Logical Aspects of Computational Linguistics: an introduction.

Patrick Blackburn (Universität des Saarlandes, Saarbrücken)
Marc Dymetman (Rank-Xerox Research Center, Grenoble)
Alain Lecomte (Université Grenoble 2)
Aarne Ranta (Helsingin Yliopisto)
Christian Retoré (INRIA-Lorraine, Nancy)
Eric Villemonte de la Clergerie (INRIA-Rocquencourt)

Abstract. The papers in this collection are all devoted to single theme: logic and its applications in computational linguistics. They share many themes, goals and techniques, and any editorial classification is bound to highlight some connections at the expense of other. Nonetheless, we have found it useful to divide these papers (somewhat arbitrarily) into the following four categories: **logical semantics of natural language**, **grammar and logic**, **mathematics with linguistic motivations**, and **computational perspectives**. In this introduction, we use this four-way classification as a guide to the papers, and, more generally, to the research agenda that underlies them. We hope that the reader will find it a useful starting point to the collection.

1 Logical semantics of natural language

Logical semantics of natural language is a diverse field that draws on many disciplines, including philosophical and mathematical logic, linguistics, computer science, and artificial intelligence. Probably the best way to get a detailed overview is to consult the many excellent survey articles in [19]; this Handbook is likely to be the standard reference for work on logic in linguistics for some time to come. In the space of a short introduction like this, it is difficult to give an accurate impression of such a diverse area; nonetheless, by approaching the subject historically, we can at least hope to give a sense of the general direction the field is taking, and indicate how the articles on logical semantics to be found in this collection fit in with recent developments.

Modern logic begins with the work of Frege, and so does logical semantics. His celebrated Begriffsschrift [50] was not only one of the first formal treatments of quantified logic, it also discussed the structure of natural language quite explicitly. Moreover, Frege's work inspired others to attempt explicit logical treatments of natural language semantics: Carnap [25] and Reichenbach [98] are important (and still widely referenced) examples. Nonetheless, in spite of such impressive precedents, it is usual to regard the work of Montague as the starting point of the truly modern phase of formal semantics. Why is this?

Montague was a highly original thinker, and his three seminal articles on natural language semantics (*English as a formal language*, *The proper treatment of quantification in ordinary English*, and *Universal Grammar*, all in [116]) synthesise the best of

what had gone before (namely, the possible worlds semantics of Carnap [25] and Kripke [72], together with the higher order logic of Church [41]) with a potent new idea: semantic construction need not be a hit-or-miss affair that relies on intuition. Precise — and indeed, reasonably straightforward — algorithms can be given which translate interesting chunks of natural language into logical formalisms. Moreover (as Montague proved in a very general way in *Universal Grammar*) if the translation procedure obeys certain natural rules, the semantics of the logical formalism induces the desired semantics on the natural language input. From the perspective of the 1990s it can be difficult to appreciate how revolutionary Montague's vision was — but this is largely because his insights are now a standard part of every working semanticist's toolkit. To be sure, semanticists rarely work with Montague's type theory any more, and Montague himself might well have been surprised by the wide range of semantic issues now addressed using logical methods. Nonetheless, the vision that drives modern logical semantics is essentially Montague's: it is possible to model an important part of natural language semantics using logical tools, and the construction of semantic representations is algorithmic.

Montague's work appeared in the late 1960s and the early 1970s. The 1970s were, by and large, a period during which his ideas were assimilated by the linguistic community. Thanks largely to the efforts of Partee (like [92]), Montague's logical methods (which linguists found strange and new, and often intimidating) slowly gained acceptance. During this period, Montague's ideas were extended to cover a wider range of linguistic phenomena; the investigations into the temporal semantics of natural language recorded in [45] by Dowty are a nice example of such work.

If the key themes of the 1970s were "assimilation" and "development" the key theme of the 1980s was "diversity". For a start, two new approaches to logical semantics were developed: Situation Semantics of Barwise and others [15] and Discourse Representation Theory of Kamp and others [68,60,69]. Like Montague semantics, Situation Semantics offered a model theoretic perspective on natural language semantics, but it changed the ground rules in seemingly radical ways. For a start, whereas Montague's logic was *total* (that is, every statement had to be either true or false, and not both, at every possible pair of possible worlds and times), Barwise and Perry took seriously the idea that real situations were often "small", partially specified, chunks of reality. Accordingly, it seemed appropriate to drop the assumption that every sentence had to be either true or false in every situation — hence situation semantics naturally leads to the use of *partial* logics. Furthermore, stipulating the conditions under which sentences were true was no longer to be the central explanatory idea in semantics. Rather, meaning was to be viewed as a relation between situations, a relation that emerged from the interplay of complex constraints.

The agenda set by Discourse Representation Theory (DRT) seemed, at first, less revolutionary. Essentially, DRT proposed using an intermediate level of representation in a more essential way than Montague's work did. Although Montague made use of translations into an intermediate level of logical representation, in his work this level is a convenience that is, in principle, eliminable. Kamp and Heim showed that by making heavier — indeed, intrinsic — use of this intermediate level, interesting treatments could be given of temporal phenomena, and a number of troublesome problems involv-

ing quantifier scopes could be resolved. This may sound reasonably innocuous — but actually DRT has proved to be far more genuinely subversive to received semantic ideas than Situation Semantics. For a start, the idea that an intermediate level of representation is intrinsic to meaning is both interesting and highly controversial philosophically. Moreover, from DRT stems one of key words of modern semantics: *dynamics*. Thinking of semantics in terms of DRT leads fairly directly to an interesting — and very different — conception of meaning: *meanings as functions which update contexts with new information*. This essentially computational metaphor has developed into an important perspective for looking at semantics; [58] by Groenendijk and Stokhof is a prominent example of such work.

While DRT and Situation Semantics are perhaps the two key new ideas that emerged in the 1980s, a number of other themes emerged then that continue to play an important role. For a start, linguists slowly became aware of other traditions in logical semantics (the game theoretic approach instigated by Hintikka is perhaps the most important example, see [105]). Moreover, perhaps for the first time, semantic investigations began feeding important ideas and questions back into logic itself; the development of generalised quantifier theory by Barwise and Cooper or van Benthem [14,16] is an important example of this.

So where do we stand in the 1990s? Here the key theme seems to be "integration". At the logical level, there seems to be an emerging consensus that what is needed are frameworks in which the insights gathered in previous work can be combined in a revealing way. For example, work in Situation Semantics and DRT has made clear that ideas such as partiality and dynamism are important — but are there well-behaved logical systems which capture such ideas? Some interesting answers are emerging here: for example, Muskens in [90] shows how the insights concerning partiality and situations can be integrated into classical type theories that are simpler (and far more elegant) than Montague's, while Ranta in [97] shows that constructive type theory offer a uniform logical setting for exploring many current problems in logical semantics, including those raised by DRT. A second theme that is emerging is the use of inference as a tool for solving semantic puzzles. This is a natural step to take: one of the advantage of doing semantics in well understood formalisms (such as various kinds of type theory) is that the inference mechanisms these formalisms possess offer plausible mechanisms for modeling further phenomena. A third — and crucial — theme is how best to link all this semantical and logical work with the new and sophisticated approaches to syntax (and phonology) that have been developed over the fifteen or so years. The syntax-semantics interface is likely to be a focus of research in coming years.

Bearing these general remarks in mind, let's consider the individual contributions.

Sloppy identity by Claire GARDENT, exploits for semantic purposes the existence of Higher Order Unification (HOU) algorithms for classical type theory. The key linguistic idea is that a wide variety of sloppy interpretation phenomena can be seen as arising from a semantic constraint on parallel structures. As the author shows, by making use of HOU — essentially a powerful pattern-matching mechanism — a precise theory of sloppy interpretation can be given for such parallel constructions. Among other things, this theory captures the interaction of sloppy/strict ambiguity with quantification and binding.

The papers *Vagueness and type theory*, by Pascal BOLDINI, and *A natural language explanation for formal proofs*, by Yann COSCOY, are both based on constructive type theory, but they use it for very different purposes. Boldini's paper gives an analysis of certain so-called vague predicates, such as "small", which, in addition to creating a problem for logical semantics, have played a role in the philosophical controversy between constructive and classical logic. Coscoy's paper is associated with one of the major computer applications of constructive type theory, proof editors. It explains an algorithm that produces natural language texts to express formal proofs, and discusses some stylistic principles to improve the texts. Coscoy's algorithm is actually available as a part of the latest release of the proof editor Coq [12].

The paper *A belief-centered treatment of pragmatic presupposition*, by Lucia MANARA and Anne DE ROECK, addresses the problem of linguistic presupposition — the phenomenon that the meaningfulness of a sentence may demand the truth of some proposition. For instance, "John's wife is away" presupposes that John has a wife. This problem is approached in terms of the logical calculus of property theory, which provides new ideas for the interference of beliefs with the phenomenon of presupposition.

Whereas the previous four papers essentially exploit in various ways the fact that formal semantics can be done in standard logical systems (be it classical type theory, constructive type theory, or property theory) *Language understanding: a procedural perspective*, Ruth KEMPSON, Wilfried MEYER VIOL and Dov GABBAY is an attempt to view the syntax-semantics interface from a logical perspective. In particular, the authors show how an incremental parser can be formulated using the idea of labeled deductions, aided by a modal logic of finite trees. They argue that this blend of ideas can deal with crossover phenomena in a way that eludes other frameworks.

2 Grammar and Logic

The history of the relations between Grammar and Logic is as old as these two fields: we say they descend from the Ancient Greeks or the first grammar of sanskrit by Pāṇini. It is worth noticing in particular that Ancient Indian linguists around the 4th century B.C. had the idea of a concise system of rules (or *sutras*) for describing the formation of words and sentences. Some commentators [108] have argued that such a requirement of economy in the description is the departure point for a scientific study of language. In modern times, we can also refer to the Port-Royal School (17th century) for which it was natural to put a bridge between Grammar and Logic just because the language was supposed to express the logical thought. The reader interested in these aspects of the history of linguistics is referred to, for instance [89,102].

The generativism of the 20th century often compares itself to these traditions, and indeed many features of the Chomskyan program were already met in the ideas of 17th century philosophers as well as those of Indian grammarians, as explained by Chomsky in [35]. The most important similarity is definitely the concept of a finite device for generating an infinite set of sentences and the notion of an explicit procedure for generating grammatical strings. The main difference lies in the tools existing for expressing such ideas at different times. When Chomsky began his research on natural language in the fifties, mathematical logic was sufficiently developed to provide the

concepts of *automaton* and *formal grammar* — see e.g. [83] for a survey. As firstly a logician, Chomsky himself helped to develop these concepts and we are indebted to him for a hierarchy of formal languages [30,31] and the relationship between linguistic theory and formal language theory [32,33,40]. Of course the notion of *rule* had no logical status by itself, even if it was possible to make a link with the notion of *sequent* in the Gentzen style presentation of logic, and grammatical symbols were conceived as atomic symbols. So, the link with traditional logic was rather poor.

But some decades before, Polish logicians, the first being Lesniewski, inspired by Husserl's philosophy, had explored the theory of *semantic categories* — see for instance [29] for an history. This had led Ajduckiewicz [8] to the idea of a very simple algorithm to check well-formedness. This algorithm for what Ajduckiewicz called "syntactic connectivity" was based on a "quasi-arithmetic notation for categories". For instance, an intransitive verb was represented by the fractional notation $\frac{s}{n}$ which expresses that in presence of a word of category n, trivial rules of simplification lead to an expression of category s. This was the starting point for *categorial grammar*.

After that, Bar-Hillel added the notion of directionality [10,11], thus making a distinction between s/n and $n\backslash s$ and considered that an expression could belong to several categories as explained in [29]. In 1958, Lambek [74] published his seminal *Mathematics of Sentence Structure*, that presents for the first time an authentic calculus of syntactic categories, defined by a system of axioms and rules and in which it was possible to deduce the set of categories to which an expression belonged. This *calculus of syntactic types* was presented as a sequent calculus and this logical formulation helped in clearly establishing important properties, and in particular the cut-elimination and sub-formula property (see next section).

In the eighties, while logicians studied the mathematical properties of the Lambek calculus (see next section), linguists became interested in categorial systems. Bach [9], Ades and Steedman [7], Dowty [46] recognized their expressive power for linguistic description essentially because of their direct account of the notion of *valency*. Steedman [112], Szabolcsi [114] and Desclés [43] (the latter influenced by a neighbouring school born in the former USSR and led by Shaumyan, that gave rise to the so-called Applicative Grammar [107]) used the relations existing between categorial grammars and *combinators* for expressing *reflexivisation*, *binding*, *gapping*, *coordination of non constituents* and *long-distance dependencies*.

Moortgat thoroughly explored the linguistic description capacities of the Lambek calculus in his book [85] and in [86], resetting in a proper way Montagovian semantics by means of the *Curry-Howard isomorphism* and also giving a description of the relation between the prosodic and the syntactic algebras. From then on, linguistic entities have been considered as triples $<$ *Syn* : *Phon* : *Sem* $>$ where *Syn* is the syntactic part, *Phon* the phonetic part and *Sem* the semantic part of the sign.

At the end of the eighties Girard introduced Linear Logic [53], and the Lambek calculus was recognised as a fragment of non-commutative linear logic [2,103], thus putting a strong connection between syntax (categorial grammar) and logic (proof theory). Indeed Linear Logic provides a neat logical system to describe resource consumption, and, through the unary connectives called "exponentials" or "modalities", neatly

relates the Lambek calculus to the classical and intuitionistic logics which are used for semantical descriptions.

But the pure Lambek calculus has many limitations, and several investigations were made in the direction of an enrichment with new connectives. These new connectives, which are different ways of combining grammatical resources, are mainly *modalities*, called "structural", and they have the same rules as those we can find in Modal Logic, or they behave like exponentials in Linear Logic [3]. These modalities were originally introduced by the Edinburgh School [13] and strongly used by Hepple [62] (island constraints) and by Morrill. The latter in [88] gives an impressive account of linguistic phenomena, including *extraction, coordination, pied-piping, prosodic phrasing*... in a grammar expressed in purely logical types.

During the same time, Ranta [97] gave another viewpoint on grammar and semantics based on the Constructive Type Theory of Martin-Löf [82]. Regarding the syntax, this approach is in certain respects more traditional than the afore mentioned advances because it relies on [30]; but is very powerful on the semantic (and even pragmatic) side because of its use of *dependent types*, allowing a proper treatment of dynamic binding (e.g. Geach sentences) and thus going beyond Montague semantics.

Finally, Moortgat in [87] provides the widest survey of *Categorial Type Logics*, including the syntax-semantics interface, multimodality, polymorphism, hybrid architectures and categorial parsing.

Chomsky's program was continued in parallel but we have no room here to trace back the long history of Generativism — one should refer to the main books by Chomsky [30,34–39] or to the introductive textbook [96]. What appears as a curious fact in the end of the nineties is a real convergence between the generativist trend and the categorial one (see below). Of course, on the theoretical side, Pentus [93] demonstrated that Lambek grammars have the same generative power as Context-Free grammars, and this is an important link. But it is only a formal result. From a historical point of view, we are presently witnessing a deep change in Generative Grammar: in the recent minimalist program [39], traditional levels (the deep level and the surface level) and even the X-bar theory are abandoned in favour of a unique system of Generalized Transformations very similar in their spirit to the reduction rules of Categorial Grammar.

Two main points concerning the so-called *Minimalist Program* [39] will be stressed here: one concerning the derivational conception advocated by Chomsky himself and the other the notion of Resource Consumption as it appears in the operations of Feature Checking for which the Generalized Transformation *Move* is required. The notion of a derivational theory is preferred to that of a representational one for many reasons and notably because it allows a better characterization of local phenomena like the satisfaction of the HMC constraint cf. [39, pp. 49–50]. Moreover, it is coherent with the general conception of Minimalism for which it is necessary to avoid all syntactic entities (like *rules*, *representations*, *representation-levels* and so on) which are not absolutely necessary for linguistic description and explanation. In a derivational perspective, only the derivation itself counts as a "representation" for an analysis of a sentence.

In this volume, *Derivational minimalism* by Edward STABLER proposes an attractive formalism to take such a view into account. His formalism provides a formulation for structure-building operations like *Merge* and *Move*, and he shows how to express the

fact that such operations are "feature-driven". That means that *Move*, for instance, is applied only when necessary in order to check a feature, weak or strong. The weak/strong distinction is used to describe an opposition between *overt* and *covert* movements in syntax and it provides a parameter for distinguishing various families of natural languages (like SVO, VSO…). This paper also provides formal examples which are of great interest if we wish to compare this formalism to others like for instance *Tree Adjoining Grammars*.

Tree Adjoining Grammars (TAGs), introduced in [65] are present among the papers published here, and for a recent survey on TAGs one should refer to [67,1] for example. They, too, provide an appealing formalism, firstly because it is mathematically a very elegant one and secondly because it has often been explored and its properties regarding complexity and generative power are well-known [66]. One point is mainly dealt with in this volume: its connection with the derivational paradigm. As we said previously, there is a strong connection between Chomsky's Minimalism and Categorial Grammar, and Categorial Grammar could be a good candidate as a framework for Minimalism: for instance, a Generalized Transformation like *Merge* is very close to Cancellation Schemes, so familiar to the categorial grammarian. Nevertheless categorial grammar lacks some important notions which are easily described in (Lexicalized) TAGs like "the notion of an extended domain of locality (EDL)" and the consequent "factoring of recursion from the domain of dependencies (FRD)". Thus, exploring the link between the derivational paradigm (expressed by Linear Logic or Lambek calculus) and TAGs, is also useful for bringing together, as intuition suggests, the derivational paradigm and the minimalist program.

The paper *Tree adjoining grammars in non-commutative linear logic* by V. Michele ABRUSCI, Christophe FOUQUERÉ and Jacqueline VAUZEILLES presents a derivational conception of TAGs. It uses the concept of *proof net* which is an essential ingredient of the proof theory of linear logic. The encoding of substitution and adjunction, mainly uses the *cut-rule* restricted to atomic formulae. Let us recall that the cut-rule is a very particular rule in every logical system with "good properties": it does not provide more theorems than the system does without it, but is a simple way to combine two proofs into one. When using the cut-rule for the TAG purpose, trees are considered as provable sequents in the Non-Commutative Intuitionistic Linear Logic (N-ILL or Lambek calculus) and adjunction as two occurrences of the cut-rule. They thus obtain a complete translation of TAG-grammars into the Lambek calculus — but the notion of grammar they use is not the one defined by Lambek.

The paper *Partial proof trees, resource sensitive logics and syntactic constraints* by Aravind K. JOSHI and Seth KULICK tries to incorporate LTAG notions like ELD and FRD into the categorial framework, thus attempting to produce a new synthesis between LTAG and Categorial Grammar. By doing so, they switch from a notion where lexical entries are *formulae* (as in all the traditions of Categorial Grammar) towards a conception according to which they are pieces of *proofs* (so-called "Partial Proof-Trees"). The standard operations of *substitution* and *adjoining* in TAG are therefore replaced by operations on Partial Proofs, namely: *application*, *stretching* and *interpolation*. As its name indicates, *stretching* consists in duplicating a node in a Partial Proof Tree and inserting a new proof between the two instances. In *interpolation*, the place is reserved for a proof

insertion between two nodes generated at the lexical stage. Exactly as in TAG-trees, dependencies can be registered as local relations in initial Proof-Trees. The authors also discuss the differences and proximities between their approach in the setting of Partial Proof Trees and the Proof Net approach, as they call the formalisms advocated in the paper by Abrusci et al., as well as by Lecomte and Retoré [78,79].

The latter viewpoint is reflected in another paper *Pomset logic and variants in natural languages* by Irene SCHENA. The author proposes another view on combining proofs by means of Pomset Logic, an extension of Linear Logic made by Retoré [99,101]. The main difference with Abrusci et al. resides in the fact that the cut-rule is used between general formulae, and also in the essential use of the cut-elimination algorithm. This formalism is particularly relevant for word-order phenomena: when *Cuts* are eliminated, new proof nets are produced, which can be read as giving the correct order of words in a sentence. The author goes deeper in the description of linguistic phenomena by means of this logic: she proposes analysis for topicalized sentences. According to her views, there are initial proof nets which are not associated with lexical entries but used in order to properly connect these entries. We can compare these modules to non-lexical signs (functional ones, associated with I and C) in the Chomskyan theory.

As we can see, these grammatical formalisms share many features: they all try to combine, in a way or another, elementary trees or their representations in some logic. Here Linear Logic is very often referred to because it gives the best account of Resource Consumption. When a feature, or a node, is used for some purpose (like checking agreement or case) it can no longer be used. The interest of Linear Logic is to keep encapsulated such "economy principles" so that we do not have to formulate them separately.

The paper *Inessential features* by Marcus KRACHT present another way, mainly model theoretical, of formulating trees in Logic. Indeed, the first order theory of trees may be axiomatised in Modal Logic. This is what this paper does, in a way similar to [20] or [57]. But Kracht's objective is not only to provide the best way for "talking about trees", it is to go very deep into the key insights of the Minimalist approach, asking when a feature is eliminable from a grammar. This is a way of taking very seriously the intentions of the Chomskyan program. If Chomsky abandons the levels of X-bar theory, it is because he intuitively thinks that such distinctions can be recovered if necessary from the remaining parts of the description. Moreover, another important point shared by all these approaches is *lexicalism*. The question arises of determining exactly what is the price to pay for reaching radical lexicalism. A true lexical theory is one where the computational system only treats features which are introduced at the lexical stage and need no other feature (like BAR, SLASH...). This is the objective that researchers like Stabler and others pursue, but a theoretical study is needed for defining in a precise way the conditions in which some "syntactic" features can be eliminated. Kracht's paper is an answer to this question.

3 Mathematics with linguistic motivations

As often the interplay between two areas of research open new research directions for both. Let us quote here a few interesting purely mathematical/logical questions motivated by linguistic considerations, related to the articles in this volume.

3.1 Formal language theory

The best known connection is certainly Formal Language Theory[1] which as stated above was initiated in the late fifties with the works of Chomsky [31,32] for natural language syntax.

Firstly this was considered as a part of logic, because of its relation to the theory of computability. Nevertheless it soon developed as an extended independent theory, with its own techniques, the major reason being its success for the study of programming languages and compilation. Therefore it is impossible to give a short account of such a wide research field; however the basic notions are by now part of the standard curriculum in computer science. For more advanced notions, the reader is referred to, for instance [104], which is a collection of comprehensive survey papers, the first one being quite introductive and general [83]. Now, let us just select a few topics relevant to this volume.

The classical grammars of Chomsky have been extended to grammars for trees. From a linguistic point of view this clearly makes sense: one is not only interested in the sentences, but also in their analysis into constituents or phrases. Thus tree grammars have been developed, in several ways: for instance one can consider regular trees, the ones generated by a tree automaton (see [52] for a survey), or the trees generated by substitution and adjunction from a finite set of trees (Tree Adjoining Grammars) (see [67] for a survey).

Formal grammars, either for strings or trees, describe a language, by means of a finite set of rules, and are especially useful from an algorithmic perspective. Indeed, a natural aim is to use them to actually parse a sentence. This leads to two mathematical questions. Can we decide whether a sentence is generated by a grammar of a given kind? If so, what is the complexity of the algorithm? For instance, one may decide in less than $O(n^3)$ whether an n-word sentence of n words is generated by a context free grammars.

Unfortunately, this kind of grammar is not sufficient for some syntactical phenomena, and does not handle non purely syntactic information, like morphology (for agreement), semantics etc. Therefore in the eighties, inspired by the Logic Programming formulation of standard grammars, unification grammars have been introduced: LFG [70,106,1], GPSG [51,106,117,1] and HPSG [95,1]. One mathematical problem put forward by these formalisms is the question of the decidability: in general it is undecidable

[1] Given that this conference took place in 1996 in France, let us seize the opportunity to salute the memory of Marcel-Paul Schützenberger (1920-1996). He had been a pioneer in formal language theory, not to mention his other numerous scientific interests. As an example of his work in Formal Language theory, let us just mention the famous paper he wrote with N. Chomsky on context-free languages [40]. He undoubtedly is one of the founders of the French school of Theoretical Computer Science, with M. Nivat and L. Nolin.

whether a sentence may be generated from such a grammar, and this is rather worrying from a computational point of view. Therefore, their logical counterpart, namely feature logics, have been studied [63,28,71] and tools for handling them have been developed like ALE [26], CUF [44], and TFS [119], a major challenge being to isolate a family of logics which would be sufficient for natural language purposes but, nevertheless decidable — and in polynomial time if possible!

But one can also consider less computational descriptions of the set of all languages generated by a particular kind of grammar, to prove properties of families of languages. These declarative descriptions may be either algebraic or model theoretic. The most familiar examples of algebraic description certainly are the regular expressions written with concatenation, union and the Kleene star operation, which describe regular languages. Model theoretic description are probably less well-known, and for a survey the reader is referred to [115]. As an example, a regular language may be viewed as the models of a formula of weak second order logic with a single successor symbol. These descriptive formulations, and the mathematical properties they allow to prove, are especially useful for characterising the expressiveness of the grammars, that is to say the set of languages they correspond to. This has an underlying theoretical linguistic motivation: which class of languages is actually needed for natural language syntax?

Let us give a famous example which stresses the linguistic relevance of non-algorithmic results in formal language theory. It has been *formally* shown that Swiss German is not context-free [109] not by the use of parsing considerations but by the use of algebraic results: the closure of context-free languages under homomorphisms and intersection with regular languages. [2]

In this volume several papers address the aforementioned questions:

Strict LT2 : Regular :: Local : Recognizable by James ROGERS is devoted to tree languages as models of second order formulae. It defines a class of logic formulae such that the tree analyses of context free grammars are exactly the sets definable in this class. Furthermore, this formula explicitly expresses how much smaller this class is than the class of trees generated by tree automata. Incidentally, the research agenda underlying this paper has much in common with that of *Inessential features* by Marcus Kracht. Roger's LT2 is yet another paper for "talking about trees". However, like Kracht, Rogers is interested in far more than simply developing a good descriptive format. The links this papers forges with formal language theory means that his language can be used to analyse the strength of various linguistic theories such as Government and Binding theory or Generalised Phrase Structure Grammar.

Semilinearity as a syntactic invariant by Jens MICHAELIS and Marcus KRACHT shows that actually some natural language, namely Old Georgian is not mildly context sensitive, because of its very particular genitive construction. The argument consists in showing that it is not semi-linear, a property which mildly context sensitive languages have.

A family of decidable logics which supports HPSG-style set and list constructions by Stephen J. HEGNER addresses the important topic of how to define a family of *decidable* feature logics which could support at least some of the key constructs of HPSG. In

[2] This is a strictly stronger result than showing that there is no *sensible* (when correctly bracketing the constituents) context-free grammar generating this language.

HPSG, sets and lists occur in fundamental ways, for example in filler-gap constructions or in the value of the feature COMP-DTRS (*subcategorization principle*). Hegner's paper brings "the only work to treat multisets and lists uniformly within a decidable feature-structure framework". Decidability is proven by mapping the logic into a sorted first-order logic, and then using variants of decidability results in that context.

3.2 Lambek calculus and non-commutative (linear) logic

Even if we take into account all the strange formal systems that have been proposed as "logics", one observes that logics are rarely non-commutative. The calculus introduced by Lambek in the late fifties [74–76] is the notable exception which undoubtedly deserves to be called a logic. It was introduced as a logical formalisation of the categorial grammars of Adjukiewicz and Bar-Hillel [10,11] together with the related calculi. Indeed, this calculus, a plain propositional logic defined by a sequent calculus à la Gentzen, admits cut-elimination and the subformula property, truth value semantics, denotational semantics, and is fairly simple: seven rules, one non-commutative conjunction and two implications (because of the non-commutativity).

The properties of this calculus, once rather isolated from other calculi, are now easily observed and formulated in the linear logic framework. Linear logic was introduced ten years ago by Girard [53] — see [54] for a recent survey and [118] for a textbook. Although it was conceived rather as a logic of computation and communication than for linguistic purposes, it explains the various aspects of logical calculi with restricted structural rules and their relations to the standard intuitionistic or classical logic, via the unary connectives called "exponentials" or "modalities". In this setting one observes the following properties of the Lambek calculus:

- There are no rules of contraction or weakening (aka thinning). This is responsible for the logic to be linear, in other words: the formulae may be viewed as resources, that may not be duplicated or erased. This property of linearity seems actually necessary to have a non-commutative logic, although this probably cannot be proved, because determining what a logic is certainly leads to Byzantine discussions.
- It is a multiplicative calculus, that is to say: in each rule with two premises, the context of the conclusion sequent is obtained by adding the contexts of the two premise sequents. This means that the calculus describes how the resources are combined. The other possibility would be to ask for both contexts to be equal, so the introduced connective would correspond to choice operators, called additive connectives.
- It is an intuitionistic calculus, that is to say there is a single formula on the right hand side. Intuitively, this means that a proof of a sequent may be viewed as a function that takes resources of the types/formulae of the left hand side and returns an object of the type of the conclusion.
- It is a non-commutative calculus. That is to say there is no exchange rule and thus no way to prove the commutativity of the product. This intuitively correspond to the fact that we handle an ordered list of resources.
- Finally, a proof should not contain any sequent without hypothesis. This is a linguistic requirement rather than a logical one: without this extra requirement, the

types of the empty sequence would be all Lambek tautologies, and thus one would not be able to specify that a word requires on its left or right a constituent whose type is a tautology. Fundamentally, it does not affect the logical properties of the calculus, except that, as shown by Zielonka in [120], the Lambek calculus is not finitely axiomatisable (in a Hilbert-style formulation) — see also [80].

This is a typical example of an elegant mathematical structure introduced for linguistic motivations. Indeed, this calculus was introduced as a lexicalised grammar, which logically formalises the Adjuckiewicz Bar-Hillel categorial grammars. The lexicon assigns a formula or type to each word; then analysing a phrase of type T consists in proving T from the lists of the types of the words in the phrase. In this parsing-as-deduction perspective, the subformula property, deduced from the cut-elimination theorem [74–76] is then highly important: it restricts the search space to a finite number of formulae, so this property makes the parsing obviously decidable and guides each step. Another important proof theoretical property of the Lambek calculus is interpolation, established by Roorda in [103], which states that whenever an implication holds there always exists an intermediate formula, whose only atomic formulae are common to the antecedent and consequent. This enables, as shown in this volume, a transformation from one bracketing of a sentence to another, e.g. from the syntagmatic one to the intonational one.

Now how does it relates to linear logic? What is the benefit of such a connection? What mathematical properties, preferably with a linguistic meaning, does it enable us to prove? The basic relation between the Lambek calculus and linear logic is that the Lambek calculus is exactly the multiplicative fragment of non-commutative linear logic. From this fact, the study of the Lambek calculus can benefit from the richer framework of linear logic — see e.g. [100] for a survey.

One of the advantages of linear logic is that it makes explicit the relationship of this calculus to intuitionistic and classical logics. Thus, the simply typed lambda calculus is easily viewed as an extension of the Lambek calculus. This way the semantical representation à la Montague, that is to say a simply typed lambda term, i.e. a proof in intuitionistic logic, has the same structure as a syntactical analysis, i.e. a proof in the Lambek calculus. This enables a particularly simple interface between syntax and semantics [85,88,27], which is especially clear in the proof net syntax [59].

This proof net syntax is precisely another convenient tool provided by linear logic [53,118,54,73]. It is a very compact and elegant notation for proofs in these logical calculi, which avoids, for instance, the so-called spurious ambiguities of categorial grammar analyses [100]. The structure of a proof net is very much like a family of trees, and thus one is able to picture in this setting numerous phenomena studied in the standard linguistic formalisms, usually described by trees; on the logical side they allow easy proofs, of cut-elimination, interpolation etc. As an example of a linguistic outcome, a recent paper [64] shows that using this kind of syntax allows one to measure the understanding complexity of center-embedded relatives. Finally, proof nets also enable the definition of efficient heuristics for parsing. This is especially useful because one exciting open question is the complexity of the parsing, or provability in the Lambek calculus.

Another advantage of having a purely logical calculus is that one may use semantics to prove properties of the calculus. Actually, there are basically two kinds of semantics that may be used: the usual truth valued semantics, for which the completeness theorem holds, and the denotational semantics, i.e., the semantics of proofs.

Denotational semantics was introduced by Scott for the semantics of programming languages and lambda calculus, see e.g. [113], and applies to typed lambda calculus and linear logic see e.g. [56,53,118]. Such a semantics associates a "space" $||F||$ to each formula F, and an object of the space $||F||$ correspond to a proof *up to cut-elimination*. The linguistic meaning of this kind of semantics is yet unclear. Nevertheless in [4] it is used to show the difference between the two analyses of the classical example "every man loves a woman". It should be observed that, as opposed to the general case, the multiplicative fragment of linear logic, quite sufficient for syntactic analysis, admits a complete denotational semantics [81]; this should help in finding a linguistic meaning for denotational semantics.

Truth value semantics, for which logic completeness results usually hold, have been substantially studied in categorial grammar, mainly by Buskowski and van Benthem, [22,23,16–18,24]. For instance, they established completeness with regards to *relational* interpretations and with regards to *monoidal* interpretations see e.g. the section 1 of Martin Emms'paper in this volume. These latter models are very similar to the so-called "phase semantics" of linear logic [53,2,118]. Roughly speaking, these models interpret a formula A as the sets of all phrases of type A, i.e. as a part of a monoid. Somehow these structures formalise the philosophical notion of a category and, technically, they lead to substantial results. The most famous concerns the relation between Lambek grammars and formal language theory. Given a context-free grammar, it is a simple exercise to find a Lambek grammar, even in a small fragment of the Lambek calculus, that generates the same language. Nevertheless it was conjectured by Chomsky in 1963 [33, p. 413] that Lambek grammars defined by the whole Lambek calculus should not go beyond context free languages. After several unsuccessful attempts, this was indeed proved by Pentus in 1993 [93], and the argument extensively uses this kind of models, together with a refinement of the interpolation theorem for the Lambek calculus established in [103].

This result confirmed the intuition: Lambek calculus is too restrictive to describe a linguistically sensible class of formal languages, as also shown in [77] and one should look for extensions of the Lambek calculus — still in a neat logical framework, if possible. A first direction, quite standard from a logical point of view, is to consider fragments of the second order Lambek calculus, i.e. to allow type variable and quantification over them, as developed by Emms [47–49]. For instance the conjunction "and" should received the type $\forall X.\ X\backslash X/X$. This way one goes beyond context-free languages [47], but still can remain within a decidable fragment of second order Lambek calculus [48] — indeed, unrestricted second order Lambek calculus is undecidable [49].

Linear logic provides newer ways to extend the Lambek calculus. Indeed, linear logic allows for several conceptions of a non-commutative logic. One way is to extend the Lambek calculus to the corresponding classical calculus defined by Abrusci [2,5]. This leads, in particular, to a mathematical definition for the philosophical notion of a syntactic category [6]. Another way, introduced by Retoré [99,101], is to enrich the

commutative calculus with a non-commutative self-dual connective. Then one has to generalise slightly the notion of a Lambek grammar: the lexicon should associate partial proofs (partial proof nets in fact) to the words, and parsing consists in combining them into a complete proof — as done by Lecomte and Retoré in [78,79] and pursued by Schena in this volume.

Models for polymorphic Lambek calculus by Martin EMMS reviews truth value models for Lambek calculus, and then defines a notion of model for the second order Lambek calculus. Then the author establishes that this semantics is complete with respect to second order Lambek calculus.

Representation theorems for residuated groupoids by Marek SZCZERBA shows that every residuated groupoid — this is a rather general notion of model for the non-associative Lambek calculus — is isomorphic to a submodel of a residuated groupoid of a particular shape, namely the ones defined, as in phase semantics, by the powerset of a groupoid (groupoids are also known as "magmas").

Connected set of types and categorial consequence by Jacek MARCINIEC studies how one can infer from parsing data the type of each word, i.e. define a categorial grammar which generates these sentences and which is as simple as possible. This is done by studying unification over sets of types.

Constructing different phonological bracketings from a proof net by Denis BECHET and Philippe DE GROOTE gives a nice proof of the interpolation lemma for the Lambek calculus in the proof net syntax. This enables to move from a proof net of the Lambek calculus corresponding to one bracketing (e.g. the syntagmatic one) to another (e.g. the intonational one).

Generation as deduction on labelled proof nets by Josep M. MERENCIANO and Glyn V. MORRILL is devoted to generation in the framework of the Lambek calculus. It provides an algorithm for reconstructing the syntactic analysis of a sentence out of a Montague-like semantics — in the restricted case where there is no logical constant. The algorithm consists in constructing a proof net, i.e. a syntactic analysis in the Lambek calculus; the construction is guided by lambda terms which label the proof net: lambda terms must unify when they are "linked" by axioms of the proof net.

4 Computational perspectives

There is now a long tradition of using Logic Programming for implementing parsers (and generators) for natural language. In fact, Logic Programming itself originates in Natural Language Processing considerations. Prolog was created in 1972 by Colmerauer and Kowalski by merging Robinson's notion of unification in Automated Theorem Proving with the insights provided by Colmerauer's previous Q-systems, a programming language which he had designed for use in Machine Translation [42].

At the beginning of the eighties, the Unification Grammar paradigm [110,106,84], strongly connected to Logic Programming [94], gained more and more popularity in Computational Linguistics circles. One of the key advantages of such formalisms was their ability to free the grammar writer from an explicit management of many cases of "transversal information flow", that is, correlations between different points of analysis (the best known instance being agreement handling).

Since then, the fields of Computational Linguistics and Logic Programming have continued to gain insights from each other, in developments such as feature logics [71], resource accounting, constraint processing [61], concurrency, higher-order constructs (functionals).

Constraint logic programming for computational linguistics by Frieder STOLZENBURG, Stephan HÖHNE, Ulrich KOCH and Martin VOLK describes a system that uses Constraint Logic Programming to deal efficiently with typed feature-structures, with applications to finite set operations, variable-free negation and linear precedence constraints.

Quantitative constraint logic programming for weighted grammar applications by Stefan RIEZLER explains how a system of weights can be combined with a certain class of constraint languages and provides a formal semantics for the resulting programs. They show how the use of weights permits the parser to select the most "plausible" analysis in case of ambiguity, using a fuzzy logic notion of plausibility, rather than a probabilistic one.

The development of realistic parsers involves specific needs such as the efficient management of large sets of complex linguistic objects (for instance a lexicon). *The automatic deduction of classificatory systems from linguistic theories* by Paul John KING and Kirl Ivanov SIMOV explains how a classification system for typed feature structures may be derived automatically from a set of linguistic constraints about these structures (in a formalism related to HPSG). The resulting classification system is implemented as a tree whose each branch may be seen as a sequence of elementary queries and expected answers that are used to discriminate the objects to be classified.

In the continuation of his previous works to enrich Logic Programming, the extended abstract *Linear logic as logic programming* by Dale MILLER shows that the whole of linear logic can be considered as a logic programming language, which enables both high-order programming (functionals) and concurrency primitives. The author mentions several existing applications of this language in Computer Science and indicates its applicability to Filler-Gap Dependency Parsers, illustrating once more the practical relevance of linear logic to Computational Linguistics.

References

1. Anne Abeillé. *Les nouvelles syntaxes*. Armand Colin, Paris, 1993.
2. V. Michele Abrusci. Phase semantics and sequent calculus for pure non-commutative classical linear logic. *Journal of Symbolic Logic*, 56(4):1403–1451, December 1991.
3. V. Michele Abrusci. Exchange connectives for non commutative classical linear propositional logic. In V. Michele Abrusci, Claudia Casadio, and Michael Moortgat, editors, *Linear Logic and Lambek Calculus*. DYANA Occasional Publications, Institute for Logic, Language and Information, Amsterdam,1993.
4. V. Michele Abrusci. Coherence semantics for non-commutative linear logic and ambiguity in natural language. Technical report, Università di Roma tre, 1997. Talk at the ATALA meeting, Paris, December 94.
5. V. Michele Abrusci. Non-commutative proof nets. In Girard et al. [55], pages 271–296.
6. V. Michele Abrusci. Syntactical categories of a natural language as facts of a cyclic phase space. Technical report, Universita di Roma tre, 1997.

7. Anthony E. Ades and Mark J. Steedman. On the order of words. *Linguistics and Philosophy*, 4:517–558, 1982.
8. Kasimierz Adjuckiewicz. Die syntaktische konnexität. *Studia Philosophica*, 1:1–27, 1935.
9. Emmon Bach. Some generalizations of categorial grammars. In Fred Landman and Frank Veltman, editors, *Varieties of Formal Semantics*, pages 1–23. Foris, Dordrecht, 1984.
10. Yehoshua Bar-Hillel. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58, 1953.
11. Yehoshua Bar-Hillel. *Language and information*. Addison Wesley, Reading MA, 1964.
12. Bruno Barras, Samuel Boutin, Cristina Cornes, Judicael Courant, Jean-Christophe Filliatre, Eduardo Gimenez, Hugo Herbelin, Gerard Huet, Cesar Munoz, Chetan Murthy, Catherine Parent, Christine Paulin-Mohring, Amokrane Saibi, Benjamin Werner The Coq Proof Assistant Reference Manual : Version 6.1 Technical Report RT-203, INRIA, Rocquencourt, May 1997.
13. Guy Barry and Glyn V. Morrill, editors. *Studies in Categorial Grammar*. Edinburgh Working Papers in Cognitive Science, Edinburgh, 1990.
14. Jon Barwise and Robin Cooper. Generalised quantifiers and natural language. *Linguistics and Philosophy*, 4:159–219, 1981.
15. Jon Barwise and John Perry. *Situations and Attitudes*. MIT Press, Cambridge MA, 1983.
16. Johan van Benthem. *Essays in logical semantics*. D. Reidel, Dordrecht, 1986.
17. Johan van Benthem. The lambek calculus. In Oehrle et al. [91], pages 35–68.
18. Johan van Benthem. *Language in action: Categories, Lambdas and Dynamic Logic*, Number 130 of *Sudies in Logic and the foundation of mathematics*. North-Holland, Amsterdam, 1991.
19. Johan van Benthem and Alice ter Meulen, editors. *Handbook of Logic and Language*. North-Holland, Amsterdam, 1997.
20. Patrick Blackburn, Wilfried Meyer-Viol, and Maarten de Rijke. A proof system for finite trees. In Hans Kleine Büning, editor, *Computer Science Logic '95*, Volume 1092 of *Lecture Notes in Computer Science*, pages 86–105. Springer, Heidelberg, 1996.
21. Joan Bresnan, editor. *The mental representation of grammatical relations*. MIT Press, Cambridge MA, 1982.
22. Wojciech Buszkowski. Completeness results for lambek syntactic calculus. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 32:13–28, 1986.
23. Wojciech Buszkowski. Generative power of categorial grammar. In Oehrle et al. [91], pages 69–94.
24. Wojciech Buszkowski. Mathematical linguistics and proof theory. In van Benthem and ter Meulen [19], pages 683–736.
25. Rudolf Carnap. *Meaning and Necessity: A Study in Semantics and Modal Logic*. University of Chicago Press, 1946.
26. Bob Carpenter. ALE: The Attribute Logic Engine user's guide. Technical report, Carnegie Mellon University, Laboratory for Computational Linguistics, 1992.
27. Bob Carpenter. *Lectures on Type-Logical Semantics*. MIT Press, Cambridge MA, 1996.
28. Bob Carpenter. *The Logic of Typed Feature Structures: With Applications to Unification Grammars, Logic Programs and Constraint Resolution* Number 32 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1992.
29. Claudia Casadio. Semantic categories and the development of categorial grammars. In Oehrle et al. [91], pages 95–124.
30. Noam Chomsky. *The logical structure of linguistic theory*. Plenum, New York, 1955.
31. Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.
32. Noam Chomsky. On certain formal properties of grammars. *Information and control*, 2:137–167, 1959.

33. Noam Chomsky. Formal properties of grammars. In *Handbook of Mathematical Psychology*, volume 2, pages 323 – 418. John Wiley and sons, New York, 1963.
34. Noam Chomsky. *Aspects of the Theory of Syntax*. MIT Press, Cambridge MA, 1965.
35. Noam Chomsky. *Cartesian Linguistics*. MIT Press, Cambridge MA,1966.
36. Noam Chomsky. *Reflections on language*. Pantheon, New York, 1975.
37. Noam Chomsky. *Some concepts and consequences of the theory of government and binding*. MIT Press, Cambridge MA, 1982.
38. Noam Chomsky. *Barriers*. MIT Press, Cambridge MA, 1987.
39. Noam Chomsky. *The Minimalist Program*. MIT Press, Cambridge MA, 1996.
40. Noam Chomsky and Marcel-Paul Schützenberger. The algebraic theory of context-free languages. In P. Bradford and D. Hirschberg, editors, *Computer programming and formal systems*, pages 118–161. North-Holland, Amsterdam, 1963.
41. Alonzo Church. A formulation of the logic of sense and denotation. In P. Henle, H. Kallen and H. Langer editors, *Structure, Method and Meaning: Essays in honor of Henry M. Sheffer*. Liberal Arts Press, New York, 1951.
42. Jacques Cohen. A view of the origins and development of Prolog. *Communications of the ACM*, 31(1):26–37, January 1988.
43. Jean-Pierre Desclés. *Langages applicatifs, langues naturelles et cognition*. Hermès, Paris, 1990.
44. Michael Dorna. The Comprehensive Unification Formalism user's manual. Technical report, Institut für maschinelle Sprachverarbeitung, Universität Stuttgart, 1994.
45. David Dowty. *Word Meaning and Montague Grammar*. D. Reidel, Dordrecht, 1979.
46. David Dowty. Type-raising, functional composition, and non-constituent coordination. In Oehrle et al. [91], pages 153–198.
47. Martin Emms. Extraction covering of the Lambek calculus are not context free. In *9th Amsterdam Colloquium*, pages 268–286, 1993.
48. Martin Emms. Parsing with polymorphism. In *6th Annual Conference of the European Chapter of the Association for Computational Linguistics*, 1993.
49. Martin Emms. Undecidability result for polymorphic Lambek calculus. In *10th Amsterdam Colloquium*, 1995.
50. Gottlob Frege. *Begriffsschrift*. Louis Nerbert, Halle A/S, 1879. English translation in: "From Frege to Gödel", edited by van Heijenoort, Harvard University Press, Cambridge MA, 1967.
51. Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. *Generalized Phrase Structure Grammar*. Harvard University Press, Cambridge MA, 1985.
52. Ferenc Gécseg and Magnus Steinby. *Tree Languages*, chapter 1, pages 1–68. Volume 3 of Rozenberg and Salomaa [104], 1996.
53. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.
54. Jean-Yves Girard. Linear logic: its syntax and semantics. In Girard et al. [55], pages 1–42.
55. Jean-Yves Girard, Yves Lafont, and Laurent Regnier, editors. *Advances in Linear Logic*, Volume 222 of *London Mathematical Society Lecture Notes*. Cambridge University Press, 1995.
56. Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.
57. Valentin Goranko and Solomon Passy. Using the universal modality: Gains and questions. *Journal of Logic and Computation*, 2:5–30, 1992.
58. Jeroen Groenendijk and Martin Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14:39–100, 1991.
59. Philippe de Groote and Christian Retoré. Semantic readings of proof nets. In Geert-Jan Kruijff, Glyn V. Morrill, and Richard T. Oehrle, editors, *Formal Grammar*, Proceedings

of the Conference of the European Summer School in Logic, Language and Information, Prague, August 1996. pages 57–70.
60. Irene Heim. *The Semantics of Definite and Indefinite Noun Phrases*. PhD thesis, University of Massachusetts, Amherst, 1982.
61. Pascal van Hentenryck. *Constraint Satisfaction in Logic Programming*. MIT Press, Cambridge MA, 1989.
62. Mark Hepple. *The Grammar and Processing of Order and Dependency, a categorial approach*. PhD thesis, Centre of Cognitive Sciences, Edinburgh, 1990.
63. Mark Johnson. *Attribute-Value Logic and the Theory of Grammar*. Number 16 in CSLI Lecture Notes. Center for the Study of Language and Information, Stanford CA, 1989. (distributed by Cambridge University Press)
64. Mark Johnson. Proof nets and the complexity of processing center-embedded constructions. Technical report, Brown University, Providence RI, 1997.
65. Aravind K. Joshi, Leon Levy, and Masako Takahashi. Tree adjunct grammar. *Journal of Computer and System Sciences*, 10:136–163, 1975.
66. Aravind K. Joshi. Tree adjoining grammars : How much context-sensitivity is required to provide reasonable structural descriptions? In David Dowty, Lauri Kartunen, and Arnold M. Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, 1988.
67. Aravind K. Joshi and Yves Schabes. *Tree-Adjoining Grammars*, chapter 2, pages 69–124. Volume 3 of Rozenberg and Salomaa [104], 1996.
68. Hans Kamp. A theory of truth and semantic representation. In Janssen, Groenendijk and Stokhof, editors, *Proceedings of the Third Amsterdam Colloquium*, Foris, Dordrecht, 1981.
69. Hans Kamp and Uwe Reyle. *From Discourse to Logic*. Kluwer, Dordrecht, 1993.
70. Ronald Kaplan and Joan Bresnan. *Lexical functional grammar: a formal system for grammatical representation*, chapter 4, pages 173–281. In Bresnan [21], 1982.
71. Bill Keller. *Feature Logics, Infinitary Descriptions, and Grammar*, volume 44 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford CA, 1989. (distributed by Cambridge University Press)
72. Saul Kripke. A completeness theorem in modal logic. *Journal of Symbolic Logic*, 24:1–14, 1959.
73. François Lamarche and Christian Retoré Proof nets for the Lambek calculus – an overview. In Abrusci and Casadio, editors, *Proofs and linguistic categories, proceedings of 1996 Roma Workshop*, pages 241–262. CLUEB, Bologna, 1996.
74. Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.
75. Joachim Lambek. On the calculus of syntactic types. In Roman Jakobson, editor, *Structure of language and its mathematical aspects*, pages 166–178. American Mathematical Society, Providence RI, 1961.
76. Joachim Lambek. Categorial and categorical grammars. In Oehrle et al. [91], pages 297–318.
77. Alain Lecomte. Proof nets and dependencies. In *COLING-92*, pages 394–401. Nantes, August 1992.
78. Alain Lecomte and Christian Retoré. Pomset logic as an alternative categorial grammar. In Glyn V. Morrill and Richard T. Oehrle, editors, *Formal Grammar*, Proceedings of the Conference of the European Summer School in Logic, Language and Information, Barcelona, 1995. pages 181–196.
79. Alain Lecomte and Christian Retoré. Words as modules: a lexicalised grammar in the framework of linear logic proof nets. In Carlos Martin-Vide, editor, *International Conference on Mathematical Linguistics II*. John Benjamins, Amsterdam, 1997.

80. Marie-Ange Légeret. *Algèbres de démonstrations en grammaires catégorielles*. Thèse de Doctorat, spécialité Mathématiques, Université Blaise Pascal, Clermont-Ferrand, janvier 1996.
81. Ralph Loader. Linear logic, totality and full completeness. In *LICS'94*, pages 292–298. IEEE computer society, Washington, 1994.
82. Per Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, Napoli, 1984. (Notes by Giovani Sambin of a series of lectures given in Padua, June 1980)
83. Alexandru Mateescu and Arto Salomaa. *Formal Languages: an Introduction and a Synopsis*, chapter 1, pages 1–40. Volume 1 of Rozenberg and Salomaa [104], 1996.
84. Philip Miller and Thérèse Torris, editors. *Formalismes syntaxiques pour le traitement automatique du langage naturel*. Hermès, Paris, 1990.
85. Michael Moortgat. *Categorial Investigations. Logical and Linguistic Aspects of the Lambek Calculus*. Foris, Dordrecht, 1988.
86. Michael Moortgat. *La grammaire catégorielle généralisée – le calcul de Lambek-Gentzen*, chapter 3, pages 127–182. In Miller and Torris [84], 1990.
87. Michael Moortgat. Categorial type logics. In van Benthem and ter Meulen [19], pages 93–177.
88. Glyn V. Morrill. *Type Logical Grammar*. Kluwer, Dordrecht, 1994.
89. Georges Mounin. *Histoire de la linguistique – des origines au XX^{e} siècle*. Presses Universitaires de France, Paris, 1967.
90. Reinhard Muskens. *Meaning and Partiality*. CSLI, 1996.
91. Richard T. Oehrle, Emmon Bach, and Deirde Wheeler, editors. *Categorial Grammars and Natural Languages Structures*. D. Reidel, Dordrecht, 1988.
92. Barbara Partee. Montague grammar and transformational grammar. *Linguistic Inquiry*, VI(2):203–300, 1975.
93. Matti Pentus. Lambek grammars are context free. In *LICS'93*, pages 371–373. IEEE computer society, Washington, 1993.
94. Fernando C. N. Pereira and Stuart M. Shieber. *Prolog and Natural Language Analysis*, volume 10 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford CA, 1987. (distributed by Cambridge University Press)
95. Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammars*. Chicago University Press, 1994.
96. Jean-Yves Pollock. *Langage et cognition – Une introduction au programme minimaliste de la grammaire générative*. Presses Universitaires de France, Paris, 1997.
97. Aarne Ranta. *Type-Theoretical Grammar*. Oxford University Press, 1994.
98. Hans Reichenbach. *Elements of Symbolic Logic*. The MacMillan Company, New York, 1948.
99. Christian Retoré. *Réseaux et Séquents Ordonnés*. Thèse de Doctorat, spécialité Mathématiques, Université Paris 7, février 1993.
100. Christian Retoré. Calcul de lambek et logique linéaire. *Traitement Automatique des Langues*, 37(2):39–70, 1996.
101. Christian Retoré. Pomset logic: a non-commutative extension of classical linear logic. In J. R. Hindley and Ph. de Groote, editors, *TLCA'97*, volume 1210 of *LNCS*, pages 300–319, 1997.
102. Robert H. Robins. *A short history of linguistics*. Longmans, Green, and Co., London, 1967.
103. Dirk Roorda. *Resource logic: proof theoretical investigations*. PhD thesis, FWI, Universiteit van Amsterdam, 1991.
104. Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of Formal Languages*. Springer, Heidelberg, 1996. (3 volumes)
105. Esa Saarinen, editor. *Game-Theoretical Semantics: Essays on Semantics by Hintikka, Carlson, Peacocke, Rantala, and Saarinen*. D. Reidel, Dordrecht, 1979.

106. Peter Sells. *An introduction to Government-Binding Theory, Generalised Phrase Structure Grammar, and Lexical-Functional Grammar*, volume 3 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford CA, 1985. (distributed by Cambridge University Press)
107. Sebastian Shaumyan. *Applicational grammar as a semantic theory of natural language*. Edinburgh University Press, 1977.
108. Betty Shefts. *Grammatical Method in Pāṇini*. American Oriental Society, New Haven CO, 1961.
109. Stuart M. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343, 1985.
110. Stuart M. Shieber. *An Introduction to Unification-Based Approaches to Grammar*, volume 4 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford CA, 1986. (distributed by Cambridge University Press)
111. Stuart M. Shieber. *Les grammaires basées sur l'unification*, chapter 1, pages 27–86. In Miller and Torris [84], 1990.
112. Mark J. Steedman. *Surface Structure and Interpretation*. Center for the Study of Language and Information, Stanford CA, 1996. (distributed by Cambridge University Press)
113. Joseph E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. MIT Press, Cambridge MA, 1977.
114. Anna Szabolcsi. Bound variables in syntax. In *Proceedings of the 6th Amsterdam Colloquium*, pages 331–351, Amsterdam, 1987. Institute for Language, Logic and Information.
115. Wolfgang Thomas. *Languages, Automata and Logic*, chapter 7, pages 389–456. Volume 3 of Rozenberg and Salomaa [104], 1996.
116. Richmond H. Thomason, editor. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven CO, 1974.
117. Thérèse Torris. *La grammaire syntagmatique généralisée*, chapter 2, pages 87–126. In Miller and Torris [84], 1990.
118. Anne Sjerp Troelstra. *Lectures on Linear Logic*, volume 29 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford CA, 1992. (distributed by Cambridge University Press)
119. Rémi Zajac. Notes on the Typed Feature System, version 4. Technical report, Institut für Informatik, Project Polygloss, Universität Stuttgart, 1991.
120. Wojciech Zielonka. Axiomatizability of Adjukiewicz-Lambek calculus by means of cancellation schemes. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 27:215–224, 1981.