INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team calligramme

Linear Logic, Proof Nets and Categorial Grammars

Lorraine

THEME 2A

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 [“vacataire permanente”, INRIA]

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

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

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

Staff member Université Nancy 2
Guy Perrier [Lecturer]

Visiting scientist
Denys Duchier [INRIA Visiting scientist]

Post-doctoral fellow
Lutz Straßburger [INRIA postdoctoral fellow (since September 1)]

Ph. D. Students
Jérôme Besombes [Université Henri Poincaré, teaching assistant at ESIAL]
Emmanuel Hainry [ENS fellow, joint thesis with project-team Protheo since Sept. 1]
Joseph Leroux [MESR fellow, since October 1]
Jean-Yves Moyen [ENS fellow]
Paulin Jacobé de Naurois [ENS fellow, joint thesis with project-team Protheo and the City University of Honk Kong (F. Cucker)]
Sylvain Salvati [MESR fellow, INPL]

Undergraduate in Training
Nicolas Barth [École des Mines, June 15–September 15]

2. Overall Objectives

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

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 [42] results from a finer analysis of the part played by structural rules in Gentzen's sequent calculus [41]. 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.

The resulting system, called *rudimentary linear logic*

Table 1. rudimentary linear logic

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

presents many interesting properties. It is endowed with four logical connectors (two conjuctions 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 strenght 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 [33] to incorporate them as logical rules that are embodied in new connectors. This original idea is what gives linear logic all its subtelty 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 in 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 [13] [35].

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

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

Participants: Guillaume Bonfante, Philippe de Groote, Bruno Guillaume, François Lamarche, Guy Perrier, Sylvain Pogodalla, Lutz Straßburger.

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 [43]

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 and Dependency Grammars

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

Participants: Jérôme Besombes, Guillaume Bonfante, Denys Duchier, Philippe de Groote, Bruno Guillaume, François Lamarche, Joseph Leroux, Jean-Yves Marion, Guy Perrier, Sylvain Pogodalla, Sylvain Salvati, Lutz Straßburger.

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 ($/$) and the backslash ($\setminus$) are interpreted respectively as fraction pairings that simplify to the right and to the left, respectively. 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 [46][47]. 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 [51].

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 [60].
- contrary to TAG where tree fragments can only be inserted, Interaction Grammars admit models where the interpretations of description fragments may overlap.

Another grammatical framework which embraces both the notion of resource sensitivity and the interpretational perspective of model theory is dependency grammar.

Dependency grammar is predicated on the notion of an asymmetrical relation of (syntactic or semantic) dependency. This analytical idea has a very long history dating back at least to Panini (450 BC) and the Ancient logicians and philosophers, and made its way into european medieval linguistics under the spreading influence of the Arabic linguistic tradition. The modern notion of dependency grammar is usually attributed to Tesnière [59] and has been further developed in such stratificational formalizations as Functional Generative Description (FGD) [58] and Meaning-Text Theory [48].

The main formal notions of dependency grammar that reflect and embody its sensitivity to resources are subcategorization (sensitivity to syntactic resources) and valency (sensitivity to semantic resources). It should be noted that these core DG concepts of head/dependent asymmetry and subcategorization/valency have been adopted by other grammatical formalisms. In the categorial grammar tradition, these notions appear respectively as directional functional application and categorial types. In HPSG, they give rise to the notion of headed structures, head daughters, and SUBCAT lists.

Dependency grammar permits non-projective analyses, i.e. where branches may cross. For this reason, it holds a special appeal for languages with free or freer word-order than French or English, such as German, Russian, Czech... and is certainly one reason for the strong renewal of interest in DG in recent years.

Duchier [37] proposed a new formulation of DG with a model theoretic interpretation that has a natural reading as a concurrent constraint program. This approach, based on set constraints, offers an efficient treatment of both lexical and structural ambiguity, and produces parsers that take full effective advantage of constraint propagation to achieve very good practical performance.

Extending this approach, Duchier and Debusmann [36] proposed Topological Dependency Grammar (TDG) as a means to equip DG with a theory of word-order. TDG adopts the ID/LP[1] perspective and explains word-order phenomena as arising through the interactions of a non-ordered ID tree of syntactic dependencies and an ordered and projective LP tree of topological dependencies. They provided a detailed, yet simple, account of the challenging word-order phenomena in the verbal complex of German verb final sentences. This extension retains the computational advantages of model elimination through powerful constraint propagation.

3.4. Implicit Complexity of Computations

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

Participants: Guillaume Bonfante, Adam Cichon, Paulin Jacobé de Naurois, Jean-Yves Marion, Jean-Yves Moyen.

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

[1] Immediate Dependence / Linear Precedence

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 Intuitionisitc Linear Logic (which have very specific properties) in a model-theoretical framework [51] and it was at first designed for modelling the syntax of natural languages [52].

4.1.3. Dependency Grammars

Dependency grammar (DG) is a resource sensitive grammar formalism related to categorial grammar. An important appeal of DG is that it naturally permits non-projective analyses and is thus especially well suited for languages with free or freer word-order (e.g. Czech, German, Russian, etc...). However, free(r) word-order poses an especially thorny challenge for processing: all efficient parsing algorithms rely fundamentally on properties of adjacency and projectivity. Duchier [37] described a declarative formulation of valid DG analyses, and showed how effective model elimination using constraint programming technology could serve as a practical foundation for DG parsing.

Of course, not all languages have free(r) word-order, and even for those, word-order is not really entirely free. The next challenge for DG was to permit an account of word-order. While this had been attempted in the past, no proposal to date was especially appealing. Duchier and Debusmann [36] proposed Topological Dependency Grammar (TDG) which took inspiration in the topological field theory that is at the heart of traditional German descriptive syntax. In TDG, an analysis consists of two trees: a non-ordered ID tree of syntactic dependencies and an ordered and projective LP tree of topological dependencies. The ID tree is related to the LP tree through a process of emancipation. Thus, in TDG, macroscopic word-order phenomena are not modeled directly, rather they are seen as emerging from (1) simple lexical constraints and (2) from the mutually constraining interactions of the ID and LP trees.

4.1.4. Meta-grammars and lexical resources

Every lexicalized grammar formalism (tree adjoining grammar, categorial grammar, interaction grammar, dependency grammar) is faced with the problem of organizing and structuring the lexicon. On the one hand, there is the pragmatic consideration that the lexicon should be modularly organized so as to make it easier to develop, maintain and extend. On the other hand it should also be possible to express linguistic generalizations (e.g. passivisation schemata).

Marie-Hélène Candito proposed and developed a meta-grammatical approach for lexicalized TAGs (Tree Adjoining Grammars) that generated a lot of interest. Unfortunately, it also exhibited a number of infelicitous properties, in particular difficulties in managing named entities and implicit crossings.

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 Crystal and Mimosa project-teams.

5. Software

5.1. Leopar

LEOPAR (formerly known as AGIR) is a parser for natural languages which is based on the formalism of Interaction Grammars (IG) [52]. The first release was developed by G. Bonfante, B. Guillaume and G. Perrier, and G. Bonfante, B. Guillaume, G. Perrier and S. Pogodalla are working on the next version. It uses a parsing principle, called "electrostatic parsing" which is based on neutralizing opposite polarities. A positive polarity corresponds to an available linguistic constituent and a negative one to an expected constituent.

Parsing a sentence with an interaction grammar consists in first selecting a lexical entry for each of its words, then in merging all selected descriptions—tree descriptions à la Vijay-Shanker—into a unique one which represents a syntactic description of the sentence. The criterion for success is that this ultimate description is a neutral tree description. As IG are based on 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 indeterminism 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. They are used in the first, tagging phase of the analysis by the additional constraint of allowing only globally neutral selections. In the second, node-merging phase, polarities are used to cut off parsing branches whose trees contain too many uncancelled polarities.

LEOPAR's first release is available on the web at http://www.loria.fr/equipes/calligramme/leopar/ under the GNU General Public License. The current implementation is provided with a small grammar for French and a lexicon suited for this grammar. The grammar contains 31 descriptions. Despite its small size, it covers several non trivial linguistic phenomena (relative clauses, negation, pied-piping in the relatives...). The lexicon contains 78 entries.

5.2. XDG

The XDG software provides parsing and grammar development facilities for eXtendable Dependency Grammar. XDG is a meta-formalism that supports the declarative specification of typed multi-dimensional lexicalized DG formalisms. XDG provides the computational linguist with a simple and convenient way to assemble his own dependency grammar formalism declaratively. As a result of such a specification, he obtains for free (1) a concrete meta-grammar formalism appropriate to his design, and (2) a constraint-based parser and workbench.

XDG was written by Denys Duchier and Ralph Debusmann. The current version of the software is available at http://www.mozart-oz.org/mogul/info/debusmann/xdg.html and contains small sample grammars for German, English, Dutch and Czech.

5.3. Meta-grammar Workbench

The metagrammar workbench offers a metagrammar formalism, compiler, and graphical visualizer for lexicalized TAGs. It currently supports a syntactic dimension adapted to Benoît Crabbé's TAG descriptions and a semantic dimension adapted to the hole semantics used by Claire Gardent, and will be extended to support Guy Perrier's Interaction Grammars. The software was written by Denys Duchier and Yannick Parmentier.

5.4. Lexical Resources

As a first stab at building free appropriate lexical resources, Nicolas Barth, second year student at École des Mines, during a training stint in Calligramme, developped an application that computes the flexed forms of French verbs from their infinitive form and derivation template. As of now, about 300,000 flexed forms can be generated automatically, for 6,500 verbs. The program is accessible at http://www-int.loria.fr/~pogodall/litote/demos/verbes.html.

6. New Results

6.1. Proof Nets, Sequent Calculus and Typed Lambda Calculi

6.1.1. Explicit Substitutions

The λ_{ws}-calculus is a λ-calculus with explicit substitutions. This calculus is the first one of that growing family of formal systems that simultanenously provides step by step simulation of β-reduction, confluence on terms with metavariables and preservation of strong normalization of the usual λ-calculus. It is presented in [34] where these properties are proved in the untyped case; the normalization property for the typed case is left open in that paper. This open question was solved for the simply typed case in [54] with a translation of λ_{ws}-terms in proof nets of linear logic. In [20], R. David and B. Guillaume give an elementary proof of the normalization result in the simply typed case, and a proof of normalization of the λ_{ws}-terms in for second-order types; this latter result is entirely new. The two results use a common untyped lemma, and it is the difficult part of the proof; the two normalization proofs are then easy adaptations of the corresponding ones for the λ-calculus.

6.1.2. Linear λ-term Matching

In [24], Philippe de Groote and Sylvain Salvati have shown that even for simple cases pattern matching in the linear λ-calculus was NP-complete. More precisely, they studied the case where the indeterminates had order two and occured linearly in the equation. This case was expected to be polynomial because of its resemblance to linear matching with tree contexts; but surprisingly the little differences between the two problems were sufficient to make the first one NP-complete while the other was polynomial.

Furthermore, Sylvain Salvati has studied the case of pattern matching in the linear λ-calculus with additives. He proved that when the right member of the equation (the part without indeterminates) was in normal form the problem remained NP-complete but he also proved that when the right member was not in normal form, the problem became PSPACE-complete (this result is being written up).

6.1.3. Symbolic Geometry

For some time François Lamarche has been working on a unifying framework for the proof theory of resource-sensitive logics; the first realization in this program is the definition of the concepts of structad and fibrational theory [13], which adapts some algebraic concepts used in modern geometry (principally operads and Grothendieck fibrations) to the situation under study.

But several problems he encountered made him realize that this had to be pushed further: the "contraction-like" correctness criteria discovered by Puite [53] and his collaborator Moot [50] show clearly that the theory of rewriting has to be extended to structads, and that this poses some difficult problems, for example the control of confluence. This research was also spurred by the innovations brought about by the discovery of the

Calculus of Structures [44], which can be summarized (too) briefly by saying that it introduces the concept of deduction-as-rewriting. Finally there was the need to extend structads to include things like contraction and weakening, so they would not be limited to multiplicative logics.

He realized that not only was there the need for a very general concept of term-rewriting system, that would include terms, graphs, related higher dimensional objects (like derivations), with the possibility of adding equations between these ("rewriting modulo"), but that the many hints of an underlying geometric nature could (and therefore *had to*) be made manifest. This has led, first, to a scientific manifesto [27] where the general ideas are exposed in a non-technical way, along with their origins in 20th century geometry and their relevance to computer science. The first truly technical paper is [26], which gives a general axiomatic theory for these "symbolic spaces", and proves some basic theorems (like the embedding of more familiar categories like structads and algebraic theories into these). The main idea is to start with a base category whose objects are finite sets, and whose elements are to be seen as unsorted variables, but where a morphism is a particular kind of relation that expresses what happens to variables when a an operation (like a substitution, a contraction, a binding...) is done on a term. The relationship to the theory of complexes in algebraic topology is stressed, and some general concepts of geometry like change of base and fibration—in particular, a new, more rigid version of the concept of Grothendiec fibration is introduced—are used extensively, giving a reusable "toolkit", equipped with a rich set of operations for stating and solving problems.

6.2. Categorial Grammars and Dependency Grammars

6.2.1. Interaction Grammars

Since G. Perrier has shown the interest of Interaction Grammars (IGs) in [52], G. Bonfante, B. Guillaume and G. Perrier have been studying the parsing issue associated to these grammars. Their purpose is to develop tools that provides a practical framework for the testing and the use of Interaction Grammars.

First of all, the theoretical problem of parsing IGs is NP-complete. So, an important problem was to find techniques that prune as early as possible the branches of the search tree that are bound to fail. They developed a new parsing process called "electrostatic parsing".

Electrostatic parsing is based on the notion of polarity, one of the defining features of Interaction Grammars. In [10], they have shown that these polarities—which were independently motivated by purely linguistic considerations—provide the basis for a new approach to parsing. First, they are used for reducing the number of potential sequences of elementary syntactic structures that are produced when assigning to each word a set of potential syntactic descriptions by means of a lexicon. Then, with the notion of feature cancellation, polarities play a central role in the control of the parsing process. Finally, they are used in heuristics which are inspired by psycholinguistic considerations for increasing parsing efficiency.

For the first step, the tagging level, the principle is to filter only taggings which are globally neutral. The use of automata allows them to manipulate sets of taggings and enhances the computational efficiency.

For the second step, node merging, they consider the fact that when reading a sentences, people use a very small amount of memory to parse it. From the point of view of IG, this can be translated as saying that one may force a predetermined bound on the number of active (polarized) constituents. With LEOPAR, the tests done so far show that this number is actually very small, at most 7 for "natural" sentences.

Initially, IGs were devoted to the syntax of natural languages but G. Perrier extended them to semantics by adding a new level to the syntactic one [23]. For this level, Directed Acyclic Graph descriptions are used to represent underspecified logical forms, and the same mechanism—feature cancellation—as for the syntactic level is used for composing the semantic representation of utterances. Synchronization between the syntactic and the semantic levels is performed by a partial function from the nodes of syntactic descriptions to the nodes of the associated semantic descriptions.

6.2.2. eXtendable Dependency Grammars (XDGs)

With Topological Dependency Grammar (TDG) [36] we had shown that difficult macroscopic word-order phenomena could be modeled as emerging from the interactions of a non-ordered ID tree of syntactic

dependencies and an ordered and projective LP tree topological dependencies related to each other through a process of emancipation.

Next, in order to tackle semantics, and in addition to the ID and LP trees, it became necessary to add a so-called tectogrammatical dimension to represent the predicate/argument structure. This suggested that we should embrace a more general multi-dimensional architecture where macroscopic linguistic phenomena arise from (1) simple lexical constraints simultaneouly constraining all dimensions, and (2) from principles stipulating how dimensions should be related. Furthermore, it should be possible to easily add new dimensions and new principles relating them.

This is what eXtendable Dependency Grammar (XDG) is about. XDG is a meta-formalism that supports the declarative specification of typed multi-dimensional lexicalized DG formalisms. XDG provides the computational linguist with a simple and convenient way to assemble his own dependency grammar formalism declaratively. As a result of such a specification, he obtains for free (1) a concrete meta-grammar formalism appropriate to his design, and (2) a constraint-based parser and workbench.

Ralph Debusmann is implementing XDG as part of his doctoral work. XDG has already been used to create a few DG instances: first, TDG was recreated, but this time we were able to take advantage of the meta-grammatical facilities provided by XDG to simplify and linguistically improve the lexicon. Second, we created STDG (Semantic Topological Dependency Grammar) which extends TDG with a syntax/semantics interface using, as additional dimensions, a tectogrammatical DAG for the predicate/argument structure, and a derivation tree for assembling the semantic logical form.

Kruijff and Duchier [22] described how to integrate information structure into TDG, and presented a constraint-based approach to modelling information structure and the various means to realize it, focusing on (possibly simultaneous use of) word-order and tune.

Meanwhile, XDG has become one of the components in a new project dedicated to the development of a preference architecture for processing. The goal is to make it possible to combine multiple sources of information to guide processing. There are several research groups involved in this effort. One (CHORUS) is more particularly interested in semantic preferences. Another (NEGRA) is more concerned with wide-coverage grammars automatically induced from corpora. For the latter, the integration of multiple sources of information is especially critical. XDG's constraint-based approach make it particularly well-suited for integrating fragmentary bits of information and drawing the most out of it by propagation.

6.2.3. Abstract Categorial Grammars (ACGs)

The study and the development of ACGs has two main current topics: their expressive power and computational complexity for parsing. Philippe de Groote and Sylvain Pogodalla proved that ACGs can encode m-LCFRS (linear context free rewriting systems) [21]. This enables ACGs to cover important (w.r.t. natural language modeling) classes of languages such as the ones generated by—because of the weak equivalence that holds among them—multicomponent tree adjoining grammars (MCTAGs) [61], multiple context-free grammars (MCFG) [57] or minimal grammars (MG) [49].

The result obtained in [24] has a deep impact on the complexity of the analysis of ACGs. Indeed, for the majority of lexicalized grammars (grammars without empty entries) it implies that the analysis is NP-complete. The new result obtained for the matching linear λ-calculus with additives makes it possible to extend the formalism of ACGs to the additives ; this result has also some impact on the complexity of the problem of analysis in that case: it remains NP-complete in the case of lexicalized grammars.

6.2.4. Grammar Learning

For learning regular tree languages as a model of natural language acquisition, Jérome Besombes and jean-Yves Marion studied a paradigm of exact learning from positive data and interactions with an Oracle. The Oracle is a person on contact with the child, who knows the language (a parent) and is able to reply to membership queries to the target language submitted by the apprentice. A polynomial algorithm based on this paradigm has been constructed ($Altex$) and its correctness proved.

Jérome Besombes and Jean-Yves Marion defined a class of categorial grammars in the spirit of Angluin [29] and Sakakibara [56]: the reversible categorial grammar. This class is identifiable from positive structured examples (the FA-structures used by Kanazawa [45]). An original algorithm ($Alfa$) has been developped and its correcteness has been showed. The main benefit of this class is that, in one hand, several types can be associated to the same word independently to any constant k (if two such types are not differing in only one atomic sub-type), that allows to learn linguistic ambiguities (verbs with transitive and non transitive forms, homonymies,...)and in the other hand, the learning algorithm is quadratic.

6.2.5. Meta-grammars

Denys Duchier organized a workgroup on the topic of metagrammars involving both Calligramme and Langue et Dialogue: http://www.loria.fr/~duchier/MetaGrammars/index.html

This resulted in a collaboration between the two projects to develop a new meta-grammar approach: Benoît Crabbé presented a new way to decompose syntactic paraphrases into tree description fragments decorated with resource annotations, and Claire Gardent described the requirements of the syntax/semantics interface for her application to hole semantics. Denys Duchier designed a new (multi-dimensional) formalism combining both inheritance and disjunction and where paraphrases are assembled by a constraint-based solver inspired by the dominance constraint solver of Duchier and Gardent [38]. The implementation work was done jointly with Yannick Parmentier and currently supports a syntactic dimension adapted to Crabbé's TAG descriptions and a semantic dimension adapted to hole semantics. The syntax/semantics interface obtains naturally from composition and coindexation constraints.

We are now working on extending this approach for Guy Perrier's Interaction Grammars. The goal is to arrive at a meta-formalism which can be suitably instantiated for various grammar formalisms, in particular for TAGs and Interaction Grammars.

6.2.6. Dominance Constraints

Dominance constraints are logical description of trees. Satisfiability of dominance constraint was proved NP-complete, but Mehlhorn etal. [2001] distinguished the subclass of "normal dominance constraints" and presented an algorithm taking time $O(n^4)$ per solved form. Thiel [2004] improved this result to $O(n^3)$ by faster satisfiability testing. We now exhibit a novel graph algorithm taking $O(n^2)$ per solved-form and also capable of handling the larger language of *weakly normal dominance constraints* [16] (Bodirsky, Duchier, Miele and Niehren [2004])

This new result has considerable practical import because dominance constraints are widely used in computational linguistics, e.g. in underspecified semantics [39][32][31], underspecified discourse [40], parsing with tree adjoining grammar [55]. In project Calligramme, they are especially relevant to parsing with Interaction Grammars, to the CLLS-based semantic component of XDG, and to the Meta-grammar workbench which all use them in an essential capacity.

6.3. Implicit Complexity of computation

Jean-Yves Marion and Jean-Yves Moyen have used Petri nets in order to analyse the termination of assembler-like programs. They translate the program into a Petri net and show that the termination of the Petri net, which can be decided via linear programming techniques, implies the program's termination. This method allows to prove in a fully automated way the termination of some not-so-trivial programs such as Euclid's algorithm for computing the greatest common divisor or the Quicksort (and thus, potentially, all algorithms of the Divide and Conquer variety).

Moreover, this method can also prove the non-size-increasingness property of some algorithms, that is, show that these program will not use more memory than initially allocated and could be written in C without using the `malloc` instruction.

This method of analysis seems to have some other good properties, such as offering the possibility of doing some basic program transformations (e.g., Deforestation) at the same time as proving the termination of both

the initial and the transformed programs. It is still under study and the main research direction is a way to add a few features to the fundamental language, in particular a real stack to store intermediate values.

8. Other Grants and Activities

8.1. 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, through Denys Duchier in particular, is a partner in the series of joint scienfic meetings organized through the Lorraine-Saarland partnership;
- Calligramme is part of the "Qualité et sûreté des Logiciels (QSL) theme of the "contrat de plan État-Région". As a matter of fact the fact the head of the whole QSL theme is Jean-Yves Marion;
- Jean-Yves Marion is head of the "Synthèse de code pour robots mobiles" (Sycomor) project, granted by the scientific council of the INPL;

8.2. National Actions

8.2.1. 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.

Web page at http://www.loria.fr/~guillaum/demonat

8.2.2. 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 Marseilles, and the co-ordinator on Calligramme's side is Jean-Yves Marion.

Web page at http://www.cmi.univ-mrs.fr/~amadio/Criss/criss.html

8.2.3. Action Concertée Incitative (ACI) Géocal

This "nouvelles interfaces des mathématiques" ACI 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 Marseilles.

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

8.2.4. Action Spécifique "Topologie Algébrique"

This is a smaller CNRS action whose full title is "Topologie algébrique pour l'étude des structures de calcul et notamment de la concurrence". It is headed by Éric Goubault at the Commissariat de l'Énergie Atomique, and many of its members are also part of the previous action.

Web page at http://www.di.ens.fr/~goubault/as.html

8.2.5. Réseau National des Technologies Logiciells (RNTL)

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

8.3. European Actions

- Calligramme is one half of a joint France-Germany (Procope) cooperation entitled "Structures and Deductions" with the Technische Universität Dresden, Germany.

- Calligramme is involved in the IST-2001-38957 european project, in the FET-Open program, entitled "Applied Semantics II".
- Calligramme is involved in the european network CoLogNET (Computational Logic Network) on the themes: logic methodology and foundational tools, logic and natural language processing.

8.4. Visits and invitation of researchers

- Simonetta Ronchi della Rocca of the university of Turin spent a week in April, being the guest of both the Calligramme and Miró teams.
- Lutz Straßburger, PhD student, University of Dresden, visited the group for one week in January and gave a talk on "Linear Logic and Noncommutativity in the Calculus of Structures";
- Alessio Guglielmi, Charles Stewart and Phiniki Stouppa, University of Dresden, visited the group for one week in October and gave talks on "A New Proof Theory with Deep Inference and Symmetry", "Partial Sharing Diagrams", and "Modal Logics in the Calculus of Structures", respectively.
- Glyn Morrill, of the Universitat Politècnica de Catalunya, made a short visit on December 12.

9. Dissemination

9.1. Activism within the scientific community

- Guillaume Bonfante is member of the hiring committee, section 27, of the INPL, since April 2003;
- Guillaume Bonfante was elected to the scientific council of the INPL in July 2003;
- Adam Cichon is substitute member of the hiring committee, section 27, of the UHO–Nancy 2;
- Adam Cichon was elected member of the "Conseil National des Universités" (CNU), section 27;
- Jean-Yves Marion is member of the steering committee of the International workshop on Implicit Computational Complexity (ICC).
- Philippe de Groote is:
  - elected permanent member of INRIA's evaluation board,
  - president of the INRIA-Lorraine audience commission for Junior Researchers (2nd class),
  - member of the INRIA admisssion jury for Junior Researchers (2nd class) and Senior Researchers (2nd class),
  - vice president of the LORIA Projects Committee,
  - member of the LORIA management board,
  - member of the scientific orientation council of the LORIA,
  - named member of the LORIA laboratory council,
  - permanent member of the hiring committe of the INPL (section 27),
  - member of the program committee of ESSLI-04,
  - member of the proghram committe of the 8th Conference on Formal Grammar (Vienna, Austria, August 16–17 2003),
  - member of the editorial board of *Papers in Formal Linguistics and Logic*, Bulzoni, Roma,
  - member of the editorial board of *Cahiers du Centre de Logique*, Academia-Bruylant, Louvain-la-Neuve,

- Since 2002 François Lamarche has been organizing the weekly Calligramme seminar. A particular event in 2003 was the organization of a "structures week", between the 6th and 10th of October.
  Web page: http://www.loria.fr/~lamarche/seminars.html
- François Lamarche is member of the board of the Département de Formation Doctorale of the Computer Science section of the Doctoral School.
- François Lamarche is head of the Committee of Grants and Fellowships section of the Projects Committee of the LORIA.
- Jean-Yves Marion is member of the hiring committee of Ecole des Mines (Professors and Lecturers), section 27, since February 2002.
- Jean-Yves Marion was elected to the scientific council of INPL in July 2003.
- Jean-Yves Marion organized the GEM-Stic Seminar on the safety and quality of software systems Jan 23 2003, Ecole des Mines de Nancy.
  Web page: http://www.loria.fr/~marionjy/GemStic230103/gemstic230103.html
- Jean-Yves Marion initiated and organized the monthly "Journées QSL" http://qsl.loria.fr
- Guy Perrier organized the *International Workshop on Parsing Technologies (IWPT 2003)*, which took place in Nancy from April 22 to 24. This event, which is held every two years, is the main international workshop on the parsing of natural languages and it was the first time that it was held in France.
- Guy Perrier was a member of the Program Comittee of the Lorraine-Saarland Workshop on Propects and Advances in the Syntax/Semantics Interface, which was held in Nancy on the 20th and 21st of October.
- Denys Duchier organized *Prospects and Advances in the Syntax/Semantics Interface* (October 20–21, 2003, Nancy, http://www.loria.fr/~duchier/Lorraine-Saarland/). This was the 6th workshop in the Lorraine-Saarland Workshop Series, a joint initiative of LORIA (Nancy) and MPI (Saarbrücken) whose primary purpose is to foster interaction and collaborative research between Lorraine and Saarland. This 2-days event attracted participants from France, Germany and Holland.
- Denys Duchier presented the metagrammar workbench prototype at the Lorraine-Saarland Workshop on Prospects and Advances in the Syntax/Semantics Interface, October 20–21, 2003, Nancy.
- Denys Duchier and Ralph Debusmann presented the XDG workbench at the Lorraine-Saarland Workshop on Prospects and Advances in the Syntax/Semantics Interface, October 20–21, 2003, Nancy.

9.2. Teaching

- Adam Cichon, with Claude Kirchner, is in charge of the DEA course on "logique et démonstration automatique";
- Adam Cichon was in charge of the DESS "Compétences Complémentaires" course on logic and artificial intelligence;
- Adam Cichon heads the "licence informatique";
- Guy Perrier is in charge of the organization of the course on *algorithmics for the parsing of natural languages*, which he is teaching with Bertrand Gaiffe in the computer science DEA of Nancy.
- Jean-Yves Marion heads the option "Maîtrise d'oeuvres et Maîtrise d'ouvrages (MOSI)" of the computer science department of ENSMN.
- Jean-Yves Marion is also in charge of the DEA course on complexity.
- Denys Duchier was invited by Robin Cooper and Tobjörn Lager to give a series of lectures at the Graduate School of Language Technology (GSLT) in Göteborg, Sweden (March 2003). Topics covered were data management and concurrency in Oz, then constraint programming and application to natural language processing.

9.3. Academic Supervision

- Guillaume Bonfante and Sylvain Pogodalla advised a second year student at École des Mines Student (Nicolas Barth) for a two month internship devoted to building flexed forms for 6,500 french verbs.
- Philippe de Groote is supervising the thesis work of Sylvain Salvati.
- Jean-Yves Marion and Guy Perrier are co-supervising the thesis work of J. Leroux.
- Jean-Yves Marion and Olivier Bournez (Project-team Protheo) are co-supervising the thesis work of Paulin Jacobé du Naurois and Emmanuel Hainry.
- Jean-Yves Marion supervised the thesis work of Jérôme Besombes and Jean-Yves Moyen.
- Denys Duchier is supervising Ralph Debusmann's Ph.D. thesis (Saarbrücken). Ralph is developing the new meta-formalism called eXtensible Dependency Grammar (XDG), with special attention to the syntax/semantics interface.

9.4. Thesis juries

We should note that three members of Calligramme saw jury duties from the other end: Jérôme Besombes defended his thesis on November 14 (with J.-Y. Marion and A. Cichon in the jury), Guy Perrier defended his Habilitation on November 12, and Jean-Yves Moyen defended his thesis on December 17 (with J.-Y. Marion in the jury).

- Jean-Yves Marion was external referee for Y. Lenir, Université de Rennes, December 15.
- François Lamarche was external referee for Lutz Straßsburger's thesis (TU Dresden, July 24).
- François Lamarche was internal referee and jury member for Fairouz Chakkour's thesis (Université Henri Poincaré, December 8).

9.5. Thesis defenses

- Jérôme Besombes defended his thesis on November 14 2003 (jury: A. Cichon, F. Denis, R. Gilleron, C. de la Higuera, M. Margenstern, J.-Y. Marion).
- Jean-Yves Moyen defended his thesis on December 17 2003 (jury: R. Amadio, B. Gaujal, N. Jones, C. Kirchner, J.-Y. Marion, J. Souquières).
- Guy Perrier defended his "Habilitation" thesis on November 12 2003 (jury: A. Abeillé, P. Blache, P. Blackburn, A. Dikovsky, G. Huet, J. Souquières, M. Steedman).

9.6. Participation to colloquia, seminars, invitations

- Seven members of Calligramme (J. Besombes, Ph. de Groote, F. Lamarche, J.-Y. Marion, G. Perrier, S. Pogodalla, S. Salvati) attended the GRACQ worshop at the LABRI, Bordeaux, March 17–18 2003.
- Bruno Guillaume attended the CSL03 conference (Vienna, Austria, 25th-30th of August). He presented the paper [20];
- Bruno Guillaume demonstrated the LEOPAR prototype at the Lorraine-Saarland Workshop on Prospects and Advances in the Syntax/Semantics Interface;
- Bruno Guillaume and Philippe de Groote visited the "Logique" group of "Université de Savoie" in Chambéry (February 19 and 20);
- Denys Duchier and Guy Perrier attended the Metagrammar reunion organized by Eric de la Clergerie (Université Paris VII, Septembre 18).

- Philippe de Groote and Sylvain Salvati attended the 14th Internatinal Conference on Rewriting Techniques and Applications (RTA'03), Valencia, Spain, June 9–11, 2003. They presented [24].
- François Lamarche attended the following meetings and seminars, and gave a talk at every one of them:
  - The 78th edition of the Peripatetic Seminar on Sheaves and Logic (Strasbourg, February 15–16).
  - The Séminaire CATIA (Université Montpellier II, April 2–5).
  - The "Journées Squier et tout ça" (Université Paris VII, May 14–16).
  - The Fields Institute Workshop on Mathematical Linguistics, (University of Ottawa, June 18–19).
  - The 2003 European Meeting on Category Theory (Haute-Bodeux, Belgium, September 8–12.)
  - The "Topologie Algébrique" meeting (Paris, November 26–27).
- In addition to the papers [15][17][18] Jean-Yves Marion gave the following invited talks:
  - Workshop "AS Mobilités" March 10, title: Extracting Feasible Programs.
  - Workshop on Linear Logic, Verona, December 17–18, title: Extracting Feasible Programs.
- Jean-Yves Moyen gave a talk at the internal LACL (Laboratoire d'Algorithmique, Complexité et Logique) seminar in Créteil on November 24 (http://www.univ-paris12.fr/lacl/).
- Lutz Straßburger visited the Proof Theory Group in Dresden for two weeks in September in order to work with Alessio Guglielmi, Paola Bruscoli and Ozan Kahrmanogullari;
- François Lamarche, Sylvain Pogodalla and Lutz Straßburger attended the "Structures Workshop" at the University of Dresden, November 20–21;
- Guy Perrier gave a talk at the Lorraine-Saarland Workshop on Propects and Advances in the Syntax/Semantics Interface, which was held in Nancy on the 20th and 21st of October 2003; [23].
- Most of the members of the team attended the International Workshop on Parsing Techniques (IWPT03, http://iwpt03.loria.fr/user/www/Nancy_en.html) that was held in Nancy.
- Sylvain Pogodalla attended the North American Summer School in Logic, Language and Information (Bloomington, Indiana, June 17–21);
- Sylvain Pogodalla attended the Mathematics of Language conference (Bloomington, Indiana, June 19–22) and presented [21];
- Denys Duchier gave the following invited talk at the GSLT conference in Lökeberg, Sweden: *The Conspiracy Theory of Language*.

10. Bibliography

Major publications by the team in recent years

[1] P. DE GROOTE. *Towards abstract categorial grammars.* in « Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Toulouse, France », pages 148-155, Jul, 2001.

[2] P. DE GROOTE, F. LAMARCHE. *Classical Non Associative Lambek Calculus.* in « Studia Logica », number 3, volume 71, Aug, 2002, pages 355-388.

[3] D. LEIVANT, J.-Y. MARION. *A characterization of alternating log time by ramified recurrence.* in « Theoretical Computer Science », number 1-2, volume 236, 2000, pages 192-208.

[4] G. PERRIER. *Interaction Grammars.* in « CoLing 2000, Sarrebrück, Germany », International Committee on Computational Linguisitcs, Aug, 2000.

Doctoral dissertations and “Habilitation” theses

[5] J. BESOMBES. *Un modèle algorithmique de la généralisation de structures dans le processus d'acquisition du langage.* Ph. D. Thesis, UHP–Nancy 1, 2003.

[6] J.-Y. MOYEN. *Analyse de la complexité et transformation de programmes.* Ph. D. Thesis, Nancy 2, 2003.

[7] G. PERRIER. *Les grammaires d'interaction.* Habilitation Thesis, Nancy 2, 2003.

Articles in referred journals and book chapters

[8] E. ALTHAUS, D. DUCHIER, A. KOLLER, K. MEHLHORN, J. NIEHREN, S. THIEL. *An Efficient Graph Algorithm for Dominance Constraints.* in « Journal of Algorithms », number 1, volume 48, May, 2003, pages 194-219, Special Issue of SODA 2001..

[9] J. BESOMBES, J.-Y. MARION. *Apprentissage des langages réguliers d'arbres et applications.* in « Traitement automatique des langues », number 1, volume 44, Jul, 2003, pages 121-153.

[10] G. BONFANTE, B. GUILLAUME, G. PERRIER. *Analyse syntaxique électrostatique.* in « Traitement Automatique des Langues (TAL) », number 3, volume 44, Dec, 2003, http://www.loria.fr/publications/2003/A03-R-245/A03-R-245.ps.

[11] D. DUCHIER. *Configuration of Labeled Trees under Lexicalized Constraints and Principles.* in « Research on Language and Computation », number 3-4, volume 1, Sep, 2003, pages 307-336.

[12] D. DUCHIER. *Dominance Constraints With Boolean Connectives : A Model-Eliminative Treatment.* in « Journal of Theoretical Computer Science (TCS) », number 2, volume 293, Feb, 2003, pages 321-343.

[13] F. LAMARCHE. *On the Algebra of Structural Contexts.* in « Mathematical Structures in Computer Science », Sep, 2003.

[14] J.-Y. MARION. *Analysing the implicit complexity of programs.* in « Information and Computation », number 1, volume 183, Mar, 2003, pages 2-18, International Workshop on Implicit Computational Complexity - ICC'99.

Publications in Conferences and Workshops

[15] J. BESOMBES, J.-Y. MARION. *Apprentissage des langages réguliers d'arbres à l'aide d'un expert.* in « Conférence d'Apprentissage - CAP'2003, Laval, France », Jul, 2003.

[16] M. BODIRSKY, D. DUCHIER, J. NIEHREN, S. MIELE. *A New Algorithm For Normal Dominance Constraints.* in « ACM-SIAM Symposium on Discrete Algorithms - SODA'2003, New Orleans, LA, USA », Jan, 2004.

[17] O. BOURNEZ, F. CUCKER, P. JACOBÉ DE NAUROIS, J.-Y. MARION. *Computability over an Arbitrary Structure. Sequential and Parallel Polynomial Time.* in « Foundations of Software Science and Computation Structures - FOSSACS'03, Warsaw, Poland », series Lecture Notes in Computer Science, volume 2620, Springer, A. D. GORDON, editor, pages 185-199, Apr, 2003.

[18] O. BOURNEZ, F. CUCKER, P. JACOBÉ DE NAUROIS, J.-Y. MARION. *Safe Recursion Over an Arbitrary Structure : PAR, PH and DPH.* in « Fifth International Workshop on Implicit Computational Complexity - ICC'2003, Ottawa, Canada », series Electronic Notes in Computer Science, number 1, volume 90, Jun, 2003.

[19] C. BRUN, M. DYMETMAN, É. FANCHON, S. LHOMME, S. POGODALLA. *Semantically-based text authoring and the concurrent documentation of experimental protocols.* in « ACM Symposium on Document Engineering - DocEng 2003, Grenoble, France », Nov, 2003.

[20] R. DAVID, B. GUILLAUME. *Strong Normalization of the Typed lambda ws-calculus.* in « 17th International Workshop on Computer Science Logic 2003 - CSL'2003, Vienne, Autriche », series Lectures Notes in Computer Science, volume 2803, Springer, J. A. M. MATTHIAS BAAZ, editor, pages 155-168, Aug, 2003.

[21] P. DE GROOTE, S. POGODALLA. *m-Linear Context-Free Rewriting Systems as Abstract Categorial Grammars.* in « Mathematics of Language 8 (MOL 8), Bloomington, Indiana, États-Unis », R. T. O. ET J. ROGERS, editor, pages 71-80, Jun, 2003, http://www.loria.fr/publications/2003/A03-R-243/A03-R-243.ps.

[22] G.-J. KRUIJFF, D. DUCHIER. *Information Structure in Topological Dependency Grammar.* in « Proceedings of EACL03 », April, 2003.

[23] G. PERRIER. *Semantics for Interaction Grammars.* in « Lorraine-Saarland Workshop on Prospects and Advances in the Syntax/Semantics Interface, Nancy, France », Oct, 2003, http://www.loria.fr/publications/2003/A03-R-309/A03-R-309.ps.

[24] S. SALVATI, P. DE GROOTE. *On the complexity of higher-order matching in the linear λ-calculus.* in « International Conference on Rewriting Techniques and Applications - RTA'2003, Valencia, Spain », series Lecture notes in Computer Science, volume 2706, R. NIEUWENHUIS, editor, pages 234-245, Jun, 2003.

Internal Reports

[25] R. DEBUSMANN, D. DUCHIER. *A Meta-Grammatical Framework for Dependency Grammar.* Rapport de recherche, Feb, 2003.

[26] F. LAMARCHE. *On the Geometric Nature of Symbolic Objects.* Technical report, Loria, 2003.

[27] F. LAMARCHE. *Vers une Géométrie Symbolique.* Technical report, Loria, Sept., 2003.

[28] J.-Y. MARION, J.-Y. MOYEN. *Termination and resource analysis of assembly programs by Petri nets.* Rapport de recherche, Nov, 2003.

Bibliography in notes

[29] D. ANGLUIN. *Inference of reversible languages.* in « Journal of the ACM », volume 29, 1982, pages 741-765.

[30] Y. BAR-HILLEL. *A quasi-arithmetical notation for syntactic description.* in « Language », volume 29, 1950, pages 47-58.

[31] J. BOS. *Predicate Logic Unplugged.* in « Proceedings of the 10th Amsterdam Colloquium », pages 133–143, 1996.

[32] A. COPESTAKE, D. FLICKINGER, I. SAG, C. POLLARD. *Minimal Recursion Semantics: An Introduction.* CSLI Draft, September, 1999.

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

[34] R. DAVID, B. GUILLAUME. *A λ-calculus with explicit weakening and explicit substitution.* in « Mathematical Structures for Computer Science », volume 11, 2001, pages 169–206.

[35] P. DE GROOTE, F. LAMARCHE. *Classical Non Associative Lambek Calculus.* in « Studia Logica », 2000.

[36] D. DUCHIER, R. DEBUSMANN. *Topological Dependency Trees: A Constraint-based Account of Linear Precedence.* in « 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001) », Toulouse, France, 9–11July, 2001.

[37] D. DUCHIER. *Axiomatizing Dependency Parsing Using Set Constraints.* in « Sixth Meeting on Mathematics of Language », pages 115–126, Orlando, Florida, July, 1999.

[38] D. DUCHIER, C. GARDENT. *A Constraint-Based Treatment of Descriptions.* in « Third International Workshop on Computational Semantics (IWCS-3) », H. BUNT, E. THIJSSE, editors, pages 71–85, Tilburg, NL, January, 1999.

[39] M. EGG, A. KOLLER, J. NIEHREN. *The Constraint Language for Lambda Structures.* in « Journal of Logic, Language, and Information », volume 10, 2001, pages 457-485.

[40] C. GARDENT, B. WEBBER. *Describing discourse semantics.* in « Proceedings of the 4th TAG+ Workshop », University of Pennsylvania, Philadelphia, 1998.

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

[42] J.-Y. GIRARD. *Linear Logic.* in « Theoretical Computer Science », volume 50, 1987, pages 1-102.

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

[44] A. GUGLIELMI, L. STRASSBURGER. *Non-commutativity and MELL in the Calculus of Structures.* in « CSL

2001 », series Lecture Notes in Computer Science, volume 2142, Springer-Verlag, L. FRIBOURG, editor, pages 54–68, 2001.

[45] M. KANAZAWA. *Learnable classes of Categorial Grammars.* CSLI, 1998.

[46] J. LAMBEK. *The mathematics of sentence structure.* in « Amer. Math. Monthly », volume 65, 1958, pages 154-170.

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

[48] I. MEL'ČUK. *Dependency Syntax: Theory and Practice.* State Univ. Press of New York, Albany/NY, 1988.

[49] J. MICHAELIS. *Transforming Linear Context-Free Rewriting Systems into Minimalist Grammars.* in « Proceedings of the conference Logical Aspects of Computational Linguistics (LACL '01) », series LNCS/LNAI, volume 2099, 2001.

[50] R. MOOT. *Proof Nets for Linguistic Analysis.* Ph. D. Thesis, Institute of Linguistics, University of Utrecht, 2002.

[51] 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 », series Lecture Notes in Artificial Intelligence, volume 2250, Springer, A. V. ROBERT NIEUWENHUIS, editor, pages 233-248, Dec, 2001.

[52] 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.

[53] Q. PUITE. *Sequents and Link Graphs.* Ph. D. Thesis, University of Utrecht, 2001.

[54] D. K. R. DI COSMO, E. POLONOVSKY. *Proof nets and Explicit subsitutions.* in « Fossacs'2000 », series Lectures Notes in Computer Science, number 1784, pages 63-81, 2000.

[55] J. ROGERS, K. VIJAY-SHANKER. *Obtaining Trees from their Descriptions: An Application to Tree-Adjoining Grammars.* in « Computational Intelligence », volume 10, 1994, pages 401–421.

[56] Y. SAKAKIBARA. *Efficient learning of context free grammars from positive structural examples.* in « Information and Computation », volume 97, 1992, pages 23-60.

[57] H. SEKI, T. MATSUMURA, M. FUJII, T. KASAMI. *On Multiple Context-Free Grammars.* in « Theoretical Computer Science », volume 223, 1991, pages 87–120.

[58] P. SGALL, L. NEBESKY, A. GORALCIKOVA, E. HAJICOVA. *A Functional Approach To Syntax In Generative Description Of Language.* 1969.

[59] L. TESNIÈRE. *Eléments de Syntaxe Structurale.* 1959.

[60] K. VIJAY-SHANKER. *Using Description of Trees in a Tree Adjoining Grammar.* in « Computational Linguistics », number 4, volume 18, 1992, pages 481-517.

[61] D. J. WEIR. *Characterizing Mildly Context-Sensitive Grammar Formalisms.* Ph. D. Thesis, University of Pennsylvania, 1988.