INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Calligramme

Linear Logic, Proof Nets and Categorial Grammars

Lorraine

THEME SYM

Activity Report

2005

Table of contents

1. Team

Head of project-team
Philippe de Groote [DR INRIA]

Vice-Head of project-team
François Lamarche [DR INRIA]

Administrative assistant
Laurence Benini [INRIA]

Staff members INRIA
Bruno Guillaume [CR INRIA]
Sylvain Pogodalla [CR INRIA]

Staff member Université Henri Poincaré-Nancy 1
Adam Cichon [Professor, UHP]

Staff members Institut National Polytechnique de Lorraine
Jean-Yves Marion [Professor, École des Mines de Nancy]
Guillaume Bonfante [Assistant Professor, École des Mines de Nancy]

Staff member Université Nancy 2
Guy Perrier [Professor, Nancy 2]

Post-doctoral fellows
Jean-Yves Moyen [Lecturer at Université Henri Poincaré (until August 31 2005)]
Paulin Jacobé de Naurois [ENS fellow (until August 31 2005)]
Kristofer Johannisson [INRIA postdoctoral fellow (since September 5 2005)]

Ph. D. Students
Mathieu Kaczmarek [BDI CNRS fellow (since October 1)]
Joseph Le Roux [MESR fellow, defense planned in 2006]
Romain Péchoux [MESR fellow, defense planned in 2007]
Sylvain Salvati [Lecturer at École des Mines de Nancy until August 31, defended on June 13]

2. Overall Objectives

2.1. Overall Objectives

Keywords: *categorial grammar*, *implicit complexity*, *lambda calculus*, *linear logic sequent calculus*, *proof nets*, *semantics of natural languages*, *syntactic analysis of natural languages*, *type theory*.

Project-team Calligramme's aim is the development of tools and methods that stem from proof theory, and in particular, linear logic. Two fields of application are emphasized: in the area of computational linguistics, the modelling of the syntax and semantics of natural languages; in the area of software engineering the study of the termination and complexity of programs.

3. Scientific Foundations

3.1. Introduction

Project-team Calligramme's research is conducted at the juncture of mathematical logic and computer science. The scientific domains that base our investigations are proof theory and the λ-calculus, more specifically linear logic. This latter theory, the brainchild of J.-Y. Girard [34] results from a finer analysis of the part played by structural rules in Gentzen's sequent calculus [33]. These rules, traditionally considered as secondary, specify that the sequences of formulas that appear in sequents can be treated as (multi) sets. In the case of intuitionistic logic, there are three of them:

$$\frac{\Gamma \vdash C}{\Gamma, A \vdash C}\ \text{(Weakening)} \qquad \frac{\Gamma, A, A \vdash C}{\Gamma, A \vdash C}\ \text{(Contraction)} \qquad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}\ \text{(Exchange)}$$

These rules have important logical weight: the weakening rule embodies the fact that some hypotheses may be dropped during a derivation; in a similar fashion the contraction rule specifies that any hypothesis can be used an unlimited number of times; as for the exchange rule it stipulates that no order of priority holds between hypotheses. Thus, the presence of the structural rules in the ordinary sequent calculus strongly conditions the properties of the logic that results. For example, in the Gentzen-style formulations of classical or intuitionistic logic, the contraction rule by itself entails the undecidability of the predicate calculus. In the same manner, the use of the weakening and contraction rules in the right half of the sequent in classical logic is responsible for the latter's non-constructive aspects.

According to this analysis, linear logic can be understood as a system that conciliates the constructivist aspect of intuitionistic logic and the symmetry of classical logic. As in intuitionistic logic the constructive character comes from the banning of the weakening and contraction rules in the right part of the sequent. But simultaneously, in order to preserve symmetry in the system, the same rules are also rejected in the other half.

	Propositional linear logic			
	Rudimentary linear logic			
	Negation	Multiplicatives	Additives	Exponentials
Negation	$A^{\perp}$			
Conjunction		$A \otimes B$	$A \& B$	
Disjunction		$A \parr B$	$A \oplus B$	
Implication		$A \multimap B$		
Constants		$1, \perp$	$\top, 0$	
Modalities				!A, ?A

The resulting system, called *rudimentary linear logic*, presents many interesting properties. It is endowed with four logical connectors (two conjunctions and two disjunctions) and the four constants that are their corresponding units. It is completely symmetrical, although constructive, and equipped with an involutive negation. As a consequence, rules similar to De Morgan's law hold in it.

In rudimentary linear logic, any hypothesis must be used once and only once during a derivation. This property, that allows linar logic to be considered as a resource calculus, is due, as we have seen, to the rejection of structural rules. But their total absence also implies that rudimentary linear logic is a much weaker system than intuitionistic or classical logic. Therefore, in order to restore its strength it is necessary to augment the system with operators that recover the logical power of the weakening and contraction rules. This is done via two modalities that give tightly controlled access to the structural rules. Thus, linear logic does not question the usefulness of the structural rules, but instead, emphasizes their logical importance. In fact, it rejects them as epitheoretical rules [32] to incorporate them as logical rules that are embodied in new connectors. This original idea is what gives linear logic all its subtlety and power.

The finer decomposition that linear logic brings to traditional logic has another consequence: the Exchange rule, which so far has been left as is, is now in a quite different position, being the only one of the traditional structural rules that is left. A natural extension of Girard's original program is to investigate its meaning, in other words, to see what happens to the rest of the logic when Exchange is tampered with. Two standard algebraic laws are contained in it: commutativity and associativity. Relaxing these rules entails looking for non-commutative, and non-associative, variants of linear logic; there are now several examples of these. The natural outcome of this proliferation is a questioning of the nature of the structure that binds formulas together in a sequent: what is the natural general replacement of the notion of (multi) set, as applied to logic? Such questions are important for Calligramme and are addressed, for example, in [6].

The activities of project-team Calligramme are organized around three research actions:

- Proof nets, sequent calculus and typed λ-calculi;
- Grammatical formalisms;
- Implicit complexity of computations.

The first one of these is essentially theoretical, the other two, presenting both a theoretical and an applied character, are our privileged fields of application.

3.2. Proof Nets, Sequent Calculus and Typed Lambda Calculi

Keywords: *Curry-Howard isomorphism, denotational semantics, lambda calculus, proof nets, sequent calculus, type theory*.

The aim of this action is the development of the theoretical tools that we use in our other research actions. We are interested, in particular, in the notion of formal proof itself, as much from a syntactical point of view (sequential derivations, proof nets, λ-terms), as from a semantical point of view.

Proof nets are graphical representations (in the sense of graph theory) of proofs in linear logic. Their role is very similar to lambda terms for more traditional logics; as a matter of fact there are several back-and-forth translations that relate several classes of lambda terms with classes of proof nets. In addition to their strong geometric character, another difference between proof nets and lambda terms is that the proof net structure of a proof of formula T can be considered as structure which is *added* to T, as a coupling between the atomic formula nodes of the usual syntactic tree graph of T. Since not all couplings correspond to proofs of T there is a need to distinguish the ones that do actually correspond to proofs; this is called a *correctness criterion*.

The discovery of new correctness criteria remains an important research problem, as much for Girard's original linear logic as for the field of non-commutative logics. Some criteria are better adapted to some applications than others. In particular, in the case of automatic proof search, correctness criteria can be used as invariants during the inductive process of proof construction.

The theory of proof nets also presents a dynamic character: cut elimination. This embodies a notion of normalization (or evaluation) akin to β-reduction in the λ-calculus.

As we said above, until the invention of proof nets, the principal tool for representing proofs in constructive logics was the λ-calculus. This is due to the Curry-Howard isomorphism, which establishes a correspondence between natural deduction systems for intuitionistic logics and typed λ-calculi.

Although the Curry-Howard isomorphism owes its existence to the functional character of intuitionistic logic, it can be extended to fragments of classical logic. It turns out that some constructions that one meets in functional progamming languages, such as control operators, can presently only be explained by the use of deduction rules that are related to proof by contradiction [35]

This extension of the Curry-Howard isomorphism to classical logic and its applications has a perennial place as research field in the project.

3.3. Categorial Grammars

Keywords: *Montague semantics, categorial grammar, semantics of natural languages, syntactic analysis of natural languages, syntactic inference, tree description*.

Lambek's syntactic calculus, which plays a central part in the theory of categorial grammars, can be seen a posteriori as a fragment of linear logic. As a matter of fact it introduces a mathematical framework that enables extensions of Lambek's original calculus as well as extensions of categorial grammars in general. The aim of this work is the development of a model, in the sense of computational linguistics, which is more flexible and efficient than the presently existing categorial models.

The relevance of linear logic for natural language processing is due to the notion of resource sensivity. A language (natural or formal) can indeed be interpreted as a system of resources. For example a sentence like *The man that Mary saw Peter slept* is incorrect because it violates an underlying principle of natural languages, according to which verbal valencies must be realized once and only once. Categorial grammars formalize this idea by specifying that a verb such as saw is a resource which will give a sentence S in the presence of a nominal subject phrase, NP, and only one direct object NP. This gives rise to the following type assigment:

Mary, Peter:		NP
saw		$(NP \setminus S)/NP$

where the slash (/) (resp. the backslash (\)) is interpreted as fraction pairings that simplify to the right (resp. to the left). However we notice very soon that this simplification scheme, which is the basis of Bar-Hillel grammars [30], is not sufficient.

Lambek solves this problem by suggesting the interpretation of slashes and backslashes as implicative connectors [36], [37]. Then not only do they obey the *modus ponens* law which turns out to be Bar-Hillel's simplification scheme

$$\frac{\Gamma \vdash A \quad \Delta \vdash A \backslash B}{\Gamma, \Delta \vdash B}\ \text{(modus ponens)} \qquad \frac{\Gamma \vdash B/A \quad \Delta \vdash A}{\Gamma, \Delta \vdash B}\ \text{(modus ponens)}$$

but also the introduction rules:

$$\frac{A, \Gamma \vdash B}{\Gamma \vdash A \backslash B}\ \backslash\text{-intro} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash B/A}\ /\text{-intro}$$

The Lambek calculus does have its own limitations. Among other things it cannot treat syntactical phenomena like medial extraction and crossed dependencies. Thus the question arises: how can we extend the Lambek calculus to treat these and related problems? This is where linear logic comes into play, by offering an adequate mathematical framework for attacking this question. In particular proof nets appear as the best adapted approach to syntactical structure in the categorial framework.

Proof nets offer a geometrical interpretation of proof construction. Premises are represented by proof net fragments with inputs and outputs which respectively model needed and offered resources. These fragments must then be combined by pairing inputs and outputs according to their types. This process can also be interpreted in a model-theoretical fashion where fragments are regarded as descriptions for certain class of models: the intuitionistic multiplicative fragment of linear logic can be interpreted on directed acyclic graphs, while for the implicative fragment, trees suffice [39].

This perspective shift from proof theory to model theory remains founded on the notion of resource sensitivity (e.g. in the form of polarities and their neutralization) but affords us the freedom to interpret these ideas in richer classes of models and leads to the formalism of Interaction Grammars. For example:

- where previously we only considered simple categories with polarities, we can now consider complex categories with polarized features.
- We can also adopt more expressive tree description languages that allow us to speak about dominance and precedence relations between nodes. In this fashion we espouse and generalize the monotonic version of Tree Adjoining Grammars (TAG) as proposed by Vijay-Shanker [41].
- Contrary to TAG where tree fragments can only be inserted, Interaction Grammars admit models where the interpretations of description fragments may overlap.

3.4. Implicit Complexity of Computations

Keywords: *Complexity theory*, *Curry-Howard isomorphism*, *lambda calculus*, *termination orders*, *theory of programming*, *types*.

The construction of software which is certified with respect to its specifications is more than ever a great necessity. It is crucial to ensure, while developing a certified program, the quality of the implementation in terms of efficiency and computational resources. Implicit complexity is an approach to the analysis of the resources that are used by a program. Its tools come essentially from proof theory. The aim is to compile a program while certifying its complexity.

The meta-theory of programming traditionally answers questions with respect to a specification, like termination. These properties all happen to be *extensional*, that is, described purely in terms of the relation between the input of the program and its output. However, other properties, like the efficiency of a program and the resources that are used to effect a computation, are excluded from this methodology. The reason for this is inherent to the nature of the questions that are posed. In the first case we are treating extensional properties, while in the second case we are inquiring about the manner in which a computation is effected. Thus, we are interested in *intensional* properties of programs.

The complexity of a program is a measure of the resources that are necessary for its execution. The resources taken into account are usually time and space. The theory of complexity studies the problems and the functions that are computable given a certain amount of resources. One should not identify the complexity of functions with the complexity of programs, since a function can be implemented by several programs. Some are efficient, others are not.

One achievement of complexity theory is the ability to tell the "programming expert" the limits of his art, whatever the amount of gigabytes and megaflops that are available to him. Another achievement is the development of a mathematical model of algorithmic complexity. But when facing these models the programming expert is often flabbergasted. There are several reasons for this; let us illustrate the problem with two examples. The linear acceleration theorem states that any program which can be executed in time $T(n)$ (where n is the size of the input) can be transformed into an equivalent problem that can be executed in time $\epsilon T(n)$, where ϵ is "as small as we want". It turns that this result has no counterpart in real life. On the other hand a function is feasible if it can be calculated by a program whose complexity is acceptable. The class of feasible functions is often identified with the class Ptime of functions that are calculable in polynomial time. A typical kind of result is the definition of a progamming language LPL and the proof that the class of functions represented by that language is exactly the class Ptime. This type of result does not answer the programming expert's needs because the programming language LPL does not allow the "right algorithms", the ones he uses daily. The gulf between the two disciplines is also explained by differences in points of view. The theory of complexity, daughter of the theory of computatibility, has conserved an extensional point of view in its modelling practices, while the theory of programming is intrinsically intensional.

The need to reason on programs is a relevant issue in the process of software development. The certification of a program is an essential property, but it is not the only one. Showing the termination of a program that has exponential complexity does not make sense with respect to our reality. Thus arises the need to construct tools for reasoning on algorithms. The theory of implicit complexity of computations takes a vast project to task, namely the analysis of the complexity of algorithms.

4. Application Domains

4.1. Modelling the Syntax and Semantics of Natural Languages

4.1.1. Abstract Categorial Grammars

Abstract Categorial Grammars (ACGs) are a new categorial formalism based on Girard's linear logic. This formalism, which sticks to the spirit of current type-logical grammars, offers the following features:

- Any ACG generates two languages, an abstract language and an object language. The abstract language may be thought as a set of abstract grammatical structures, and the object language as the set of concrete forms generated from these abstract structures. Consequently, one has a direct control on the parse structures of the grammar.
- The langages generated by the ACGs are sets of linear λ-terms. This may be seen as a generalization of both string-langages and tree-langages.
- ACGs are based on a small set of mathematical primitives that combine via simple composition rules. Consequently, the ACG framework is rather flexible.

Abstract categorial grammars are not intended as yet another grammatical formalism that would compete with other established formalisms. It should rather be seen as the kernel of a grammatical framework in which other existing grammatical models may be encoded.

4.1.2. Interaction Grammars

Interaction Grammars (IGs) are a linguistic formalism that aims at modelling both the syntax and the semantics of natural languages according to the following principles:

- An IG is a monotonic system of constraints, as opposed to a derivational/transformational system, and this system is multidimensional: at the syntactic level, basic objects are tree descriptions and at the semantic level, basic objects are Directed Acyclic Graph descriptions.
- The synchronization between the syntactic and the semantic levels is realized in a flexible way by a partial function that maps syntactic nodes to semantic nodes.
- Much in the spirit of Categorial Grammars, the resource sensitivity of natural language is built-in in the formalism: syntactic composition is driven by an operation of cancellation between polarized morpho-syntactic features and in parallel, semantic composition is driven by a similar operation of cancellation between polarized semantic features.

The formalism of IG stems from a reformulation of proof nets of Intuitionistic Linear Logic (which have very specific properties) in a model-theoretical framework [39] and it was at first designed for modelling the syntax of natural languages [40].

4.1.3. Grammatical and lexical resources for French

The relevance of new linguistic formalisms needs to be proved by experiments on real corpora. Parsing real corpora requires large scale grammars and lexicons. There is a crucial lack of such resources for French and all researchers committed in NLP projects for French based on different formalisms are confronted with the same problem. Now, building large scale grammars and lexicons for French demands a lot of time and human resources and it is crucial to overcome the multiplicity of existing formalisms by developing common and reusable tools and data. This is the sense of two directions of research:

1. The modular organization of formal grammars in a hierarchy of classes allows the expression of linguistic generalizations and it makes possible their development and their maintenance on a large scale. To be used in NLP applications such modular grammars have to be compiled into operational grammars. By comparison with the area of programming languages, we write source grammars in a language with a high abstraction level and then we compile them automatically to object grammars, directly usable by NLP applications.
   Considering the multiplicity of linguistic formalisms, it would be interesting to express the various source grammars that can written in different formalisms, in a common abstract language and to compile them with the same tool associated to this language. XMG [21] is a first experiment in this direction: for the moment, it allows the edition and the compilation of source grammars for TAGs and IGs. Moreover, we can hope that the use of a common language of syntactic description with a high level of abstraction make easier the reusability of some parts of grammars from one formalism to another.

2. With the same preoccupation of reusability, it is important to develop syntactic and semantic lexicons which contain only purely linguistic information and which are independent of the different existing grammatical formalisms. Now, a mechanism must be foreseen to combine these lexicons with the grammars built in the various formalisms. A convenient way of doing this is to design the entries of such lexicons in the form of feature structures and to associate also feature structures with the elementary constructions of the grammars. Then, their anchoring in the lexicons is realized by unification of the two kinds of feature structures. The construction of a syntactic and a semantic lexicon for French can be envisaged either by acquisition from corpora or be re-use of existing lexical information.

4.2. Termination and complexity of programs

The theory of implicit complexity is quite new and there are still many things to do. So, it is really important to translate current theoretical tools into real applications; this should allow to validate and guide our hypotheses. In order to do so, three directions are being explored.

1. First order functional programming. A first prototype, called ICAR has been developed and should be integrated into ELAN (http://elan.loria.fr).
2. Extracting programs from proofs. Here, one should build logical theories in which programs extracted via the Curry-Howard isomorphism are efficient.
3. Application to mobile code system. This work starts in collaboration with the INRIA Cristal and Mimosa project-teams.

5. Software

5.1. Leopar

5.1.1. Description of the software

LEOPAR is a parser for natural languages which is based on the formalism of Interaction Grammars (IG) [40]. It uses a parsing principle, called "electrostatic parsing" which is based on neutralizing opposite polarities. A positive polarity corresponds to an available linguistic feature and a negative one to an expected feature.

Parsing a sentence with an Interaction Grammar (IG) consists in first selecting a lexical entry for each of its words. A lexical entry is an underspecified syntactic tree, a tree description in other words. Then, all selected tree descriptions are combined by partial superposition guided by the aim of neutralizing polarities: two opposite polarities are neutralized by merging their support nodes. Parsing succeeds if the process ends with a minimal and neutral tree. As IG are based on polarities and under-specified trees, LEOPAR uses some specific and non-trivial data-structures and algorithms.

The electrostatic principle has been intensively considered in LEOPAR. The theoretical problem of parsing IGs is NP-complete; the nondeterminism usually associated to NP-completeness is present at two levels: when a description for each word is selected from the lexicon, and when a choice of what nodes to merge is made. Polarities have shown their efficiency in pruning the search tree for these two steps:

- In the first step (tagging the words of the sentence with tree descriptions), we forget the structure of descriptions, and only keep the bag of their features. In this case, parsing inside the formalism is greatly simplified because composition rules reduce to the neutralization of a negative feature-value pair $f \longleftarrow v$ by a dual positive feature-value pair $f \rightarrow v$. As a consequence, parsing reduces to a counting of positive and negative polarities present in the selected tagging for every pair (f, v): every positive occurrence counts for +1 and every negative occurrence for –1, the sum must be 0.
- In the second step (node-merging phase), polarities are used to cut off parsing branches whose trees contain too many uncancelled polarities.

5.1.2. *Current state of the implementation*

A first prototype has been developed until 2003 by Guillaume Bonfante, Bruno Guillaume. This implementation has many drawbacks and is not maintained.

Since 2004, a new implementation of LEOPAR started. Guillaume Bonfante, Bruno Guillaume, Guy Perrier and Sylvain Pogodalla work on this new implementation. The current implementation (17,000 lines of Ocaml) provides different running modes:

- automatic parsing of a sentence or a set of sentences;
- manual parsing (the user chooses the couple of nodes to merge);
- visualization of grammars produced by XMG or of set of description trees associated to some French word.

The main improvements with respect to the previous implementation are:

- a finer data structure for tree description: there are now two notions of precedence (direct and large) and there is arity constraint on nodes;
- a new algorithm for the first step (tagging) which uses deterministic automata and provides a finer control on the way the filters are applied;
- a new algorithm for the node-merging phase: more constraint propagations are used (hence the search space is reduced);
- grammars created with XMG are now directly usable in LEOPAR;
- a new graphical interface (using GTK) which is useful for debugging of grammar.

The current implementation is available on the web (http://www.loria.fr/equipes/calligramme/leopar/) under the CECILL License (http://www.cecill.info).

The current implementation comes with a middle-size coverage grammar for French (710 tree descriptions in the grammar produced with XMG). It includes also morphological and syntactical lexicons that cover the French examples of the TSNLP (Test Suite for Natural Langage Processing) [38].

5.2. XMG

The eXtensible MetaGrammar [21] (XMG) is a tool for generating large coverage grammars from concise descriptions of linguistic phenomenena (the so-called *metagrammar*). This software is a Calligramme and Langue Et Dialogue joint work and was formerly known as The Metagrammar Workbench.

This software is based on 2 important concepts from logic programming, namely the Warren's Abstract Machine and constraints on finite set. It has been developed by Benoît Crabbé, Yannick Parmentier, Denys Duchier and Joseph Le Roux. The first release is available at http://sourcesup.cru.fr/xmg. It is now maintained by PhD students Yannick Parmentier and Joseph Le Roux.

At current stage of implementation, XMG generates Tree adjoining grammars and Interaction grammars but the underlying formalism is generic so it could be extended to others grammars like dependency grammars or lexical functional grammars, depending on users' requests.

XMG has been used in order to design *realistic grammars* for French, that is to say grammars covering common linguistic knowledge and phenomena. Guy Perrier wrote an Interaction Grammar that is available with LEOPAR. Benoît Crabbé wrote a Tree Adjoining Grammar inspired by the well known FTAG evaluated in less than 3 months. Claire Gardent is using XMG to design a tree adjoining grammar with semantics. Joseph Le Roux is also designing an Interaction Grammar of coordination with XMG.

XMG also has users outside the LORIA: Owen Rambow (Colombia University) is implementing a grammar for Arab, designed with XMG and PhD students from Penn University also work with this tool.

5.3. Linguistic Resource Development

In order to get actual lexicons to run LEOPAR, we needed to develop some lexical resources. The general architecture is the following:

1. Lexicon resources are described in two different databases: one for morphological informations and the other one for syntactical aspects; the two databases are compiled in a morpho-syntactical lexicon that combines the two kinds of information. In this compiled lexicon, feature structures are used to represent morpho-syntactical features associated to each flexed form.
2. From a metagrammar, through XMG (see 5.2), we generate anonymous tree descriptions that can occur in the targeted language (French); each tree description comes with a feature structure (called interface) that describes how this tree should be anchored in the lexicon database.
3. Finally, we use feature structure unification to combine grammatical and morpho-syntactic databases. When unification between the feature structure of a word (given by the morpho-syntactic database) and the interface of a tree description succeeds, the word anchored the corresponding tree description which is now fully instantiated.

To this end, in addition to the tools to merge the different kinds of lexicons, Bruno Guillaume and Sylvain Pogodalla have developed a tool[1] that can produce LEOPAR formatted morphological lexicons from external morphological lexicons (as for now, from our own verb descriptions[2] and from the morphological lexicon Morphalou[3] provided by the ATILF[4]).

This tool is also used in the concordancers we provide[5] (based on the Test Suites for Natural Language Processing (TSNLP[6]) and on *Le tour du Monde en 80 jours*[7](J. Verne). These concordancers are used in our project-team and in the Langue et Dialogue INRIA project-team to help grammar writers.

Two students at École des Mines (Damien Auricchio and Nelson Da Silva) also worked on factorizing morphological informations of flexed forms and on comparison of UNITEX[8] morphological lexicon and our own verb lexicon during a training period of three months.

5.4. ACG related software

A development environment for ACGs is being developed by Bruno Guillaume, Philippe de Groote and Sylvain Pogodalla. The main features are the abilities to read signatures and lexicons and to realize object terms from abstract ones. This new version integrate the ability to use features in types. Parsing (to build abstract terms from object terms) and example grammars are being developed.

6. New Results

6.1. Proof Nets, Sequent Calculus and Typed Lambda Calculi

6.1.1. Proof Nets for Classical Logic

The papers [25], [24] show the ongoing development of the theory of proof denotations for classical propositional logic pursued by François Lamarche and Lutz Straßburger. The first paper presents two concrete proof net models, that differ by the semiring of coefficients which is used to tally—much like ordinary axiom links in linear logic—how axioms are used in a proof. The model based on $\mathbb{N}$, the ordinary integers, counts the

[1] http://www.loria.fr/equipes/calligramme/litote/LIB/LEX-READER/lex-reader.tar.gz
[2] About 6 000 verbs, 300 000 flexed forms, available at http://www.loria.fr/equipes/calligramme/litote/
[3] http://actarus.atilf.fr/morphalou/
[4] Analyse et Traitement Informatique de la Langue Française http://www.atilf.fr/
[5] http://www.loria.fr/equipes/calligramme/litote/concordancer/
[6] http://cl-www.dfki.uni-sb.de/tsnlp/
[7] Available on the site *ABU : la Bibliothèque Universelle* http://abu.cnam.fr/
[8] http://www-igm.univ-mlv.fr/~unitex/

number of times an axiom is used, but its cut-elimination process is not confluent. This desirable property is obeyed by the other model, based on the semiring $\mathbb{B}$ of Booleans, which displays only the presence or absence of an axiom. These models have surprising properties with regards to what people have always expected about the relationship between proof denotations and computations (the “Curry-Howard isomorphism”). They also are intimately related to complexity problems on Boolean satisfiability.

The second paper is a complete study of the category-theoretical properties of the $\mathbb{B}$-based model. A hierarchy of axioms is proposed to give several possible answer to the question "what is a categorical model of Boolean logic", and the $\mathbb{B}$-model is shown to be the free category (with atomic formulas as generators) for the right choice of axioms among this lot. This paper’s approach has some things in common with the work of Führmann and Pym (who start with completely different concrete models), but it also shows several important differences, in particular the avoidance of any 2-categorical structure in the axiomatization.

6.2. Categorial Grammars

6.2.1. Lambek-calculus

We have studied the product-free associative Lambek calculus extended with a structural modality à la Girard, which allows the left structural rules (weakening, contraction, and exchange) to be performed in a controlled way. In particular, we have shown that any recursively enumerable language can be described by a categorial grammar based on this calculus [14].

6.2.2. Abstract Categorial Grammars

We have studied the expressive power of the Abstract Categorial Grammars (ACG) by showing how to represent several grammatical formalisms as ACGs, including Tree Adjunction Grammars, and Linear Context Free Rewriting Systems [15].

We have studied some of the language-theoretic properties of the ACGs, such as the decidability and complexity of membership, universal membership, and emptiness. In particular, we have established the NP-completeness of membership for arbitrary lexicalized ACGs. We have also shown that this same problem is polynomial for second-order ACGs, by developing Earley-like parsing algorithms [7].

6.2.3. Interaction Grammars

Joseph Le Roux has adapted the so-called Earley parsing algorithm to Interaction Grammars. Although the parsing problem is NP-Complete, this tabular algorithm lets us reuse common material between the different parses of a sentence. That algorithm will soon be implemented in Leopar to test the actual improvement on real corpora.

6.3. Development of linguistic resources

6.3.1. Extraction of a syntactical lexicon from Maurice Gross’ grammar lexicon

For French, there exists to date no reference lexicon that would contain detailed extensive subcategorisation information (that is, information about the complements of natural language predicative items such as verbs, deverbal nouns and predicative adjectives).

In the papers [23] and [12], Claire Gardent (Langue et Dialogue team) Bruno Guillaume, Guy Perrier and Ingrid Falk (ATILF) propose a method for producing such a syntactical lexicon from the LADL tables (Maurice Gross’ grammar lexicon in other words).

LADL tables provides a systematic description of the syntactic properties of the functors of French namely, verbs, predicative nouns and adjectives. The subcategorisation information contained in this lexicon is both detailed and extensive.

Although the LADL tables are rich in content, their current format makes them difficult to use in NLP application. The reasons for this are threefold:

1. The format itself is non standard. In NLP applications, subcategorisation information is standardly gathered within a syntactic lexicon which associates with each predicative item the set of its possible subcategorisation frames. Further, subcategorisation frames are usually represented by a set of feature structures where each element in the set encode the linguistic properties either of the verb or one of the argument occuring in the frame being described. To be easily usable by NLP applications, it is highly desirable that such a syntactic lexicon be derived from the LADL tables.
2. The structure of the tables is either implicit in the headings or altogether absent (in the electronic version available). For instance, the dependency between columns is not marked; subset of columns that describe atomic disjunction needs to be automatically recovered from the fact that the columns are adjacent and share the same feature in their heading.
3. The headings are non standard and need to be translated in feature structure specification that are more in line with current practice in syntactic annotation.

We propose a method for extracting from the LADL tables, an NLP oriented syntactic lexicon. In essence, this method aims at making the table structure explicit and at translating the headings into standard practice feature structure notation. Specifically, it consists in the following three steps:

1. For each table, a graph is (manually) produced which represents the interpretation of the table. This graph makes the table structure explicit and translates the headings into path equations.
2. A graph traversal algorithm is specified such that, given a graph and a table, it produces for each entry in that table the set of subcategorisation frames associated by the table with that entry. The resulting lexicon is called a LADL-lexicon and closely reflects the content of the LADL table. Some of the information it contains is not currently used by most NLP tools in particular, parsers and surface realisers.
3. A simplification algorithm is specified such that given a LADL-lexicon, it produces an NLP-lexicon. The NLP lexicon is a simplified version of the LADL-lexicon where only features relevant for parsing/generating are preserved and which only partially reflects the content of the LADL table. It is with this lexicon that NLP is expected to proceed.

6.3.2. Development of an interaction Grammar for French

By using XMG, Guy Perrier has developed an interaction grammar for French. The methodology is inspired by Benoit Crabbé, who has developed a large French TAG [31].

The source grammar is composed of 312 classes organized in a inheritance hierarchy with two operators of conjunction and disjunction. The leaves of the hierarchy describe elementary phenomena of the grammar. Conjunctions and disjunctions express two ways of representing complex phenomena: for instance, a particular diathesis for a verb can result from the conjunction of classes representing specific realizations of its aguments and the realization of a particular predicate argument structure can be expressed by the disjunction of the classes representing the different diatheses.

The compiled grammar is composed of 710 tree descriptions mainly covering the following phenomena of the French syntax:

- most subcategorisation frames for verbs, predicative adjectives and nouns,
- active, passive, middle and reflexive diatheses combined with personal and impersonal subject constructions,
- grammatical words and related syntactic constructions (clitics, personal, relative and interrogative pronouns, complementizers, prepositions, negations, auxiliary verbs ...),

- some phenomena hard to model such as: pied-piping in relative and interrogative clauses, islands for wh-extraction, long distance dependencies related to negative expressions (*“ne...aucun”*, *“ne...personne”*), past participle agreement in presence of the auxiliary *“avoir”*, control of the subject for the infinitives…

The grammar is in the process of being evaluated on the TSNLP test suite [38].

6.4. Implicit Complexity of Computation

The goal is to determine and to guaranty resources by static analysis which are necessary to run a system. By resources, we mean heap memory, stack size, size of the function output values, the runtime of a program...One of the originality of our approach is that our ideas are taken from logics, type theory and termination methods. This field of research is called implicit computational complexity.

More precisely, our goals split into two complementary points. The first point concerns quasi-interpretations. See the survey [10]. Our objectives are to try to demonstrate that this approach is feasible. For this, it is necessary to have heuristic to find program quasi-interpretations, see [18], [19]. Then, we try to extend the quasi-interpretations methods in order to analyze more algorithms in a more easily way. Thus, we have introduce the concept of “sup-interpretation”. And the result has just been accepted at Flops06. The long term goal is to have methods to analyze functional and imperative programs. Another directions that we are currently exploring is to use automata theory to predict resources. Lastly, in a more short term goal, we work inside a Pessoa PAI with R. Kähle and I. Oitavem to characterize small complexity classes like NC^k. We also work on the more fundamental question of understanding the BSS computational models over real numbers, see [11], [20].

This year we make a research turn by considering computer virus, see [9], [16], [17]. Indeed, attacks of the type denial of services for which the memory resource is critical may cause for example a “buffer overflow”. So, we could expect that static analysis that we develop in the context of programming language of high level with quasi-intepretations or low level with the methods developed by Marion & Moyen based on Petri nets and linear algebra could apply.

7. Other Grants and Activities

7.1. Regional Actions

- Calligramme is part of the “Ingénierie des langues, du document, de l’information scientifique et culturelle” theme of the “contrat de plan État-Région”. Calligramme’s contributions range over the syntactical and semantical analysis of natural language, the building of wide coverage lexical resources, and the development of software specialized for those tasks;
- Calligramme is part of the “Qualité et sûreté des Logiciels (QSL)” theme of the “contrat de plan État-Région”. Jean-Yves Marion is head of the QSL theme;
  Web page at http://qsl.loria.fr

7.2. National Actions

7.2.1. Action Concertée Incitative (ACI) Demonat

Calligramme is involved in the ACI DEMONAT, in section “Nouvelles interfaces des mathématiques”, together with the “Logique” group of “Université de Savoie” and the “TALaNa” team of “Université Paris 7”. The project concerns the parsing and the checking of mathematical proofs written in natural language.

7.2.2. Action Concertée Incitative (ACI) CRISS

Calligramme is involved in the ACI CRISS, in section “Sécurité informatique”. Its purpose, which can be read from the full title, is “Contrôle de ressources et d’interfaces pour les systèmes synchrones”. It is headed by Roberto Amadio at the University of Marseille, and the co-ordinator on Calligramme’s side is Jean-Yves Marion.

Web page at http://www.pps.jussieu.fr/~amadio/Criss/criss.html

7.2.3. Action Concertée Incitative (ACI) Géocal

This “nouvelles interfaces des mathématiques” ACI (2003-2006) regroups several research teams in both mathematics and computer science and is concerned, as its name implies, with the application to computer science of techniques developed for modern geometry. It is headed by Thomas Ehrhard at the CNRS in Marseille, and the co-ordinator on Calligramme’s side is Jean-Yves Marion.

Web page at http://iml.univ-mrs.fr/~ehrhard/geocal/geocal.html

7.2.4. Action Concertée incitative (ACI) Inval

Headed by Eric Goubault, this three-year action (starting in November) is the direct descendent of Géocal and a smaller ACI that ended in 2005. Its aims are the study and development of algebraic invariants of computation, inspired by traditional homology and homotopy in algebraic topology. The co-ordinator on Calligramme’s side is François Lamarche.

7.2.5. Réseau National des Technologies Logicielles (RNTL)

Calligramme, through Jean-Yves Marion, is a participant in the Ministry of Industry RNTL project Averroes.

Web page: http://www-verimag.imag.fr/AVERROES/

7.2.6. LexSynt project

Calligramme is involved in the LexSynt project. Thirteen French-speaking research teams work on this project. It aims at developping a syntactic lexicon with large coverage for French. In order to be usable in various NLP applications, this lexicon is independent of any grammatical formalism.

Web page: http://lexsynt.inria.fr

7.3. European Actions

- Calligramme is involved in the european network CoLogNET (Computational Logic Network) on the themes: logic methodology and foundational tools, logic and natural language processing.

7.4. Visits and invitation of researchers

- Philippe de Groote and Sylvain Salvati visited Makoto Kanazawa (NII, Tokyo) from February 12th to February 27th.
- Philippe de Groote and Sylvain Pogodalla visited Carl Pollard (Ohio State University) from November 30th to December 6th.
- Ryo Yoshinaka (Makoto Kanazawa’s PhD student) visited the Calligramme Project from December 8th to December 21th.

8. Dissemination

8.1. Activism within the scientific community

- Guillaume Bonfante is the vice president of the hiring committee, section 27, of the INPL, since April 2003.
- Guillaume Bonfante is an elected member of the scientific council of the INPL since July 2003.
- Guillaume Bonfante is a member of the engineering part of the Comipers hiring committee at LORIA.
- Adam Cichon was elected member of the “Conseil National des Universités” (CNU), section 27.
- Philippe de Groote is President of the INRIA-Lorraine Projects Committee (starting September 2004), and a member of INRIA’s evaluation board.
- Philippe de Groote is a member of the LORIA management board, and of the LORIA laboratory council.
- Philippe de Groote is an associate editor of the journal *Higher-Order and Symbolic Computation*. He belongs to the editorial board of the series *Papers in Formal Linguistics and Logic* (Bulzoni, Roma), and *Cahiers du Centre de Logique* (Academia-Bruylant, Louvain-la-Neuve).
- Philippe de Groote was member of the program committees of LACL’05, and UNIF’05.
- François Lamarche is member of the Bureau of the Département de Formation Doctorale in Computer Science of the IAEM doctoral school.
- François Lamarche heads the research (theses, postdocs and *ingénieurs spécialistes*) section of the Comipers hiring commitee at LORIA.
- François Lamarche was chairman (both Program Committe and Organization Committee) of the “Structures and Deductions 2005” (SD05) http://www.prooftheory.org/sd05/ which was held in Lisbon on July 16–17, as a satellite worshop of the ICALP 2005 international conference. The worshop’s theme was the emergence of new methods in proof theory; the proceedings are available at http://www.ki.inf.tu-dresden.de/~paola/SD05/SD05-Proc.pdf.
- Jean-Yves Marion is member of the steering committee of the International workshop on Logic and Computational Complexity (LCC).
- Jean-Yves Marion is member of the hiring committee (CS) at the University of Metz, section 27, since Sept. 2004.
- Jean-Yves Marion is member of the hiring committee at INPL (Professors and Lecturers), section 27, since February 2002.
- Jean-Yves Marion was elected to the scientific council of INPL in July 2003 and member of the board.
- Jean-Yves Marion initiated and organizes the monthly QSL seminars http://qsl.loria.fr. Every seminar gathers between 10 and 40 participants. There were 22 seminars since January 2003.
- Guy Perrier is a member of the editorial board of the revue *Traitement Automatique des Langues*.
- Guy Perrier is a member of the Program Committee of the conference TALN’2006.
- Guy Perrier is a member of the Bureau of the Département de Formation Doctorale in Computer Science of the IAEM doctoral school.

8.2. Teaching

- Jean-Yves Marion is in charge of the option “Ingénierie des systèmes informatiques” at École des Mines starting in September.
- Jean-Yves Marion took part in the creation of the formation in computational biology at École des Mines and is in charge of the course on “Bases et banques de données”.
- Guy Perrier heads the specialization “Traitement Automatique des Langues” which is common to the masters in computer science and cognitive sciences of the universities Nancy2 and Henri Poincaré.
- Guy Perrier is in charge of the organization of the course on *tools and algorithms for the parsing of natural languages*, which he is teaching with Bertrand Gaiffe in the master’s specialization “Traitement Automatique des Langues”.
- Philippe de Groote and Sylvain Pogodalla gave a course on ACGs at the ESSLLI 2005 (European Summer School in Logic, Language and Information), in August in Edinburgh.
- Philippe de Groote is teaching the course “Sémantique computationnelle” of the Nancy master specialization “Traitement Automatique des Langues”.
- Philippe de Groote and Gérard Huet are teaching the course “ Structures Informatiques et Logiques pour la Modélisation Linguistique” of the “Master Parisien de Recherche en Informatique”.

8.3. Academic Supervision

- Philippe de Groote has been supervising the thesis work of Sylvain Salvati.
- Guy Perrier is supervising the thesis work of Joseph Le Roux.
- Jean-Yves Marion is supervising the thesis work of Romain Péchoux from September 2004.
- Jean-Yves Marion and Simona Ronchi Della Rocca (Turino university) are co-supervising the thesis work of Marco Gaboardi.
- Jean-Yves Marion and Guillaume Bonfante are co-supervising the thesis work of Mathieu Kaczmarek.
- Sylvain Pogodalla advised three third year students at École des Mines (Amal Laouaj, Guillaume Princelle et Lisa Rouhban) for a two month internship devoted to studying french tokenization.
- Bruno Guillaume advised two second year students at École des Mines (Damien Auricchio and Nelson Da Silva) for a three month internship devoted to factorizing morphological informations of flexed forms and comparing morphological lexicons.
- Bruno Guillaume advised two second year students at École des Mines (Vincent Domange and Romain Jacquier) for a three month internship devoted to interfacing lexicons and anonymous grammars.

8.4. Thesis juries

- Philippe de Groote was jury member for Sylvain Salvati’s thesis, Nancy, June 13.
- Philippe de Groote was jury president for Benoît Crabbé’s thesis, Nancy, June 14.
- Philippe de Groote was referee and jury member for Hugo Herbelin’s HDR, Paris 11, December 7.
- Jean-Yves Marion was jury member for Clara Bertolissi’s thesis, Nancy October 28.
- Jean-Yves Marion was jury member for Julien Fondrevelle’s thesis, Nancy November 10.

8.5. Thesis defenses

- Sylvain Salvati defended his thesis on June 13, 2005 (jury: Éric de la Clergerie, Alexander Dikovsky, Philippe de Groote, Dale Miller, Glynn Morrill, Karl Tombre).

8.6. Participation to colloquia, seminars, invitations

- Bruno Guillaume and Guy Perrier attended the "Journée ATALA : Interface lexique-grammaire et lexiques syntaxiques et sémantiques" on March 12, in Paris. They presented a talk and a poster.
- Philippe de Groote, Bruno Guillaume, Sylvain Pogodalla and Sylvain Salvati attended the Demonat workshop in Nancy in April..
- Philippe de Groote attended the LACL'05 conference, in Bordeaux, April 28-30.
- Jean-Yves Marion gave an invited talk *Data tiering as a complexity tool which jumps from discrete to real computation* at the International Workshop "Computations on the continuum", June 2005, Lisbon.

9. Bibliography

Major publications by the team in recent years

[1] J. BESOMBES, J.-Y. MARION. *Apprentissage des langages réguliers d'arbres et applications*, in "Traitement automatique de langues", vol. 44, n^o 1, July 2003, p. 121–153.

[2] D. LEIVANT, J.-Y. MARION. *A characterization of alternating log time by ramified recurrence*, in "Theoretical Computer Science", vol. 236, n^o 1-2, 2000, p. 192–208.

[3] G. PERRIER. *Interaction Grammars*, in "CoLing 2000, Sarrebrücken, Germany", International Committee on Computational Linguisitcs, August 2000, p. 600–606.

[4] G. PERRIER. *La sémantique dans les grammaires d'interaction*, in "Traitement Automatique des Langues", vol. 45, n^o 3, 2004, p. 123–144.

[5] P. DE GROOTE. *Towards abstract categorial grammars*, in "Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Toulouse, France", July 2001, p. 148–155.

[6] P. DE GROOTE, F. LAMARCHE. *Classical Non Associative Lambek Calculus*, in "Studia Logica", vol. 71, n^o 3, August 2002, p. 355–388.

Doctoral dissertations and Habilitation theses

[7] S. SALVATI. *Problèmes de filtrage et problèmes d'analyse dans les grammaires catégorielles abstraites*, Ph. D. Thesis, Institut National Polytechnique de Lorraine, June 2005.

Articles in refereed journals and book chapters

[8] J. BESOMBES, J.-Y. MARION. *Learning tree languages from positive examples and membership queries*, in "Theoretical Computer Science", To appear, 2005.

[9] G. BONFANTE, M. KACZMAREK, J.-Y. MARION. *Abstract computer virology : a recursion theory approach*, in "Journal in Computer Virology", To appear, 2005.

[10] G. BONFANTE, J.-Y. MOYEN, J.-Y. MARION. *Quasi-interpretations, a way to control resources*, in "Theoretical Computer Science", To appear, 2005.

[11] O. BOURNEZ, C. FELIPE, P. JACOBÉ DE NAUROIS, J.-Y. MARION. *Implicit complexity over an arbitrary structure : Sequential and parallel polynomial time*, in "Journal of Logic and Computation", vol. 15, February 2005, p. 41–58.

[12] C. GARDENT, B. GUILLAUME, G. PERRIER, I. FALK. *Extracting subcategorisation information from Maurice Gross' grammar lexicon*, in "Archives of Control Sciences", vol. 15, n^o 3, 2005, p. 253-264, http://hal.inria.fr/inria-00000985/en/.

[13] F. LAMARCHE. *On the algebra of structural contexts*, in "Mathematical Structures in Computer Science", To appear, 2005.

[14] P. DE GROOTE. *Language and Grammar*, Casadio,C. and Scott,P.J. and Seely, R.A.G. editors, chap. On the expressive power of the Lambek calculus extended with a structural modality, CSLI Lecture Notes, 2005, p. 95–111.

[15] P. DE GROOTE, S. POGODALLA. *On the expressive power of Abstract Categorial Grammars: Representing context-free formalisms*, in "Journal of Logic, Language and Information", vol. 13, n^o 4, December 2004, p. 421–438.

Publications in Conferences and Workshops

[16] G. BONFANTE, M. KACZMAREK, J.-Y. MARION. *Abstract Detection of Computer Viruses*, in "Appsem II, Munich", 2005.

[17] G. BONFANTE, M. KACZMAREK, J.-Y. MARION. *Toward an abstract computer virology*, in "ICTAC05 - International Colloquium on Theoretical Aspects of Computing", 2005.

[18] G. BONFANTE, J.-Y. MARION, J.-Y. MOYEN, R. PÉCHOUX. *Synthesis of Quasi-Interpretations*, in "Seventh International Workshop on Logic and Computational Complexity, Chicago", 2005.

[19] G. BONFANTE, J.-Y. MOYEN, J.-Y. MARION. *Quasi-interpretations and Small Space Bounds*, in "16th International Conference on Rewriting Techniques and Applications, RTA'2005", Lecture Notes in Computer Science, vol. 3467, 2005, p. 150–164.

[20] O. BOURNEZ, F. CUCKER, P. JACOBÉ DE NAUROIS, J.-Y. MARION. *Logical Characterizations of $P_{\mathcal{K}}$ and $NP_{\mathcal{K}}$ Over an Arbitrary Structure K*, in "CIE 2005:New Computational Paradigms, Amsterdam", 2005.

[21] D. DUCHIER, J. LE ROUX, Y. PARMENTIER. *XMG: un Compilateur de Métagrammaire Extensible*, in "Conference Traitement Automatique des Langues Naturelles (TALN'2005), Dourdan", 2005.

[22] P. FONTAINE, K. GUPTA, J.-Y. MARION, L. NIETO, S. MERZ, A. TIU. *Towards a Combination of Heterogeneous Deductive Tools for System Verification: A Case Study on Clock Synchronization*, in "Appsem II, Munich", 2005.

[23] C. GARDENT, B. GUILLAUME, G. PERRIER, I. FALK. *Maurice Gross' Grammar Lexicon and Natural Language Processing*, in "2nd Language and Technology Conference - L&T'05", 2005.

[24] F. LAMARCHE, L. STRASSBURGER. *Constructing Free Boolean Categories*, in "Twentieth Annual IEEE Symposium on Logic in Computer Science - LICS 2005, Chicago, USA", P. PANANGADEN (editor). , IEEE Computer Society Press, Jun 2005, p. 209–218.

[25] F. LAMARCHE, L. STRASSBURGER. *Naming Proofs in Classical Propositional Logic*, in "7th International Conference on Typed Lambda Calculi and Applications - TLCA 2005, Nara, Japan", P. URZYCZYN (editor). , Lecture Notes in Computer Science, vol. 3461, Springer-Verlag, Apr 2005, p. 246-261.

[26] J.-Y. MARION, J.-Y. MOYEN. *Termination and Non Size Increasingness of assembly programs*, in "Appsem II, Munich", 2005.

Internal Reports

[27] S. POGODALLA, C. RETORÉ. *Handsome Non-Commutative Proof-Nets: perfect matchings, series-parallel orders and Hamiltonian circuits*, Research Report, n^o RR-5409, INRIA, December 2004, http://www.inria.fr/rrrt/rr-5409.html.

Miscellaneous

[28] C. GARDENT, B. GUILLAUME, I. FALK, G. PERRIER. *Le lexique-grammaire de M. Gross et le traitement automatique des langues*, Communication à la jounée ATALA : Interface lexique-grammaire et lexiques syntaxiques et sémantiques, 2005, http://www.atala.org/article.php3?id_article=240.

[29] B. GUILLAUME, G. PERRIER. *Interface lexique-grammaire via des structures de traits*, Communication à la journée ATALA : Interface lexique-grammaire et lexiques syntaxiques et sémantiques, 2005, http://www.atala.org/article.php3?id_article=240.

Bibliography in notes

[30] Y. BAR-HILLEL. *A quasi-arithmetical notation for syntactic description*, in "Language", vol. 29, 1950, p. 47-58.

[31] B. CRABBÉ. *Computational representation of strongly lexicalised syntactic formalisms: application to Tree Adjoining Grammars*, thèse de doctorat, university Nancy2, 2005.

[32] H. CURRY. *Foundations of mathematical logic*, Dover Publications, 1977.

[33] G. GENTZEN. *Recherches sur la déduction logique (Untersuchungen über das logische schließen)*, Traduction et commentaire par R. Feys et J. Ladrière, Presses Universitaires de France, 1955.

[34] J.-Y. GIRARD. *Linear Logic*, in "Theoretical Computer Science", vol. 50, 1987, p. 1-102.

[35] T. G. GRIFFIN. *A formulae-as-types notion of control*, in "Conference record of the seventeenth annual ACM symposium on Principles of Programming Languages", 1990, p. 47-58.

[36] J. LAMBEK. *The mathematics of sentence structure*, in "Amer. Math. Monthly", vol. 65, 1958, p. 154-170.

[37] J. LAMBEK. *On the calculus of syntactic types*, in "Studies of Language and its Mathematical Aspects, Providence", Proc. of the 12th Symp. Appl. Math., 1961, p. 166-178.

[38] S. LEHMANN, S. OEPEN, S. REGNIER-PROST, K. NETTER, V. LUX, J. KLEIN, K. FALKEDAL, F. FOUVRY, D. ESTIVAL, E. DAUPHIN, H. COMPAGNION, J. BAUR, L. BALKAN, D. ARNOLD. TSNLP — *Test Suites for Natural Language Processing*, in "Proceedings of COLING 1996,Kopenhagen", 1996.

[39] G. PERRIER. *Intuitionistic Multiplicative Proof Nets as Models of Directed Acyclic Graph Descriptions*, in "8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning - LPAR 2001, Havana, Cuba", A. V. ROBERT NIEUWENHUIS (editor). , Lecture Notes in Artificial Intelligence, vol. 2250, Springer, Dec 2001, p. 233-248.

[40] G. PERRIER. *Descriptions d'arbres avec polarités : les Grammaires d'Interaction*, in "9ième Conférence annuelle sur le Traitement Automatique des Langues Naturelles - TALN'02, Nancy, France", Jun 2002, http://www.loria.fr/publications/2002/A02-R-123/A02-R-123.ps.

[41] K. VIJAY-SHANKER. *Using Description of Trees in a Tree Adjoining Grammar*, in "Computational Linguistics", vol. 18, n^o 4, 1992, p. 481-517.