\parr$/*Par*, $\otimes$/*Tenseur*), and the last logical rule in the sequent calculus proof correspond to a final link in the prenet — a link which is the root of one of the subformula trees — while the converse does not hold. We nevertheless have the following property, that will be useful later on :

Proposition 63 *In a proof-net the conclusions of which are all Times or Axioms, and which is not reduced to a single Axiom at least one of the final Times links is* splitting

that is each of the two premise B edges is a bridge — an edge the suppression of which increases the number of connected components.

PROOF : As we have a proof-net, at least one sequent calculus proof translates into it. The final rule of the sequent calculus correspond to a final link, so is a *Times* link. From the translation given above, both the premise B edges of this link are bridges of the graph. ◇

Observe that not all final *Times* links are splitting. For instance in the example 58 the final *Times* $n^\perp \otimes n$ is not splitting, and never can be the translation of the final rule of a corresponding sequent calculus proof. The final *Times* links $(n^\perp \parr n) \otimes (n^\perp \otimes n)$ and $n \otimes np^\perp$ are splitting *Times* links, and this is supported by the sequentialisations given in examples 46 and 47.

A minimal representation of prenets and proof-nets To define a prenet or a proof-net Π it is enough to give its conclusions and the pairs of propositional variables which are linked by an axiom link. These pairs can be depicted by a 2-permutation σ_Π — that is a permutation such that $\sigma_\Pi^2 = Id$ and $\forall x\ \sigma_\Pi(x) \neq x$ — defined on the set of occurrences of atoms in the sequence of conclusions. This representation will become necessary when we will deal with proof-nets for the Lambek calculus, that are parse structure for Lambek categorial grammars.

Up to now, representing the conclusions by a graph is needed to check whether a prenet is a proof-net [48,74,80,81,82]. This graph can be minimized in more abstract representation [77]. There exists an alternative criterion relying on denotational semantics [83] which does not need such a graph, but, unfortunately, the correctness checking the correctness becomes exponential.

Let us give the description of the examples 58 and 55 by means of 2-permutations.

EXAMPLE **64**

[48] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[74] Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.

[80] Andrea Asperti. A linguistic approach to dead-lock. Technical Report LIENS 91-15, Dép. Maths et Info, Ecole Normale Supérieure, Paris, 1991.

[81] Andrea Asperti and Giovanna Dore. Yet another correctness criterion for multiplicative linear logic with mix. In A. Nerode and Yu. Matiyasevich, editors, *Logical Fundations of Computer Science*, volume 813 of *Lecture Notes in Computer Science*, pages 34–46, St. Petersburg, 1994. Springer Verlag.

[82] François Métayer. Homology of proof-nets. Prépublication 39, Equipe de Logique , Université Paris 7, 1993.

[77] Christian Retoré. Handsome proof-nets: perfect matchings and cographs. *Theoretical Computer Science*, 294(3):473–488, 2003. Complete version RR-3652 http://www.inria.fr.

[83] Christian Retoré. A semantic characterisation of the correctness of a proof net. *Mathematical Structures in Computer Science*, 7(5):445–452, 1997.

Proof-Net Π	Example 58										Example 55			
Conclusions of Π	$n^\perp \otimes n$		$(n^\perp \parr n) \otimes (n^\perp \otimes n)$				$n^\perp$	$n \otimes np^\perp$		np	$n^\perp \otimes n$		$n \parr n^\perp$	
Atom occurrences x	$n_1^\perp$	n_2	$n_3^\perp$	n_4	$n_5^\perp$	n_6	$n_7^\perp$	n_8	$np_9^\perp$	np_{10}	$n_1^\perp$	n_2	n_3	$n_4^\perp$
$\sigma_\Pi(x)$	n_4	$n_3^\perp$	n_2	$n_1^\perp$	n_8	$n_7^\perp$	n_6	$n_5^\perp$	np_{10}	$np_9^\perp$	n_3	$n_4^\perp$	$n_1^\perp$	n_2

3.4.3 Intuitionistic proof-nets

Definition 65 *An intuitionistic proof-net with conclusions $F_1, \ldots, F_n$ is a proof-net satisfying :*

INTUI : $\forall i \in [1,n]\ F_i \in \mathsf{Li}^\circ \cup \mathsf{Li}^\bullet$.

For instance the example 56 is not an intuitionistic proof-net since $n \parr n \notin \mathsf{Li}^\bullet \cup \mathsf{Li}^\circ$.

Theorem 66 *Every sequent calculus proof $A_1, \ldots, A_n \vdash B$ in IMLL translates into an intuitionistic proof-net with conclusions $-A_n, \ldots, -A_1, +B$.*

Conversely, let Π a proof-net with conclusions $F_1, \ldots, F_n \in \mathsf{Li}$. Then there exists a unique index i_0 in $[1,n]$ such that $F_{i_0} \in \mathsf{Li}^\circ$ and $F_i \in \mathsf{Li}^\bullet$, for $i \neq i_0$, and Π is the translation of a proof in IMLL of

$$(F_{i_0-1})^\bullet_{\mathsf{Lp}}, (F_{i_0-2})^\bullet_{\mathsf{Lp}}, \ldots, (F_1)^\bullet_{\mathsf{Lp}}, (F_n)^\bullet_{\mathsf{Lp}}, \ldots, (F_{i_0+1})^\bullet_{\mathsf{Lp}} \vdash (X_{i_0})^\circ_{\mathsf{Lp}}$$

PROOF : The first part is obvious.

For the converse, we first have to justify the existence of i_0. This existence is justified by theorem 62 (it shows that Π is the translation of proof of MLL) and proposition 44 (which shows that a proof in MLL with all its conclusions in $\mathsf{Li}^\bullet \cup \mathsf{Li}^\circ$ has exactly one conclusion in in Li° and all the others in $\mathsf{Li}^\bullet$). Once the existence of i_0 is established, the result follows from proposition 45, which shows that given a sequentialisation of Π in MLL, with conclusions $\vdash F_1, \ldots, F_n$ (with F_{i_0} in Li° and all the others in $\mathsf{Li}^\bullet$) corresponds to a proof in IMLL of

$$(F_{i_0-1})^\bullet_{\mathsf{Lp}}, (F_{i_0-2})^\bullet_{\mathsf{Lp}}, \ldots, (F_1)^\bullet_{\mathsf{Lp}}, (F_n)^\bullet_{\mathsf{Lp}}, \ldots, (F_{i_0+1})^\bullet_{\mathsf{Lp}} \vdash (X_{i_0})^\circ_{\mathsf{Lp}}$$

$\diamond$

3.4.4 Cyclic proof-nets

We now turn our attention towards proof-nets for NC-MLL. These are proof-nets which can be drawn in the plane without intersecting axioms, keeping the same design and up-down orientation for links. This condition is strictly stronger than being a planar graph (because we ask for the links to be as we draw them). Consequently we shall present this condition without any reference to an embedding of the graph in the plane, but by means of 2-permutation (brackettings from formal language theory would work just the same). This restriction, combined with the previous one for intuitionistic proof-nets from the previous paragraph, will allow for a characterisation of proof-nets for the Lambek calculus, and therefore to directly parse phrases and sentences with proof-nets.

3.4.4.1 Cyclic permutations and compatibility of a 2-permutation

A permutation ψ over a set E is said to be cyclic whenever:

$$\forall x,y \in E \quad \exists k \in [0,n-1] \quad y = \psi^k(x) \text{ (with } \psi^0(x) = x)$$

such a permutation ψ can be described by an expression :

$$\rhd\, x; \psi(x); \psi(\psi(x)); \cdots; \psi^{n-1}(x) \,\rhd$$

Given $x,y \in E$, and an index $k \in [0,n-1]$ such that $y = \psi^k(x)$, we write $[x,y]$ for the set $\{z/\exists j \in [0,k] \quad z = \psi^j(x)\}$; similarly $[x,y[$ is defined as $\{z/\exists j \in [0,k[\quad z = \psi^j(x)\}$ etc.

Given a set E endowed with a cyclic permutation ψ and a 2-permutation σ we can give an algebraic account of the following geometric fact: if we place the points of E on a circle following the cyclic order ψ the chords joining x and $\sigma(x)$ *do not intersect* — in other words, σ is a correct bracketing, w.r.t. the cyclic order ψ over E.

Definition 67 *A 2-permutation σ of E is said to be* compatible *with a cyclic permutation ψ of E whenever* $\forall x,y \in E\ x \in [y,\sigma(y)] \Rightarrow \sigma(x) \in [y,\sigma(y)]$.

For instance the 2-permutation σ_Π of the example 64 $(n_1^\perp, n_3), (n_2, n_4^\perp)$ is not compatible with the cyclic permutation $\rhd\, n_1^\perp; n_2; n_3; n_4^\perp \,\rhd$. Indeed, $n_2 \in [n_1^\perp, \sigma_\Pi(n_1^\perp){=}n_3]$ while $\sigma_\Pi(n_2) = n_4^\perp \notin [n_1^\perp, n_3]$.

In the following definition the E_i's should be viewed as the conclusions of a proof-net Π, endowed with the cyclic permutation Ψ_Π. The induced cyclic permutation is the cyclic permutation induced on the atoms — thus viewing σ of the previous definition as the axioms of Π we are able to express that axioms do not intersect.

Definition 68 *Let $\rhd E_1; \cdots; E_n \,\rhd$ be a cyclic permutation of $M = \{E_1, \ldots, E_n\}$ where each E_i is a sequence of symbols $a_i^1, a_i^2, \ldots, a_i^{p_i}$. The cyclic permutation induced by Ψ over the disjoint sum of the symbols of the E_i's is the cyclic permutation defined by :*

$$\rhd\, a_1^1; a_1^2; \cdots; a_1^{j_1}; a_2^1; a_2^2; \cdots; a_2^{j_2}; \cdots; a_n^1; a_n^2; \cdots; a_n^{j_n} \,\rhd$$

In order to characterize the proof-nets for the Lambek calculus we shall need the following proposition:

Proposition 69 *Let Ψ be a cyclic permutation over a finite set M of n sequences of symbols $M = E_1, \ldots, E_n$. Let ψ be the cyclic permutation induced on $E = \oplus E_i$, as in definition 68. Let σ be a 2-permutation of E, compatible with ψ, as in definition 67. Let Σ be the following (symmetric) relation over M: $E_i \Sigma E_j$ whenever there exists $x_i \in E_i$ such that $\sigma(x_i) \in E_j$. Let Σ^* be the transitive closure of Σ; if Σ^* has* exactly two equivalence classes *$\mathscr{G}$ et $\mathscr{D}$, then there exist $G \in \mathscr{G}$ and $D \in \mathscr{D}$ such that : $\mathscr{G} = [G, D[$ et $\mathscr{D} = [D, G[$.*

PROOF : By induction on $\#E + n$.

If one of the class contains only one element, the result is obvious — this necessarily happens when a class has a single element, for instance when $n = 2$.

There exists z such that $\psi(z) = \sigma(z)$ Let z be a point such that $\#]z, \sigma(z)[$ has the smallest number of elements, and let us show that $\#]z, \sigma(z)[= 0$ — hence $\psi(z) = \sigma(z)$. Assume that there exists $y \in]z, \sigma(z)[$; since σ is compatible with ψ, $\sigma(y) \in]z, \sigma(z)[$. Thus one of the two intervals $]y, \sigma(y)[$ or $]\sigma(y), \sigma(\sigma(y)) = y[$ is a subset of $]z, \sigma(z)[$, and since none of them contains y, they have strictly less elements than $\#]z, \sigma(z)[$, contradiction.

Let z be an element such that $\psi(z) = \sigma(z)$ and let i be the index such that $z \in E_i$. Three cases can happen:

$\sigma(z) \in E_i$ et $E_i = z, \sigma(z)$ In this case, E_i is the only element in its equivalence class, and the result is clear.

$\sigma(z) \in E_i$ et $E_i = \ldots, z, \sigma(z), \ldots$ In this case, replace E_i with $E_i \setminus \{z, \sigma(z)\}$, restrict σ and ψ to $E \setminus \{z, \sigma(z)\}$. The induction hypothesis apply, and since Σ^* remains unchanged, the D and G provided by the induction hypothesis are solutions for the original problem.

$\sigma(z) \notin E_i$. In this case $\sigma(z)$ is the first symbol of $E_{i+1} = \Psi(E_i)$. Let us consider the following reduce problem:

let Ψ' be the cyclic permutation $\rhd E_1; \ldots; E_{i-1}; E_{i(i+1)}; E_{i+2}; \ldots; E_n \rhd$ where $E_{i(i+1)}$ is the sequence of symbols E_i, E_j

Observe that E, ψ and σ remains unchanged, and therefore σ is compatible with ψ. Since $E_i \Sigma E_{i+1}$ the equivalence relation Σ'^* for this reduced problem also has exactly two classes.

Hence we are faced to a similar problem with $\#M' = n - 1$. The induction hypothesis yields G' and D' such that $\mathscr{G}' = [G', D'[$ and $\mathscr{D}' = [D, G'[$. A solution to the original problem is given by $G = G'$ and $D = D'$ —- if G' (resp. D') is $E_{i(i+1)}$, then G (resp. D) should be E_i. ◇

3.4.4.2 Cyclic proof-nets

Definition 70 *A cyclic prenet with conclusions* $\Psi :\rhd A_1;\cdots;A_n \rhd$ *is a prenet with conclusions* $A_1,\ldots,A_n$ *endowed with the cyclic permutation* $\Psi_\Pi :\rhd A_1,\ldots,A_n \rhd$*. We write* Ψ^{at}_Π *for the cyclic permutation induced by* Ψ_Π *on the atoms of* Ψ *— in the sense of the definition 68.*

Definition 71 *A cyclic prenet with conclusion* $\Psi :\rhd A_1,\cdots,A_n \rhd$ *is a cyclic proof-net if and only if it is a proof-net with conclusion* $A_1,\ldots,A_n$ *(the conditions* ØÆ *and* SAT*are satisfied) and:*

NC : σ_Π *is compatible with* Ψ^{at}_Π

For instance the example 55 is not a cyclic proof-net. Indeed, $\Psi_\Pi = \rhd n_1^\perp \otimes n_2; n_3 \parr n_4^\perp \rhd$ (there are only two conclusions, so there is only one possible cyclic permutation), and $\Psi^{at}_\Pi = \rhd n_1^\perp; n_2; n_3; n_4^\perp \rhd$, while the 2-permutation σ_Π of its axiom links, given in example 64, is not compatible with Ψ^{at}_Π — as we have seen after the definition 67.

The proof-nets of the examples 56, 57, 58 and 59 are cyclic proof-nets.

Theorem 72 *Every sequent calculus proof of* $\vdash A_1,\ldots,A_n$ *in NC-MLL translates into a cyclic proof-net with conclusions* $\rhd A_1;\cdots;A_n \rhd$*.*

Conversely, every cyclic proof-net with conclusion $n \rhd A_1;\cdots;A_n \rhd$ *is the translation of at least a sequent calculus proof of* $\vdash A_1,\ldots,A_n$ *in NC-MLL.*

PROOF : The first part is rather simple to establish by induction on the sequent calculus proof. Nevertheless one should take care of the compatibility of Ψ^{at}_Π with σ_Π; to do so, one should place atoms on a circle, and draw axiom links as chords of this circle, and draw R&B subformula trees outside the circle. Observe that the cyclic exchange (cx) corresponds to the equality of the proof-nets.

The converse is proved by induction on the number of links of the proof-net Π. As it is a proof-net, the proposition 63 applies.

If Π is an axiom $\rhd \alpha, \alpha^\perp \rhd = \rhd \alpha, \alpha^\perp \rhd$ a sequentialisation is provided by the axiom $\vdash \alpha, \alpha^\perp$ of NC-MLL.

If Π has a final *Par* link $A_i = A \parr A'$, let us consider Π' the proof-net obtained from Π by suppressing this final *Par* link and endowed with the cyclic permutation $\rhd A_1;\ldots;A_{i-1};A;A';A_{i+1};\cdots A_n \rhd$. The proof-net Π' is a cyclic proof-net as well, since $\Psi^{at}_{\Pi'} = \Psi^{at}_\Pi$ and $\sigma_{\Pi'} = \sigma_\Pi$. By induction hypothesis there exists a sequent calculus proof in NC-MLL corresponding to Π', and applying a $\parr$ rule to this proof yields a sequentialisation of Π.

Otherwise, Π has a splitting *Times*, say $A_i = A \otimes A'$. Suppressing this final link yields two proof-nets Π_A and $\Pi_{A'}$ with conclusions $\Gamma_A = A_{i_1},\ldots,A_{i_p},A$ and $\Gamma_{A'} = A_{j_1},\ldots,A_{j_q},A$ with $\{i_1,\ldots,i_p,j_1,\ldots,j_q\} = [1,n] \setminus \{i\}$. Consider the prenet $\Pi' = \Pi_A \cup \Pi_{A'}$ and endow its conclusions with the cyclic permutation

$\triangleright A_1;\cdots;A_{i-1};A;A';A_{i+1};\cdots;A_n \triangleright$. Since $\Psi^{at}_{\Pi'} = \Psi^{at}_{\Pi}$ and $\sigma_{\Pi'} = \sigma_{\Pi}$, the 2-permutation $\sigma_{\Pi'}$ is compatible with $\Psi^{at}_{\Pi'}$. Let Σ be the (symmetric) relation between the conclusions of Π' defined by: $\exists x \in C\ \sigma_{\Pi}(x) \in C'$ — in other words, this relation holds whenever Π contains an axiom with a conclusion in C and the other in C'. The link $A \otimes B$ is splitting in Π, means that Σ^* has exactly two equivalence classes Γ_A and $\Gamma_{A'}$. Because of the proposition 69 the cyclic permutation of the conclusions of Π' can be written as $\triangleright A_{i_1};\cdots,A_{i_p};A;A',A_{j_1};\cdots;A_{j_q} \triangleright$. Thus Π_A (resp. $\Pi_{A'}$) endowed with the cyclic permutation $\triangleright A_{i_1};\cdots,A_{i_p};A \triangleright$ (resp. $\triangleright A',A_{j_1};\cdots;A_{j_q} \triangleright$) is a cyclic proof-net. Indeed Π_A is a proof-net and since σ_{Π_A} and $\Psi^{at}_{\Pi_A}$ are the restrictions to Γ_A of σ_{Π} and Ψ^{at}_{Π} compatibility is preserved — the same argument works for $\Pi_{A'}$.

Therefore, by induction hypothesis we have two sequent calculus proofs in NC-MLL with conclusions $\vdash A_{i_1};\cdots,A_{i_p};A$ and $\vdash A';A_{j_1};\cdots;A_{j_q}$ corresponding to Π_A and Π'_A. Applying the rule $\otimes$ of NC-MLL yields a proof with conclusion $\vdash \Gamma_A, A \otimes B, \Gamma_B$ corresponding to Π. $\diamond$

For instance the proofs of the examples 46 and 47 correspond to the cyclic proof-net of the example 58, which is *equal* to the proof-net of the example 59. Indeed expressions $\triangleright n^\perp \otimes n; (n^\perp \parr n) \otimes (n^\perp \otimes n); n^\perp; n \otimes np^\perp; np \triangleright$ and $\triangleright n^\perp; n \otimes np^\perp; np; n^\perp \otimes n; (n^\perp \parr n) \otimes (n^\perp \otimes n) \triangleright$ denotes the same cyclic permutation.

3.4.5 Proof-nets for the Lambek calculus — with or without empty antecedent

In order to characterize the proof-nets of the Lambek calculus L, which exclude sequents with empty antecedents, we need the following proposition. It involves the notion of a sub-prenet and subproof-net: a sub-prenet (sub-proof-net) is a subgraph of a prenet (proof-net) which is itself a prenet (proof-net). A sub-prenet of a proof-net is not always a proof-net it is possible that SAT does not hold in the sub-prenet (but ØÆ holds).

Proposition 73 *Let* Π *be a proof-net; the following statements are all equivalent:*

1. *Every sub-prenet of* Π *has at least two conclusions. (*ε-FREE*)*
2. *Every sub-proof-net of* Π *has at least two conclusions.*
3. *Every sequentialisation of* Π *contains only sequents with at least two conclusions.*
4. *There exists a sequentialisation of* Π *which contains only sequents with at least two conclusions.*

PROOF : Implications $1 \Rightarrow 2$, $2 \Rightarrow 3$ and $3 \Rightarrow 4$ are straightforward.

$4 \Rightarrow 1$ is shown by induction on the number of links in Π, which is equal to the number of axioms and logical rules of every sequentialisation of Π. Let us consider a sequentialisation Π^* of Π, such that every sequent of it contains at least two formulae. We can assume the last rule of Π^* is not an exchange rule: indeed the same proof without this exchange rule is also a sequentialisation of Π, with all sequents having at least two formulae.

If the last rule of Π^* is an axiom, Π^* consists in this axiom, which contains two formulae. In this case Π is an axiom, whose only sub-prenet is itself, which has two conclusions.

If that rule of Π^* is a two premise rule, applied to two proofs Π'^* and Π''^*, the corresponding link of Π is a splitting *Times* link: Π is obtained from two smaller proof-nets Π' and Π'' connected by this *Times* link. The two proofs Π'^* and Π''^* are possible sequentialisations for Π' and Π'' and these proofs also have sequents with at least two formulae. Thus the induction hypothesis can be applied to Π' and Π'' : every sub-prenet of Π' or of Π'' has at least two conclusions. The intersection of a sub-prenet $s\Pi$ of Π, with Π' (resp. Π'') is a sub-prenet of Π' (resp. Π'') which has $p > 1$ (resp. $q > 1$) conclusions. If the *Times* link is part of $s\Pi$ then the number of conclusions of $s\Pi$ is $p+q-1 > 1$, and otherwise the number of conclusion of $s\Pi$ is $p+q > 1$. Thus, in any case Π satisfies ε-FREE.

If the last rule of Π^* is a one premise rule applied to some proof Π'^*, the corresponding link of Π is a final *Par* link. Let Π' be the proof-net obtained from Π by removing this final *Par* link; it is a proof-net with strictly less links, which has a sequentialisation Π'^* with sequents with more than one conclusions. Hence, by induction hypothesis every sub-prenet of Π' has at least two conclusions. Given a sub-prenet $s\Pi$ of Π, its intersection $s\Pi'$ with Π' has at least two conclusions. It is impossible that $s\Pi'$ has only the two conclusions X and Y. Indeed we know that Π has at least two conclusions, hence it has another conclusion Z in addition to $X \parr Y$. Since Π is a proof-net it is connected, and there exists a path joining $s\Pi'$ to Z conclusion, and this path can be assumed to lie outside $s\Pi'$ — by cutting the part inside $s\Pi'$. So there exists an edge of Π, incident to $s\Pi'$ starting this path. This edge can neither be the R edge below X, nor the one below Y, since any path starting by one of these edges has to enter again $s\Pi'$. But the only way to leave a sub-prenet is from one of its conclusions : therefore $s\Pi'$ has a conclusion which is neither X nor Y. Let p be the number of conclusions of $s\Pi'$. If X and Y are among the p conclusions of $s\Pi'$, then $s\Pi'$ has another conclusion and $p > 2$. Therefore, either $s\Pi$ has $p > 2$ conclusions (when $X \parr Y$ is not one of its conclusions), or $s\Pi$ has $p-1 > 1$ conclusions (when $X \parr Y$ is one of its conclusions). If X or Y is not a conclusion of $s\Pi'$, then $X \parr Y$ is not a conclusion of $s\Pi$, and $s\Pi$ and $s\Pi'$ have the same number of conclusions $p > 1$.

In any case $s\Pi$ has at least two conclusions. ◇

Definition 74 *A Lambek proof-net of conclusion $\Psi_\Pi = \rhd F_1;\cdots;F_n \rhd$ is an intuitionistic cyclic proof-net, i.e. a prenet satisfying*

ØÆ : *there is no æ cycle* alternate elementary cycle.

SAT : *There always exists an æ path between any two vertices.*

INTUI : *Every conclusion F_i is in $\mathsf{Li}^\bullet \cup \mathsf{Li}^\circ$.*

NC : *σ_Π is compatible with Ψ_Π^{at} — the axioms of Π do not intersect.*

A Lambek proof-net is said to be without empty antecedent if, moreover:

ε-FREE : *Every sub-prenet of Π has at least two conclusion.*

Among the four equivalent statements given above, we have chosen the first one, because subprenet are easier to define. It is enough to chose a set of vertices of the proof-net, and to close it by subformula and axiom links, without verifying SAT or ØÆ. When NC and ØÆ hold, this amounts to the following fact: for every subformula G of a conclusion, the first and last atom of G are never linked by an axiom. If $G = H \otimes H'$ then this holds, and if $G = H \parr H'$, this exactly means that there is no sub-net with a single conclusion.

Theorem 75 *Every sequent calculus proof with conclusion $A_1,\ldots,A_n \vdash B$ in L_ε (resp. L) translates into a Lambek proof-net (resp. a Lambek proof-net without empty antecedent) with conclusions $\rhd -A_n;\cdots,-A_1;+B \rhd$.*

Conversely, let Π be a Lambek proof-net (resp. a Lambek proof-net without empty antecedent) with conclusions $\rhd F_1;\ldots;F_n \rhd$. and let i_0 be the unique index in $[1,n]$ such that $F_{i_0} \in \mathsf{Li}^\circ$ and $F_i \in \mathsf{Li}^\bullet$, for $i \neq i_0$. The proof-net Π is the translation of at least a sequent calculus proof in L_ε (resp. L) of

$$(F_{i_0-1})^\bullet_L, (F_{i_0-2})^\bullet_{\mathsf{Lp}}, \ldots, (F_1)^\bullet_{\mathsf{Lp}}, (F_n)^\bullet_{\mathsf{Lp}}, \ldots, (F_{i_0+1})^\bullet_{\mathsf{Lp}} \vdash (F_{i_0})^\circ_{\mathsf{Lp}}$$

PROOF : The first part is a straightforward induction on the sequent calculus proof in L_ε (resp. L).

For the second part, we know from the proposition 72 that there is a sequentialisation corresponding to Π in NC-MLL, with conclusion $\vdash F_1,\cdots,F_n$. Because of proposition 45, this sequent calculus proof in NC-MLL corresponds to a proof of $(F_{i_0-1})^\bullet_L, (F_{i_0-2})^\bullet_{\mathsf{Lp}}, \ldots, (F_1)^\bullet_{\mathsf{Lp}}, (F_n)^\bullet_{\mathsf{Lp}}, \ldots, (F_{i_0+1})^\bullet_{\mathsf{Lp}} \vdash (F_{i_0})^\circ_{\mathsf{Lp}}$ in L_ε. Using $1 \Rightarrow 3$ of proposition 73, it is easily seen that whenever Π is a Lambek proof-net without empty antecedent, the sequentialisation in L_ε is in fact in Li.e. does not contain sequents with only one formula. ◇

Among our proof-net examples, only the examples 57, 58 and 59 are Lambek proof-nets. The example 58 corresponds to the parse structures 41 and 42 : we thus got rid off one *spurious ambiguity* — a classical drawback of categorial grammars, which provides to many proofs/parse structures for a single analysis. One advantage to work with cyclic permutation is that the examples 58 and 59 are *equal*. The example 57 is not a Lambek proof-net without empty antecedent: indeed it does contain a sub-net whose only conclusion is $n^{\perp} \parr n$. It does correspond to the example 39 in L_{ε}.

3.5 Parsing as proof-net construction

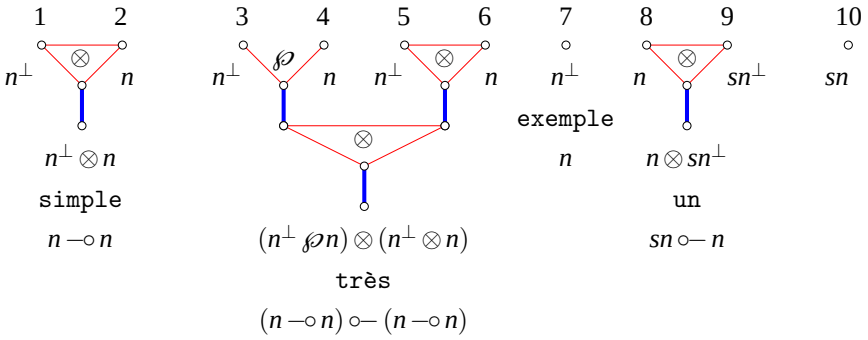
Assume we want to analyze the noun phrase `un exemple très simple`, according to the lexicon provided in the example 39. We need a proof in Lof

$$np \,/\, n, n, (n \setminus n) \,/\, (n \setminus n), n \setminus n \vdash np$$

Because of proposition 75 this amounts to construct a Lambek proof-net without empty antecedent with conclusions:

$$\triangleright\, n^{\perp} \otimes n; (n^{\perp} \parr n) \otimes (n^{\perp} \otimes n); n^{\perp}; n \otimes np^{\perp}; np \,\triangleright$$

— these "linear types" are automatically computed as we did in the example 36, and the order is inverted (see proposition 45). So the lexicon provides automatically the `R&B` subformula trees of the proof-net :



What is missing to obtain a proof-net is σ_{Π}, the axiom links between the occurrences

$$n_1^{\perp}, n_2, n_3^{\perp}, n_4, n_5^{\perp}, n_6 n_7^{\perp}, n_8, np_9^{\perp}, np_{10}$$

They should be placed in such a way that the conditions ØÆ, SAT, INTUI, NC, ε-FREE are met. Of course, INTUI is automatically satisfied since all conclusions belong to $(\mathsf{Lp})^{\perp}$ and one (S) is in Lp

Because axioms link dual formulae there must be an axiom $(np_9^{\perp}, np_{10})$. One should then link the n and the $n^{\perp}$, and this makes 24 possibilities. But, because of the constraints expressed by ØÆ, SAT, NC et ε-FREE we almost have no choice:

$(n_4, n_5^{\perp}) \notin \sigma_\Pi$ — ØÆ, æ cycle with the *Times* link $(n_3^{\perp} \wp n_4) \otimes (n_5^{\perp} \otimes n_6)$.

$(n_5^{\perp}, n_6) \notin \sigma_\Pi$ — ØÆ, æ cycle with the *Times* link between these two atoms.

$(n_3^{\perp}, n_4) \notin \sigma_\Pi$ — ε-FREE, sub-prenet with a single conclusion.

$(n_4, n_7^{\perp}) \notin \sigma_\Pi$ — NC this would force $(n_4, n_5^{\perp})$, which was shown to be impossible. .

$(n_1^{\perp}, n_4) \in \sigma_\Pi$ — only possible choice for n_4.

$(n_2, n_3^{\perp}) \in \sigma_\Pi$ — NC, because of the previous line.

$(n_7^{\perp}, n_8) \notin \sigma_\Pi$ — SAT, yields a non connected proof-net.

$(n_5^{\perp}, n_8), (n_6, n_7^{\perp}) \in \sigma_\Pi$ — only possible choice for these atoms, according to the above decisions.

Hence the only possible solution is the 2-permutation σ_Π given in the example 64 : $(n_1^{\perp}, n_4), (n_2, n_3^{\perp}), (n_5^{\perp}, n_8), (n_6, n_7^{\perp}), (np_9^{\perp}, np_{10})$. It corresponds to the prenet 58.

Next one has to check that the result is a Lambek proof-net, without empty antecedent, and this is straightforward and quick. It corresponds to the sequent calculus proofs given in examples 41 et 42. The identification of various sequent calculus proofs into a single proof-net leads to less possibilities when constructing the proof.

A natural question is the algorithmic complexity of this parsing algorithm. For the less constrained calculus MLL(only satisfying ØÆ and SAT) it is known to be NP complete [84], but the notion of splitting *Times* leads to efficient heuristics using the fact that there never can be any axiom link between the two side of a *Times* link [85]. This considerably reduces the search space. The intuitionistic restriction does not lead to any improvement.

For the non commutative calculi, and in particular for the Lambek calculus, the order constraint NC is so restrictive that one may think that the complexity is polynomial but up to now, the issue is uncertain. The most recent work on this issue is [86], which improves the tabulation techniques introduced in [87]. This idea is to use dynamic programming for the placement of axiom links, defining them by a context-free grammar. Up to now the authors have not been able to bound the size of the information which

---

[84] Patrick Lincoln, John Mitchell, Andre Scedrov, and Natarajan Shankar. Decision problems for propositional linear logic. *Annals of Pure and Applied Logic*, 56(1-3):239–311, 1992.

[85] Philippe de Groote. Linear logic with Isabelle: pruning the proof search tree. In *4^{t}h Workshop on theorem proving with analytic tableaux and related methods*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1995.

[86] Philippe de Groote. A dynamic programming approach to categorial deduction. In *Conference on Automated Deduction, CADE'99*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1999.

[87] Glyn V. Morrill. Memoisation of categorial proof nets: parallelism in categorial processing. In Vito Michele Abrusci and Claudia Casadio, editors, *Third Roma Workshop: Proofs and Linguistics Categories – Applications of Logic to the analysis and implementation of Natural Language*. Bologna:CLUEB, 1996.

has to be memorized in the table, so the polynomiality is not proved. Nevertheless proof-nets seem a good technique for facing this question, the other one being a more precise correspondence with context-free grammars than the Pentus construction.

3.6 Proof-nets for Lambek calculus with cut

The previous sections only deal with cut-free proof-nets which are enough for parsing, because L enjoys cut-elimination. In the mean time Abrusci and Maringelli in [88] extended our characterisation to non cut-free proof-nets, and which up to now is the only one to avoid to refer to embedding in the plane. Let us explain it — this explanation relies on the description of proof-nets provided in previous sections.

We already gave the cut-rule of L, and the cut rule for NC-MLLis

$$\frac{\vdash \Gamma, A \qquad \vdash A^{\perp}, \Delta}{\Gamma, \Delta}\ cut$$

Using the same correspondence between one sided and two sided calculus by means of polarities of the previous sections, this cut rule exactly corresponds to the cut rule of the Lambek calculus, provided that all formulae are intuitionistic.

Firstly a cut is an R-edges between two dual conclusions K and $K^{\perp}$ and the premises K and $K^{\perp}$ of the cut link are not anymore considered as conclusions once there is such an R-edge.

The R-edges are directed in formulation of Abrusci and Maringelli, and their direction is defined as follows (this refers to the pictures in the definition 50)

$\wp$-link Here is the orientation of the two R-edges of this link:

- The R-edge between A and the B-edge $A \wp B$ is directed from the B-edge $A \wp B$ to A.
- The R-edge between B and the B-edge $A \wp B$ is directed from B to the B-edge $A \wp B$.

$\otimes$-link Here is the orientation of the two R-edges of this link:

- The R-edge between A and the B-edge $A \otimes B$ is directed from the B-edge $A \otimes B$ to A.
- The R-edge between B and the B-edge $A \otimes B$ is directed from B to the B-edge $A \otimes B$.
- The R-edge between A and B is directed from A to B.

---

[88] Vito Michele Abrusci and Elena Maringelli. A new correctness criterion for cyclic multiplicative proof-nets. In Retoré [7], pages 449–459.

[7] C. Retoré, editor. *Special Issue on Recent Advances in Logical and Algebraic Approaches to Grammar*, volume 7(4) of *Journal of Logic Language and Information*. Kluwer, 1998.

cut-link This R-edge is not directed, or there is a pair of R-arcs, one from K to $K^\perp$ and one from $K^\perp$ to K.

With this description of proof structure there is no need to say that the axiom links are compatible with the order on the atoms. Indeed, it can be retrieved from the directed graph.

Let us define a *maximal alternate elementary (m.a.e.)* path as follows: it is a *directed* alternate elementary path, starting and ending with a B-edge, which is maximal. These m.a.e. paths always go

from a conclusion or the second premise of a $\wp$-link

to a conclusion or the first premise of a $\wp$-link.

Observe that with this formulation there is no need to consider left or right to find out which of the premises is the first one or the second one: the direction of the R-edges expresses this information.

We also have no order on the conclusions, but using m.a.e. paths we are able to reconstruct it when the proof structure is correct. Given two conclusions of the proof structure we say that $A \varsigma B$ whenever there is an m.a.e. path from B to A.

Then they obtained the following characterisation:

Proposition 76 *A directed bicoloured proof structure* Π *with directed* R*-edges is a cyclic proof-net (i.e. corresponds to a proof of C-MLL) if and only if:*

- *the underlying bicoloured undirected graph contains no alternate elementary cycle*
- *for each* $\wp$*-link there is an m.a.e. path from its second premise to its first premise*
- ς *is a cycle: it is bijective and its order is the number of conclusions.*

Proposition 77 *A directed bicoloured proof structure is a Lambek proof-net if and only if*

- *it is a cyclic proof-net*
- *all its conclusions are intuitionistic formulae*

In fact the criterion for cyclic proof-nets implies that the cut formulae are intuitionistic as well.

3.7 Proof-nets and human processing

Started with a study by Johnson [89] for center embedded relatives and then improved and extended by Morrill [90], proof-nets happen to be interesting parse structure not only from a mathematical viewpoint, but also from a linguistic viewpoint. Indeed they are able to address various performance questions like garden pathing, center embedding unacceptability, preference for lower attachment, and heavy noun phrase shift, that can be observed from parse structure as proof-nets.

We follow Morrill [90] and consider the examples:

Garden pathing

1(a) The horse raced past the barn.

1(b) The horse raced past the barn felt.

2(a) The boat floated down the river.

2(b) ?The boat floated down the river sank.

3(a) The dog that knew the cat disappeared.

3(b) ?The dog that knew the cat disappeared was rescued.

The (b) sentences are correct but seem incorrect. Indeed there is a natural tendency to interpret the first part of the (b) sentences (that are the(a) sentences) hence the other analysis, the correct one, is left out.

Quantifier-scope ambiguity Here are example of quantifier-scope ambiguity, with the preferred reading:

I(a) Someone loves everyone. $\exists\forall$

I(b) Everyone is loved by someone. $\forall\exists$

---

[89] Mark E. Johnson. Proof nets and the complexity of processing center-embedded constructions. In Retoré [7], pages 433–447.

[7] C. Retoré, editor. *Special Issue on Recent Advances in Logical and Algebraic Approaches to Grammar*, volume 7(4) of *Journal of Logic Language and Information*. Kluwer, 1998.

[90] Glyn Morrill. Incremental processing and acceptability. *Computational Linguistics*, 26(3):319–338, 2000. preliminary version: UPC Report de Recerca LSI-98-46-R, 1998.

II(a) Everyone loves someone. $\forall\exists$

II(b) Someone is loved by everyone. $\exists\forall$

So in fact the preference goes for the first quantifier having the wider scope.

Embedded relative clauses.

A(a) The dog that chased the rat barked.

A(b) The dog that chased the cat that saw the rat barked.

A(c) The dog that chased the cat that saw the rat that ate the cheese barked.

B(a) The cheese that the rat ate stank.

B(b) ? The cheese that the rat that the cat saw ate stank.

B(c) ?? The cheese that the rat that the cat that the dog chased saw ate stank.

X(a) That two plus two equals four surprised Jack.

X(b) ?That that two plus two equals four surprised Jack astonished Ingrid.

X(c) ??That that that two plus two equals four surprised Jack astonished Ingrid bothered Frank.

Y(a) Jack was surprised that two plus two equals four.

Y(b) Ingrid was astonished that Jack was surprised that two plus two equals four.

Y(c) Frank was bothered that Ingrid was astonished that Jack was surprised that two plus two equals four.

In his paper [90] Morrill provides an account of our processing preferences, based on our choice for a lower complexity profile. Given an analysis in Lambek calculus of a sentence depicted by a proof-net, we have conclusions corresponding to the syntactic types of the words, and a single conclusion corresponding to S. All these conclusions are cyclically ordered. This cyclic order is easily turned into a linear order by choosing

[90] Glyn Morrill. Incremental processing and acceptability. *Computational Linguistics*, 26(3):319–338, 2000. preliminary version: UPC Report de Recerca LSI-98-46-R, 1998.

a conclusion and a rotation sense. Let us take the output conclusion S as the first conclusion, and let us choose the clockwise rotation with respect to the proof-nets of the previous sections. According to the way proof-nets are drawn we thus are moving from right to left, and we successively meet S, the type of the first word, the type of the second word, etc.

Now let us define *the complexity of a place in between two words w_n and w_{n+1}* (w_0 being a fake word corresponding to S) as the number of axioms $a - a^\perp$ which pass over this place, and such that the a belongs to a conclusion which is, in the linear order, before the conclusion containing $a^\perp$.

Observe that this measure relies on the fact that Lambek calculus is an intuitionistic or polarized calculus in which a and $a^\perp$ are of a different nature: indeed waiting for a category is not the same as providing a category. This measure also depends on the fact that we chose the output S to be the first conclusion: this corresponds to the fact that when someone starts speaking we are expecting a sentence (it could be another category as well, but still we expect some well formed utterance).

Now we can associate to a sentence with n words a sequence of n integers (since S has been added there are n places) called its *complexity profile*.

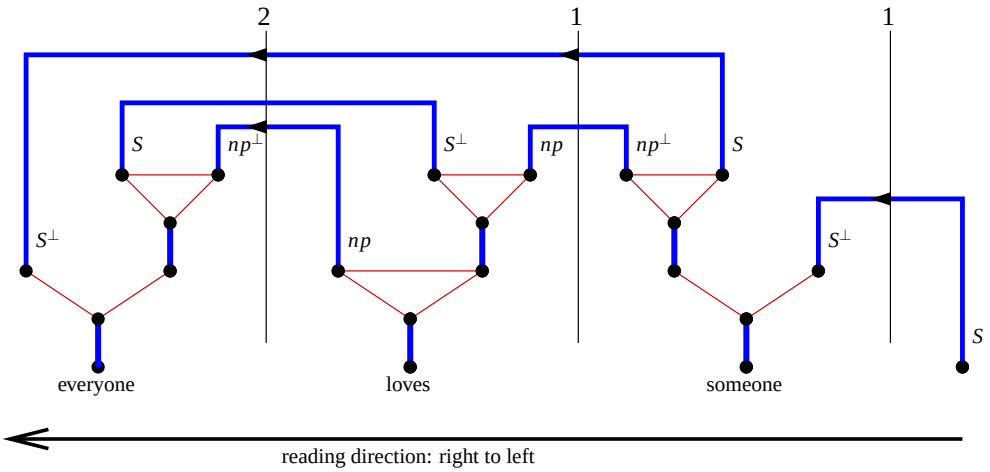
In all examples above, the preferred reading always has the lower profile (that is a profile which is always lower, or at least does not go as high) and hardly parsable sentences have a high profile.

Here we only present one example, as the others provide excellent exercises (and drawing proof-nets on the computer is painful).

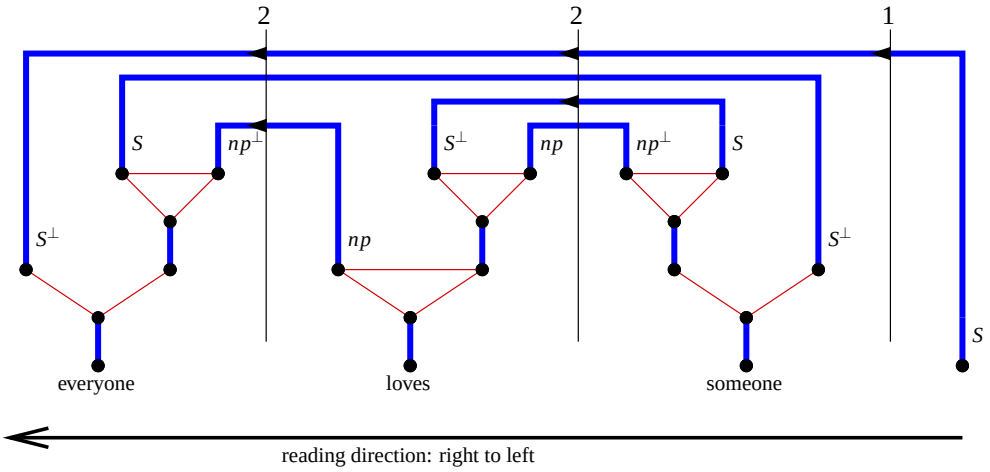
word	type u	$u^\perp$ for constructing the proof-net
someone (subject)	$S/(np\backslash S)$	$S \parr (S^\perp \otimes np)$
(object)	$(S/np)\backslash S$	$(np \otimes S^\perp) \parr S$
everyone (subject)	$S/(np\backslash S)$	$S \parr (S^\perp \otimes np)$
(object)	$(S/np)\backslash S$	$(np \otimes S^\perp) \parr S$
loves:	$(np\backslash S)/np$	$np \otimes (S^\perp \otimes np)$

To complete the example, one should compute the semantics according to the algorithm given in section 2.14.

someone loves everyone ∃∀



someone loves everyone ∀∃



3.8 Semantic uses of proof-nets

Once one is convinced by the relevance of proof-nets as parse structures, it is worth looking at what else can be achieved with proof-nets, in order to avoid translating from one formalism into another, which unpleasant and algorithmically lengthy. As a major advantage of categorial grammars is their relation to Montague semantics, there has been several work in this direction.

As intuitionistic logic can be embedded into linear logic [48] the algorithm fro computing semantic readings can be performed within linear logic. Indeed λ-terms can be depicted as proof-nets, and β-reduction (or cut-elimination) for proof-nets is extremely efficient. In particular the translation can limit the use of replication to its strict minimum. This has been explored with de Groote in [49].

The correspondence between syntax and semantics with proof-nets has been used for generation, firstly by Merenciano and Morrill [91]. Assuming that the semantic of a sentence is known, as well as the semantics of the words, the problem is to reconstruct a syntactic analysis out of these informations. This mainly consists in reversing the process involved in the previous paragraph, which is essentially cut elimination. Using a representation of cut elimination by matrix computations (graphs can be viewed as matrices) Pogodalla has thus defined an efficient method for generation. [92,93,94].

---

[48] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[49] Philippe de Groote and Christian Retoré. Semantic readings of proof nets. In Geert-Jan Kruijff, Glyn Morrill, and Dick Oehrle, editors, *Formal Grammar*, pages 57–70, Prague, 1996. FoLLI.

[91] Josep Maria Merenciano and Glyn Morrill. Generation as deduction. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics, LACL'96*, volume 1328 of *LNCS/LNAI*, pages 77–80. Springer-Verlag, 1996.

[92] Sylvain Pogodalla. Generation with semantic proof nets. Research Report 3878, INRIA, January 2000. http://www.inria.fr/.

[93] Sylvain Pogodalla. Generation in the Lambek calculus framework: an approach with semantic proof nets. In *proceedings of NAACL 2000*, May 2000.

[94] Sylvain Pogodalla. Generation, Lambek calculus, montague's semantics and semantic proof nets. In *proceedings of Coling 2000*, August 2000.