Joachim Lambek

Logic and Grammar

Abstract. Grammar can be formulated as a kind of substructural propositional logic. In support of this claim, we survey bare Gentzen style deductive systems and two kinds of non-commutative linear logic: intuitionistic and compact bilinear logic. We also glance at their categorical refinements.

Keywords: Categorial grammar, Substructural logic, Pregroup grammar, Bilinear logic, Compact bicategories.

The medieval undergraduate university curriculum consisted of seven *liberal arts*. Four of them, the *quadrivium*, comprised pure and applied mathematics and had been inherited from the ancient Pythagorean school. (This may be a surprise to students who enroll in a liberal arts course expecting to avoid mathematics.) The more elementary *trivium* consisted of logic, grammar and rhetoric. Logic has long been accepted as a necessary ingredient of the foundations of mathematics, and grammar too has recently entered the realm of mathematics, even if a majority of linguists have not welcomed this development.

The main difference between formal logic and formal grammar is the absence in the latter of Gentzen's three structural rules: *weakening, contraction* and *interchange*. Gentzen had introduced his so-called *sequents* to represent deductions in intuitionistic logic. We write

$$f : A_1 \cdots A_m \to A_{m+1},$$

where the A_i are formulas or empty strings. (I have replaced Gentzen's original comma by juxtaposition and written 1 for the empty string or just left a blank.) In particular, there were the *identity* axiom $1_A : A \to A$ and the *cut* rule

$$\frac{f : \Lambda \to A \;\; g : \Gamma A \Delta \to B}{g\langle f \rangle : \Gamma \Lambda \Delta \to B} \; .$$

Special issue: Logic and Natural Language
Edited by **Nissim Francez and Ian Pratt-Hartmann**

Studia Logica (2012) 100: 667–681
DOI: 10.1007/s11225-012-9426-7
© Springer 2012

(We use capital Greek letters to denote strings of formulas.) There were three *structural* rules:

$$weakening: \quad \frac{f : \Gamma\Delta \to B}{f_\omega : \Gamma A\Delta \to B} ,$$

$$contraction: \quad \frac{f : \Gamma AA\Delta \to B}{f_c : \Gamma A\Delta \to B} ,$$

$$interchange: \quad \frac{f : \Gamma AB\Delta \to C}{f_i : \Gamma BA\Delta \to C} .$$

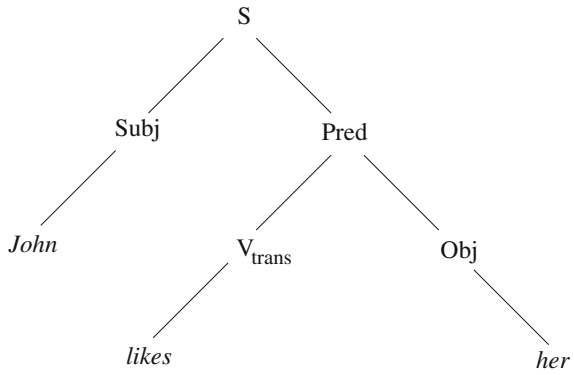
If we ignore these structural rules, Gentzen's sequents may represent *contextfree* derivations in linguistics, although our arrow, replacing Gentzen's symbol $\vdash$, suggests that we are recognizing, rather than generating, sentences. For example, to recognize the sentence

$$John\ likes\ her$$

we might adopt the derivations

$$John \to \text{Subj},\ likes \to V_{\text{trans}},\ her \to \text{Obj},$$
$$V_{\text{trans}}\text{Obj} \to \text{Pred},\ \text{Subj Pred} \to S.$$

The resulting compound derivation may be represented by a tree, usually pictured upside-down:



It is known that contextfree derivations do not suffice to describe certain languages, e.g. Dutch, Swiss German and Bambara. So we are tempted to turn to Gentzen's sequents for classical logic:

$$A_1 \cdots A_m \to B_1 \cdots B_n .$$

In the absence of the classical structural rules (not stated here), these will do service for non-commutative classical bilinear logic, aka *bilinear* logic. However, juxtaposition (replacing Gentzen's comma) on the left stands for the tensor product and juxtaposition on the right for its de Morgan dual, called "par" by linear logicians.

In linguistic applications, a derivation $\Gamma \to \Delta$ is known as a *rewrite* rule, but juxtaposition on the right plays the same role as that on the left. Thus the tensor product and its de Morgan dual should be identified, as it will be in the "compact bilinear logic" to be discussed presently. We thus obtain what linguists call a production grammar and mathematicians call a *semi-Thue system*.

A production grammar was first employed by Panini in the fifth century *BCA* for describing Sanskrit. The above sentence will serve to show how a production grammar works for English.

First, one augments the English vocabulary by a number of grammatical terms, such as the above:

S = declarative sentence,

Subj = subject,

Pred = predicate,

Obj = object,

V_{trans} = transitive verb phrase,

as well as what I have called "inflectors":

P_3 = third person singular,

T_1 = present tense,

Acc = accusative,

C_{13} = present tense third person singular.

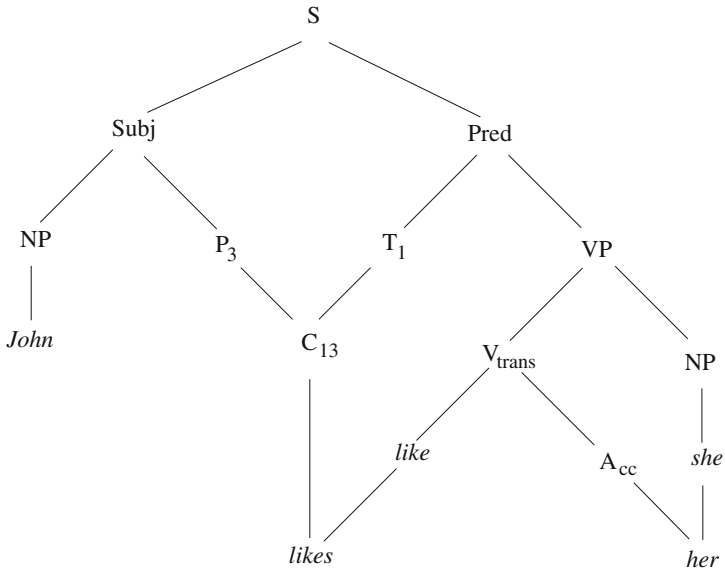
Thus, one adopts not only contextfree rules, but also the morphological rules

$P_3T_1 \to C_{13}$,

C_{13} *like* $\to$ *likes*,

Acc *she* $\to$ *her*.

Finally, one employs the appropriate derivations to show how the string of words *John likes her* reduces to S, as pictured in the following diagram, no longer a tree:



Although we have used the arrow primarily to recognize sentences, the reversed arrow can be used to generate sentences, thus showing that the set of sentences is recursively enumerable. Production grammars are useful for handling morphology, such as verb conjugation as in [LY2008], and for limited semantic fields, e.g. kinship relations as in [BL1995].

Logicians employ sequents subject to structural rules as a basic framework for introducing connectives such as $\Rightarrow$, $\wedge$ or $\vee$. Logical systems have algebraic analogues: Boolean algebras or Heyting algebras for classical or intuitionistic logic respectively. Although linguists have used rewrite systems without structural rules and without connectives to formulate grammars of natural languages, some mathematically inclined linguists prefer to carry out calculations not on words, but on associated *types* (that used to be called "categories"), which live in a substructural logical system or its algebraic equivalent.

The main idea of a "categorial" grammar is to associate with each word or morpheme of the language one or more types, which are listed in the mental dictionary, and then to perform a logical or algebraic calculation on the sequence of types associated with a string of words to check whether it is a well-formed sentence or some other syntactic entity. I myself have worked on two such systems: the *syntactic calculus* [1958] and *compact bilinear logic* [1999b, 2008].

The *syntactic calculus* involved three binary operations, $\otimes, \backslash$ and $/$, satisfying the rules

$$a \otimes b \to c \text{ iff } a \to c/b \text{ iff } b \to a\backslash c.$$

These connectives may be introduced by Gentzen style rules such as $\otimes$ and $/$:

$$\frac{\Gamma \to a \;\; \Delta \to b}{\Gamma\Delta \to a \otimes b} \;,$$

$$\frac{\Lambda b \to a}{\Lambda \to a/b} \;, \quad \frac{\Lambda \to b \;\; \Gamma a \Delta \to c}{\Gamma(a/b)\Lambda\Delta \to c} \;,$$

and similarly for $\backslash$. It was shown in [L1958] that the resulting system satisfies Gentzen's *cut elimination* theorem when the types are freely generated from a set of basic types. It asserts that the cut rule is then redundant, and so is the identity rule, except for basic types. Sometimes the syntactic calculus is augmented by a nullary operation I, satisfying the introduction rules

$$\frac{\Lambda \to 1}{\Lambda \to I} \;, \quad \frac{\Gamma\Lambda \to c}{\Gamma I \Delta \to c}$$

(Recall that 1 denotes the empty string.)

From an algebraic point of view, the resulting system is a *residuated* semigroup, or monoid if the identity element I is admitted, freely generated from the set of basic types. At first [1958], this set was taken to be $\{s, n\}$, s for "sentence" and n for "name" or "noun phrase", but later more basic types were admitted and the set of basic types was allowed to be partially ordered.

Here is an example illustrating the original approach:

$$\begin{array}{c} \textit{John likes her} \\ n \;\; n\backslash(s/n) \;\; (s/n)\backslash s \end{array}$$

which is justified by the cutfree proof, using only the axiom $a \to a$ for basic types:

$$\dfrac{n \to n \quad \dfrac{\dfrac{s \to s}{(s/n)n \to s}}{s/n \to s/n \qquad s \to s}}{\dfrac{n \to n \quad (s/n)((s/n)\backslash s) \to s}{n(n\backslash(s/n))((s/n)\backslash s) \to s}}$$

(The identity rule for s/n can easily be derived from those for s and n.)

Unfortunately, such a calculation may easily overload the temporary storage capacity of the brain. Recall Miller's [1956] observation that our

short-term memory cannot hold more than about seven chunks of information. (This is easily checked by anyone trying to dial a nine-digit telephone number.)

The question had been raised by Noam Chomsky whether languages describable by the syntactic calculus could also be described by a contextfree grammar. This question remained open for many years and was finally answered positively by Mati Pentus [1993]. Perhaps this is a good place to remind the reader not to take the equivalence of two grammars too seriously. It may happen that a computation in one grammar obeys Miller's restriction to seven chunks of information in the short-term memory, but that the corresponding computation in the equivalent grammar exceeds it.

The syntactic calculus had predecessors in the grammars of Ajdukiewicz [1935] and Bar-Hillel [1953]. It has now been around for half a century, but was unable to compete with the more powerful techniques of Chomsky [1957] and his school, which made use of rewrite systems enhanced by transformations. On the other hand, it was recognized that the syntactic calculus had an intriguing affinity to the semantic calculi of Curry [1961] and Montague [1974]. This observation led to further developments by a small but dedicated group of *type logical* grammarians (see e.g. Moortgat [1988,1997]).

Other substructural logics were studied by philosophers, e.g. relevance logic, which dropped the weakening rule, and by computer scientists, e.g. the linear logic of Girard [1987], which retained only the interchange rule. Evidently, the latter system was not useful for syntactic applications, and so Claudia Casadio proposed to turn to non-commutative classical linear logic, now called *bilinear* logic. However, the distinction between the tensor product and its de Morgan dual, called *par* by Girard, seemed to be irrelevant for linguistic purposes. We therefore decided to drop it (see [CL2002]) and turn to what is now known as *compact bilinear* logic and its algebraic presentation in the form of *pregroups*. These are partially ordered monoids in which each element a has both a *left adjoint* a^ℓ and a *right adjoint* a^r satisfying

$$a^\ell a \to 1 \to aa^\ell,\ aa^r \to 1 \to a^r a,$$

Evidently, a pregroup is a residuated monoid if we define

$$a/b = ab^\ell,\ a\backslash b = a^r b.$$

The converse is not the case, since $(ab)/c$ and $a(b/c)$ both correspond to abc^ℓ, in view of the associative law.

The idea of applying pregroups to grammar is again to ascribe to each word of the language one or more types, a *type* now being an element of the

pregroup freely generated by a partially ordered set of basic types. A decision procedure is obtained [L1999b] by realizing that, without loss of generality, one may assume that all contractions $a^\ell a \to 1$ and $aa^r \to 1$ precede all expansions $1 \to aa^\ell$ and $1 \to a^r a$. In particular, if one wants to show that a string of types reduces to s, or any other basic type for that matter, no expansions are needed at all.

This observation does in fact amount to cut elimination, as was shown by Buszkowski [2002]. Thus, in practice, one works with the kind of grammar already advocated by Zellig Harris [1966,1968], although one now has the advantage of being able to use double adjoints. These are useful in contexts where Chomsky [1980] would introduce *traces*, e.g. in questions and relative clauses. Let us just look at a few examples:

$$\begin{array}{c} \textit{John likes her} \\ n(\pi_3^r s_1 o^\ell)\ o \ \to s \end{array}$$

$$\begin{array}{c} \textit{Does John like her ?} \\ (q_1 i^\ell \pi_3^\ell) n (i o^\ell) o \to q \end{array}$$

$$\begin{array}{c} \textit{Whom does John like} -? \\ (q o^{\ell\ell} q^\ell)(q_1 i^\ell \pi_3^\ell) n (i o^\ell) \to q \end{array}$$

Here we have made use of the basic types

n = noun phrase,
π_3 = third person singular,
s = declarative sentence,
s_1 = declarative sentence in present tense,
o = direct object,
q = question,
q_1 = question in present tense,
i = infinitive of intransitive verb phrase.

We also assume

$$n \to \pi_3,\ s_1 \to s,\ q_1 \to q$$

in the partially ordered set of basic types. Note that

$$n\pi_3^r \to \pi_3 \pi_3^r \to 1.$$

The dash in the third sentence above represents a Chomskyan trace. Underlinks indicating contractions had already been used by Harris and correspond to the proofnets of linear logicians.

One advantage of the pregroup analysis over the earlier syntactic calculus is that now proofs can proceed horizontally, as in algebra, and need not fill whole pages or blackboards as proof-trees did before.

If working with one free pregroup offers a decision procedure, the same is true, if we work with the direct product of two or more free pregroups. This allows us, for example, to make separate calculations on syntactic types and *feature* types, as illustrated by the following example from French [L2010]:

$$\begin{array}{llll} \textit{elle} & \textit{veut} & \textit{être} & \textit{embrassée} \\ \pi_3 & (\pi_3^r s_1 j^\ell) & (io^{\ell\ell} p_2^\ell) & (p_2 o^\ell) \qquad \to s_1 \\ \pi_{3f} & & & \pi_{3f}^{'r} \qquad \to 1 \end{array}$$

Here $i \to j$ represent intransitive infinitives, p_2 any intransitive past participle, and $\pi_{3f} \to \pi'_{3f}$ stand for third person singular feminine.

The idea to perform parallel computation on multiple free pregroups had been proposed by Ed Stabler [2008] and Brendon Gillon [oral communication] and was carried out by some of their students, notably Kobele [2008] and Kusalik [2008]. Further developments along these lines are to be expected, for example to deal with natural languages known not to be describable by contextfree grammars. The simplest example of a non-contextfree *formal* language is in fact the intersection of two contextfree ones, hence can be analyzed by parallel computation in two free pregroups, in view of Buszkowski's [2001] proof of the equivalence of free pregroup grammars and contextfree ones.

Pure mathematicians try to prove theorems about an abstract Platonic world, but in proof theory the proofs or deductions themselves become the objects of study. Proof theoretic techniques may also be applied to grammatical derivations. To start with, deductions and derivations may be viewed as arrows in a category or multicategory. (I will disregard the classical polycategories in this article.)

So far, we have used the arrow to denote a partial order. In logic this is deducability, usually denoted by the symbol $\vdash$. From now on we will regard a labelled arrow as an actual deduction and take it to be a morphism in a category, for example a function between sets.

The original Gentzen sequent calculus may be conceived as dealing with functions of many variables. The structural rules then correspond to common manipulation of functions:

$$\frac{f : \Gamma AB\Delta \to C}{f_i : \Gamma BA\Delta \to C}, \qquad \begin{array}{c} f_i(\cdots yx\cdots) = f(\cdots xy\cdots) \\ \text{interchanging variables,} \end{array}$$

$$\frac{f : \Gamma\Delta \to C}{f_w : \Gamma A\Delta \to C}, \qquad \begin{array}{c} f_w(\cdots x\cdots) = f(\cdots\cdots) \\ \text{introducing a superfluous variable,} \end{array}$$

$$\frac{f : \Gamma AA\Delta \to C}{f_c : \Gamma A\Delta \to C}, \qquad \begin{array}{c} f_c(\cdots x\cdots) = f(\cdots xx\cdots) \\ \text{replacing two identical variables by one.} \end{array}$$

We also interpret the identity arrow as the identity function:

$$1_A : A \to A,\ 1_A(x) = x,$$

and Gentzen's cut rule represents substitution:

$$\frac{f : \Lambda \to A\ g : \Gamma A\Delta \to B}{g\langle f\rangle : \Gamma\Lambda\Delta \to B}, \qquad \begin{array}{c} g\langle f\rangle(\cdots\cdots) = g(\cdots f(\cdots)\cdots) \\ \text{substituting a function for a variable.} \end{array}$$

Of course, the structural rules play no role in our substructural grammars and even the identity and cut rules may be eliminated except for the basic types.

Abstractly, the intuitionistic Gentzen systems turn into multicategories (see e.g. [L2004]). The classical Gentzen systems turn into *polycategories*; not to be considered in this article.

We may adopt the functional notation even in the absence of structural rules, interpreting arrows as *multilinear* functions. The easiest way to justify this is to introduce *indeterminate* arrows $x : 1 \to A$ and to define

$$1_A x = x$$

and

$$g\langle f\rangle(\cdots\cdots\cdots) = g(\cdots f(\cdots)\cdots)$$

for $f : \Lambda \to A$ and $g : \Gamma A\Delta \to B$. It is then seen that the arrows of a multicategory must satisfy the following *identity, associativity* and *commutativity* laws:

$$\begin{array}{ll} 1_A\langle f\rangle = f & \text{when } f : \Lambda \to A \\ g\langle 1_A\rangle = g & \text{when } g : \Gamma A\Delta \to B, \\ h\langle g\langle f\rangle\rangle = h\langle g\rangle\langle f\rangle & \text{when } h : \Phi B\Psi \to C \end{array}$$

and f, g as above, but

$$k\langle f\rangle\langle f'\rangle = k\langle f'\rangle\langle f\rangle$$

when $f : \Lambda \to A,\ f' : \Delta \to B$ and $k : \Phi A\Theta B\Psi \to C$.

Following the categorical imperative, we replace partially ordered algebras by categories with additional structure, thus partially ordered monoids

by monoidal categories, residuated monoids by residuated (monoidal) categories and pregroups by compact (monoidal) categories.

Monoidal categories are categories endowed with a binary operation $\otimes$, the tensor product, between objects and an identity object I. These may be introduced in Gentzen style (as pioneered by Bourbaki [1948]), adopting labelled bilinear arrows,

$$m_{AB} : AB \to A \otimes B,\ \iota : 1 \to I$$

and the rules, assigning to each multilinear arrow

$$h : \Gamma AB\Delta \to C \text{ a unique } h' : \Gamma A \otimes B\Delta \to C \text{ such that}$$

$$h'\langle mAB\rangle = h,$$

and to each $h : \Gamma\Delta \to C$ a unique $h' : \Gamma I\Delta \to C$ such that

$$h'\langle\iota\rangle = h.$$

Everything you want to know about $\otimes$ and I follows from these categorized introduction rules. For example, the associativity of the tensor product is obtained with the help of an arrow $\alpha_{ABC} : (A \otimes B) \otimes C \to A \otimes (B \otimes C)$, which is defined below.

When $x : 1 \to A$ and $y : 1 \to B$ are indeterminates, it is convenient to write

$$\{x, y\} = m_{AB}xy.$$

Given $f : A \to B$ and $g : C \to D$, we may then define $f \otimes g : A \otimes B \to C \otimes D$ by putting

$$(f \otimes g)\{x, y\} = \{fx, gy\}$$

and verifying functionality, e.g. that

$$(f \otimes g)(h \otimes k) = fh \otimes gk.$$

The associativity arrow $\alpha_{ABC} : (A \otimes B) \otimes C \to A \otimes (B \otimes C)$ is now defined by putting

$$\alpha_{ABC}\{\{x, y\}, z\} = \{x, \{y, z\}\}.$$

It is easily checked that α is a natural isomorphism which satisfies Mac Lane's [1971] pentagonal condition. (For more details see [L2004]).

In a *residuated* category we have two binary operations on objects, in addition to the tensor product, namely / (*over*) and \ (*under*) to satisfy

$$Hom(B, A\backslash C) \simeq Hom(A \otimes B, C) \simeq Hom(A, C/B).$$

The operation / can be introduced via a multicategory as follows:

$$e_{CB} : (C/B)B \to C,$$

and for all $h : \Lambda B \to C$ there exists a unique

$$h' : \Lambda \to C/B \text{ such that } e_{CB}\langle h' \rangle = h.$$

The operation \ is of course introduced similarly.

Although residuated categories are mentioned here because they provide an insight into the proof theory of the syntactic calculus, they originated in the category of $R - R$-bimodules, R being any ring. In particular, C/B and $B\backslash C$ stood for $\mathrm{Hom}(B_R, C_R)$ and $\mathrm{Hom}({}_RB, {}_RC)$ respectively, viewed as $R - R$-bimodules. In fact, this ring-theoretic application was the original inspiration for the syntactic calculus.

A *compact* monoidal category has, in addition to the tensor product, two unary operations on objects, called *left* and *right adjunction* and satisfying

$$Hom(B, A^r) \simeq Hom(A \otimes B, I) \simeq Hom(A, B^\ell).$$

A ring-theoretical example is the category of all $R - R$-bimodules where R is a *division ring* and the bimodules are *finitely generated* on each side.

These algebraic examples call for an obvious generalization to $R - S$-bimodules, where R and S are two rings, hence for a generalized multicategory in which strings $A_1 \cdots A_m$ are subject to the restriction that each A_i has a *source* and a *target* and that the target of A_i is the source of A_{i+1}. The string $A_1 \cdots A_m$ is then assumed to have the same source as A_1 and the same target as A_m. Even the empty string 1 will have a source and a target, depending on where it occurs.

These generalized multicategories have a direct application to grammar, as is implicit in the proposal by Mishal Brame [1984,1985,1987]. One allows the arrows A_i to be words of a natural language, say English, and the objects serving as sources and targets grammatical terms. For example, one might analyze *the boy likes her* as follows:

$$\begin{array}{cccc} \mathit{the} & \mathit{boy} & \mathit{likes} & \mathit{her}. \\ 1 \to Art & Art \to NP & NP \to V_{trans} & V_{trans} \to S \end{array}$$

Here each English word is viewed as an arrow between grammatical types, e.g. *likes* transforms a noun phrase into a transitive verb phrase.

Generalized multicategories may also be used for studying categories with more than one object. For example, a *bicategory* has been described as a

monoidal category with several objects, although it is more reasonable to say that a monoidal category is a bicategory with one object, better: one 0-cell (see below). In particular, a *strictly* monoidal category, in which the tensor product is associative on the nose, is known as a 2-*category* when it has more than one object. For a direct definition of a 2-category, see Mac Lane[1971]. In bicategories and 2-categories, it is customary to call

$$\begin{aligned} \text{objects} &= 0-\text{cells},\\ \text{arrows between objects} &= 1-\text{cells},\\ \text{arrows between arrows} &= 2-\text{cells}. \end{aligned}$$

Note that, in addition to the expected *vertical* composition of 2-cells, there is a *horizontal* composition:

$$\text{if } f_i : A_i \to B_i \text{ then } f_1 \cdots f_m : A_1 \cdots A_m \to B_1 \cdots B_n.$$

From our point of view, a 2-category is just a categorical refinement of a production grammar.

Of possible interest in grammar are residuated and compact 2-categories. In the latter, each 1-cell has both a left and a right adjoint, in the original sense of "adjoint", as first defined by Peter Freyd.

Examples of residuated and compact bicategories are easily found in algebra. A typical residuated bicategory has

$$\begin{aligned} 0-\text{cells} &= \text{rings}\\ 1-\text{cells} &= \text{bimodules}\\ 2-\text{cells} &= \text{bilinear transformations}. \end{aligned}$$

To obtain a compact bicategory one must also assume that the rings are division rings and that the bimodules are finitely generated on each side.

Returning from grammar and algebra to logic, we note that the Heyting algebras corresponding to the intuitionistic propositional calculus give rise to the cartesian closed categories introduced by Bill Lawvere [1969]. In particular, T becomes the initial object I, $A \wedge B$ becomes the direct product $A \times B$ and $B \Rightarrow C$ becomes the exponential C^B. The free cartesian closed category generated by two objects E (entities) and Ω (truth values) has been used for the semantic calculus studied by Haskell Curry and his followers, although they did not employ the language of category theory. Curry replaced the arrows $I \to A$ by *combinators* avoiding bound variables, while other authors preferred to make use of the lambda-calculus of Alonzo Church who admitted them. Richard Montague [1974] even envisaged the cartesian closed category of sets, albeit an intensional version of it, with Ω

the object of Boolean truth values. More generally, we would expect an elementary topos, also introduced by Lawvere [1972]. Its objects need not be sets, but could be sheaves, and Ω need not be Boolean, allowing the full sway of higher order intuitionistic logic.

An obvious generalization of the original syntactic calculus leads to the residuated bicategory freely generated by two objects S and N. Its interpretation in a topos, here indicated by square brackets, may be viewed as a structure preserving functor (see Benson [1970]) which sends S on to Ω, N on to E, $A \otimes B$ on to $[A \times B] = [A] \times [B]$ and both C/A and $A\backslash C$ on to $[C]^{[A]}$, the set of functions from [A] to [C].

To target semantics in the usual category of sets causes one problem already faced by Montague. It assumes that in the latter one can substitute equals for equals: if Jack is the murderer one could infer that the police know that Jack is the murderer. One way to overcome this difficulty is to allow the topos $\mathcal{T}$ in which the speaker lives to embrace an internal topos $\mathcal{T}'$ representing the world in the speaker's mind. From the truth of a proposition in $\mathcal{T}$ one cannot infer its truth in $\mathcal{T}'$.

How should the internal topos be constructed? Categorists have a fancy way of doing this; but Gödel's process of arithmetization will do as well. For example, the so-called *free topos*, the initial object in the category of all small toposes with natural numbers object, acceptable as the world of mathematics to moderate intuitionists, can be constructed from natural numbers and lives in any topos that contains them.

POSTSCRIPT

While the above account hints at two methods for looking at non-context-free grammars, a promising alternative approach has been proposed by Francez and Kaminski [2007] and adopted by Preller [2010]. It allows the freely generated pregroup to be augmented by a finite set of commuting inequations, yet retains the above decision procedure and is capable of handling mildly context-sensitive languages.

References

[1] Ajdukiewicz, K. 1935, Die Syntaktische Konnexität, *Studia Philosophica* 1:1–27. English translation: Syntactic connection, in S. McCall (ed.), *Polish Logic*, Oxford, Clarendon Press, 1967.

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

[3] Bar-Hillel, Y. 1964, *Language and Information: Selected Essays*, Palo Alto, Addison-Wesley.

[4] Benson, D. B. 1970, Syntax and semantics, a categorical view, *Information and Control* 17:145–166.
[5] Bhargava, M., and J. Lambek 1995, A rewrite system of the Western Pacific: Lounsbury's analysis of Trobriand kinship terminology, *Theoretical Linguistics* 21:241–253.
[6] Bourbaki, N. 1948, *Algèbre linéaire*, Paris, Hermann.
[7] Brame, M. 1984, 1985, 1987, Recursive categorical syntax and morphology I, II, III, *Linguistic Analysis* 14,15,17.
[8] Buszkowski, W. 2001, Lambek grammars based on pregroups, in P. G. De Groote et al. (eds.), *Logical aspects of computational linguistics*, LNAI 2099, Berlin, Springer.
[9] Buszkowski, W. 2002, Cut elimination for the Lambek calculus of adjoints, in V. M. Abrusci et al. (eds.), *New perspectives in logic and formal linguistics*, Rome, Bulzoni.
[10] Casadio, C., and J. Lambek 2002, A tale of four grammars, *Studia Logica* 71:315–329.
[11] Chomsky, N. 1957, *Syntactic Structures*, The Hague, Mouton.
[12] Chomsky, N. 1980, *Rules and representations*, New York, Columbia University Press.
[13] Curry, H. B. 1961, Some logical aspects of grammatical structure, in R. Jacobson (ed.), *Structure of language and its mathematical aspects*, Providence, R.I., AMS.
[14] Francez, N., and M. Kaminski 2007, Commutation-Augmented Pregroup Grammars and Mildly Context-Sensitive Languages, *Studia Logica* 87:295–321.
[15] Girard, J.-Y. 1987, Linear logic, *Theoretical Computer Science* 50:1–102.
[16] Harris, Z. S. 1966, A cyclic cancellation automaton for sentence well-formedness, *International Computation Centre Bulletin* 5:69–94.
[17] Harris, Z. S. 1968, *Mathematical structure of language*, New York, John Wiley and Sons.
[18] Kobele, G. M. 2008, Agreement bottlenecks in Italian, in C. Casadio et al. (eds.), *Computational algebraic approaches to natural languages*, Milan, Polimetrica.
[19] Kusalik, T. 2008, *Product pregroups as an alternative to inflectors*, in C. Casadio et al. (eds.), *Computational algebraic approaches to natural languages*, Milan, Polimetrica.
[20] Lambek, J. 1958, The mathematics of sentence structure, *American Mathematical Monthly* 65:154–170.
[21] Lambek, J. 1999a, Deductive systems and categories in linguistics, in H. J. Ohlbach et al. (eds.), *Logic, Language and Reasoning*, Dordrecht, Kluwer.
[22] Lambek, J. 1999b, Type grammar revisited, in Lecomte et al. (eds.), *Logical aspects of computational linguistics* LNCS/LNAI 1582, Berlin, Springer.
[23] Lambek, J. 2004, Bicategories in algebra and linguistics, in T. Ehrhard et al. (eds.), *Linear logic in computer science*, London Mathematical Society Lecture Notes Series 316, Cambridge University Press.
[24] Lambek, J. 2008, *From word to sentence*, Milan, Polimetrica.
[25] Lambek, J. 2010, Exploring feature agreement in French with parallel pregroup computations, *Journal of Logic, Language and Information* 19:75–88.
[26] Lambek, J., and N. S. Yanofsky 2008, A computational approach to Biblical Hebrew, in C. Casadio et al. (eds.), *Computational algebraic approaches to natural languages*, Milan, Polimetrica.

[27] Lawvere, F. W., *Toposes, Algebraic Geometry and Logic*, Lecture Note in Mathematics 274, New York, Springer.
[28] Miller, G. A. 1956, The magical number seven plus or minus two, *Psychological Review* 63:81–97.
[29] Mac Lane, S. 1971, *Categories for the working mathematician*, New York, Springer.
[30] Montague, R. 1974, *Formal Philosophy: Selected Papers*, New Haven, Yale University Press.
[31] Moortgat, M. 1988, *Categorial Investigations*, Dordrecht, Foris.
[32] Moortgat, M. 1997, Categorial type logics, in J. van Benthem et al. (eds.), *Handbook of Logic and Language*, Amsterdam, Elsevier.
[33] Pentus, M. 1993, Lambek grammars are context free, in *Proceedings of the 8th annual symposium of logic in Computer Science*, pp. 429–433.
[34] Preller, A. 2010, Polynomial pregroup grammars pursue context sensitive languages, *Linguistic Analysis* 36:483.
[35] Stabler, E. P. 2008, Tupled pregroup grammars, in C. Casadio et al. (eds.), *Computational algebraic approaches to natural languages*, Milan, Polimetrica.

Joachim Lambek
Department of Mathematics and Statistics
McGill University
805 Sherbrooke Street West
Montreal, Canada
lambek@math.mcgill.ca