Lambek Calculus and Substructural Logics

Wojciech Buszkowski
Adam Mickiewicz University in Poznań
University of Warmia and Mazury in Olsztyn

1. Introduction

Substructural logics are nonclassical logics whose sequent systems avoid some structural rules, e.g Exchange ($a \cdot b = b \cdot a$ in algebraic setting), Contraction ($a \leq a \cdot a$), and Weakening ($a \cdot b \leq a$), which occur in standard sequent systems of classical and intuitionistic logics. These logics have a long tradition (Schroeder-Heister and Dosen, 1993); well-known substructural logics are relevant logics (omit Weakening) and some many-valued logics (omit Contraction). More recently, linear logics, introduced by Girard (1987), admit Exchange (commutative versions) or drop the three structural rules (noncommutative versions). Algebraic methods in substructural logics were developed in (Ono and Komori, 1985) and many other works; see the recent book (Galatos *et al.*, 2007).

Syntactic Calculus of Lambek (1958) is a typical noncommutative substructural logic, intended to provide nontrivial transformations of syntactic types. From the modern perspective, this calculus, now called the Lambek calculus, is equivalent to the intuitionistic noncommutative Multiplicative Linear Logic (precisely, it is true for the Lambek calculus with unit). Its commutative version (with Exchange) is the Lambek-van Benthem calculus of semantic types, corresponding via the Curry-Howard isomorphism to linear lambda terms, not containing closed subterms (van Benthem, 1991). The nonassociative Lambek calculus, introduced by Lambek (1961), even lacks the associativity rule, whence the antecedents of sequents are trees, not simply sequences, of formulas (types). This system is natural for the representation of linguistic expressions in the form of trees (phrase structures). The nonassociative Lambek calculus is regarded as a basic type logic, which can be further extended by new operations, e.g. modalities, allowing a controlled usage of certain structural rules (Moortgat, 1997). There are natural connections between the Lambek calculus and category theory (Lambek, 1995). The logic of pregroups - an extension of Syntac-

Linguistic Analysis, 36 1-4

P.O. Box 2237, Vashon, WA 98070

tic Calculus, due to Lambek (1999)) - is closely related to compact 2-categories (Preller and Lambek, 2007).

The mathematical research on Lambek systems revived in the early 1980-ties. The present author and his colleagues from Poznań (W. Zielonka, M. Kandulski, and others) investigated logical and computational properties of Lambek systems and categorial grammars, based on them (Lambek grammars); some properties of classical categorial grammars and other type grammars were also studied. At the same time, the Dutch school (J. van Benthem, F. Zwarts, M. Moortgat, and their students) investigated Lambek systems in relation to the lambda calculus and natural language semantics; there were developments toward general logics of information processing (van Benthem, 1986, 1991, 2005). Moortgat (1996) and Morrill (1994) elaborated different modal versions of Lambek systems, with interesting applications in language description. Some parallel research was done by scholars adhering to the tradition of Montague Grammar (B. Hall-Partee, E. Bach, R. T. Oehrle) and the Curry-Shaumyan combinatory grammars (M. Steedman, A. Szabolcsi). This early period is well documented in collection volumes (Buszkowski *et al.*, 1988; Oehrle *et al.*, 1988).

Further research was partially influenced by the origin of linear logics and extensive studies on them in the early 1990-ties. Also conversely, some fundamental ideas of linear logics seem to have been anticipated (if not directly influenced) by the earlier work on Lambek systems. For instance, Girard's usage of exponentials, which reintroduces structural rules (under control) in linear logic, influences Moortgat's and Morrill's application of modalities. On the other hand, Girard's proof of the completeness of Propositional Linear Logic with respect to phase-space semantics is a refinement of the proof of completeness of the product-free Lambek calculus with respect to powerset frames over free semigroups, given in (Buszkowski, 1986a) and its commutative version in (van Benthem, 1988a). The literature on Lambek systems and categorial grammars shows many other links of this kind. The present paper aims to discuss some of them.

To keep this paper in a reasonable size, I omit many interesting topics, e.g. Curry-Howard isomorphism, Abstract Categorial Grammars (an explicit application of the typed lambda-calculus as a grammar formalism, due to de Groote (2001)), proof nets (a graph-theoretic representation of proofs in multiplicative linear logics),

modal categorial grammars, combinatory grammars, and learning theory. For some information on these topics see survey articles (Moortgat, 1997; Buszkowski, 1997, 2003b); also see the books cited above, the collection (Casadio *et al.*, 2005), and two special issues of *Studia Logica*: 71.3 (2002) and 87.2-3 (2007).

Section 2 presents different Lambek systems and their algebraic models. It also provides some basic linguistic interpretations of type logics. Section 3 introduces sequent systems of type logics, discusses cut elimination and its consequences, and shows some applications of sequent systems in proofs of fine completeness theorems. Section 4 is concerned with categorial grammars, based on different type logics; we show how certain proof-theoretic tools (cut elimination, normalization, interpolation) are used to establish basic properties of Lambek systems and grammars (decidability, generative capacity, complexity). Section 5 briefly outlines some ways in the opposite direction: some results on Lambek systems and grammars help to solve problems in substructural logics.

2. Lambek calculi and their algebraic models

Lambek (1958) introduced Syntactic Calculus as a formal logic of syntactic types, an extension of the type reduction system underlying categorial grammars, due to Ajdukiewicz (1935). The calculus is referred to as (Associative) Lambek Calculus (**L**). Its nonassociative version, introduced in (Lambek, 1961), is called Nonassociative Lambek Calculus (**NL**). The latter is a pure logic of residuation triple: a basic (residuated) operation $\cdot$ with two residual operations $\backslash$, $/$, which satisfy the residuation law:

$$\text{(RES)}\ a \cdot b \leq c \text{ iff } b \leq a\backslash c \text{ iff } a \leq c/b\,,$$

where $\leq$ is a partial ordering. One refers to $\cdot$ as *product* and to $\backslash$ (resp. $/$) as the *right* (resp. *left*) residual operation for product.

In algebraic terms, **NL** corresponds to *residuated groupoids*, i.e. ordered algebras $(M, \leq, \cdot, \backslash, /)$ such that $(M, \leq)$ is a poset, and $\cdot, \backslash, /$ are binary operations on M, satisfying (RES). **L** corresponds to *residuated semigroups*, i.e. residuated groupoids in which product is associative. In what follows, $\mathcal{M}$ (possibly with subscripts etc.) denotes an algebra and M its universe.

Every residuated groupoid fulfills the following monotonicity laws:

(MON1) if $a \leq b$ then $c \cdot a \leq c \cdot b$ and $a \cdot c \leq b \cdot c$,

(MON2) if $a \leq b$ then $c\backslash a \leq c\backslash b$ and $b\backslash c \leq a\backslash c$,

(MON3) if $a \leq b$ then $a/c \leq b/c$ and $c/b \leq c/a$,

for all elements a, b, c. We often write ab for $a \cdot b$.

The algebraic form of **NL** can be presented as follows. *Types* are formed out of variables (also called: *atomic types*) by means of three operation symbols $\cdot, \backslash, /$. Types amount to *formulas* of our formal systems. Atomic types are denoted by $p, q, r, \ldots$ and arbitrary types by $A, B, C, \ldots$. *Simple sequents* are formal expressions $A \Rightarrow B$. The axioms are:

$$\text{(Id)}\ A \Rightarrow A\,,$$

and the inference rules are:

$$\text{(RES-R1)}\ \frac{A \cdot B \Rightarrow C}{B \Rightarrow A\backslash C}\,,\quad \text{(RES-R2)}\ \frac{B \Rightarrow A\backslash C}{A \cdot B \Rightarrow C}\,,$$

$$\text{(RES-R3)}\ \frac{A \cdot B \Rightarrow C}{A \Rightarrow C/B}\,,\quad \text{(RES-R4)}\ \frac{A \Rightarrow C/B}{A \cdot B \Rightarrow C}\,,$$

$$\text{(S-CUT)}\ \frac{A \Rightarrow B\,;\ B \Rightarrow C}{A \Rightarrow C}\,.$$

$\Rightarrow$ is interpreted as $\leq$. The algebraic form of **L** is obtained by affixing two new axioms, which express the associative law:

$$\text{(As1)}\ (A \cdot B) \cdot C \Rightarrow A \cdot (B \cdot C)\,,\ \text{(As2)}\ A \cdot (B \cdot C) \Rightarrow (A \cdot B) \cdot C\,.$$

The following rules are derivable in both **NL** and **L**:

$$\text{(MON-R1)}\ \frac{A \Rightarrow B}{C \cdot A \Rightarrow C \cdot B}\,,\quad \text{(MON-R1')}\ \frac{A \Rightarrow B}{A \cdot C \Rightarrow B \cdot C}\,,$$

$$\text{(MON-R2)}\ \frac{A \rightarrow B}{C\backslash A \Rightarrow C\backslash B}\,,\quad \text{(MON-R2')}\ \frac{A \Rightarrow B}{B\backslash C \Rightarrow A\backslash C}\,,$$

$$\text{(MON-R3)}\ \frac{A \Rightarrow B}{A/C \Rightarrow B/C}\,,\quad \text{(MON-R3')}\ \frac{A \Rightarrow B}{C/B \Rightarrow C/A}\,.$$

Recall that a rule $\frac{\Phi}{S}$, where Φ is a set of sequents and S is a sequent, is *derivable* in a logic $\mathcal{L}$, if S is provable in $\mathcal{L}$ enriched with all sequents from Φ as *assumptions* (in other words: S is provable from Φ in $\mathcal{L}$). We write $\Phi \vdash_{\mathcal{L}} S$ for: S is provable from Φ in $\mathcal{L}$. So $\vdash_{\mathcal{L}} S$ means that S is provable in the pure $\mathcal{L}$ (without any assumptions). Assumptions are concrete sequents; one does not require that Φ is closed under substitutions (of arbitrary types for variables). On the contrary, the set of *axioms* is always required to be closed under substitutions. For instance, (Id), (As1) and (As2) represent all possible sequents of the above form.

We list several laws provable in **L**.

(L1) $A \cdot (A\backslash B) \Rightarrow B$, $(B/A) \cdot A \Rightarrow B$ (application laws),

(L2) $(A\backslash B) \cdot (B\backslash C) \Rightarrow A\backslash C$, $(A/B) \cdot (B/C) \Rightarrow A/C$ (composition laws),

(L3) $A\backslash B \Rightarrow (C\backslash A)\backslash(C\backslash B)$, $A/B \Rightarrow (A/C)/(B/C)$ (Geach laws),

(L4) $A \Rightarrow (B/A)\backslash B$, $A \Rightarrow B/(A\backslash B)$ (type-raising laws),

(L5) $A \Rightarrow B\backslash(B \cdot A)$, $A \Rightarrow (A \cdot B)/B$ (expansion laws),

(L6) $(A\backslash B) \cdot C \Rightarrow A\backslash(B \cdot C)$, $A \cdot (B/C) \Rightarrow (A \cdot B)/C$ (switching laws).

Only (L1), (L4) and (L5) are provable in **NL**. To prove (L1) apply (RES-R2) (resp. (RES-R4) to (Id) $A\backslash B \Rightarrow A\backslash B$ (resp. $B/A \Rightarrow B/A$). (L2) follow from (L1), using (MON-R1), associativity and (RES-R1), (RES-R3). The remaining proofs are left to the reader.

Let us briefly discuss the linguistic meaning of these laws. The basic interpretation refers to syntax. Σ is the lexicon of some language. Sentences and other phrases are represented as strings of words from Σ. Types represent certain sets of strings, corresponding to syntactic categories. Atomic types represent some basic categories, e.g. s - the category of declarative sentences (statements), n - the category of proper nouns, N - the category of common nouns. Complex types represent functor categories. A string x is of type $A\backslash B$ (resp. B/A) if and only if, for any y of type A, the concatenation yx (resp. xy) is of type B; one says that x is a left-looking (resp. right-looking) functor from category A to category B. It means that we interpret $\backslash$ and $/$ as in frames $P(\Sigma^+)$; see below. Also z is of type $A \cdot B$ if and only if $z = xy$, for some x of type A and y of type B.

So $n\backslash s$ represents the category of verb phrases, $(n\backslash s)/n$ the category of transitive verb phrases, $s/(n\backslash s)$ the category of (full) noun phrases, $(s/(n\backslash s))/N$ the category of determiners, and so on. Here we ignore agreement, flexion etc. To regard these features a finer typing is necessary. For instance, s_1 for statements in present tense, s_2 for statements in past tense, n_i, $i = 1, 2, 3$, for subjects in the i−th person. Then, 'likes' is of type $n_3\backslash s_1$, 'like' is of types $n_1\backslash s_1$ and $n_2\backslash s_1$, and 'liked' is of types $n_i\backslash s_2$, for $i = 1, 2, 3$. One may use n as a general type of subjects, assuming $n_i \Rightarrow n$, for $i = 1, 2, 3$, and similarly s as a general type of statements, assuming $s_i \Rightarrow s$, for $i = 1, 2$. (This naturally leads to Lambek calculi with assumptions.) Then, 'liked' can be assigned type $n\backslash s_2$, and $n\backslash s_2 \Rightarrow n\backslash s$ is provable, by (MON-R2).

Application laws (L1) correspond to reduction laws, underlying classical categorial grammars of (Bar-Hillel *et al.*, 1960), tracing back to the type reduction procedure of (Ajdukiewicz, 1935). If types are interpreted in semantic domains, then they express the basic fact that a function $f : A \mapsto B$ applied to an argument $a \in A$ yields the value $f(a) \in B$.

The semantical interpretation of composition laws (L2) is the following: if $f : A \mapsto B$ and $g : B \mapsto C$ then $g \circ f : A \mapsto C$. In syntax they allow some transformations of tree structures. For instance, the sentential negation 'not' of type s/s can also be connected with a noun phrase of type $s/(n\backslash s)$ to yield a negative noun phrase: $(s/s) \cdot (s/(n\backslash s)) \Rightarrow s/(n\backslash s)$. Thus 'not every student sings' can be parsed as (not ((every student) sings)) and ((not (every student)) sings). (L3) express the same phenomenon by means of type lifting.

Type-raising laws (L4) are also referred to as Montague laws, since in Montague Grammar the type of entities e is raised to the type of noun phrases $((e, t), t)$ in contexts like 'John and some student' in order to represent the meaning of 'and' as the boolean meet on noun phrase denotations. Types e, t are semantical counterparts of n, s, and (a, b) represents both $a\backslash b$ and b/a.

(L5) and (L6) contain product. Types of main syntactic categories are product-free, whence one cannot find any natural, simple illustration of these laws. Types with product play an essential role in theoretical investigations of algebraic models and in some more advanced proof-theoretic and computational results. Let us notice that (L5), (L6) cannot be reversed. Residuated monoids satisfying the reverses of (L6) are precisely pregroups (algebras underlying

pregroup grammars of (Lambek, 1999), which we discuss later on).

A model is a pair $(\mathcal{M}, \mu)$ such that $\mathcal{M}$ is an algebra and μ is a mapping from the set of atomic types to M, called *a valuation*. Every valuation μ is uniquely extended to a mapping from the set of types to M by setting:

$$\mu(A \cdot B) = \mu(A) \cdot \mu(B)\,,\ \mu(A \backslash B) = \mu(A) \backslash \mu(B)\,,\ \mu(A / B) = \mu(A) / \mu(B)\,.$$

Here we assume that $\mathcal{M}$ admits operations $\cdot, \backslash, /$. A sequent $A \Rightarrow B$ is *true* in a model $(\mathcal{M}, \mu)$, if $\mu(A) \leq \mu(B)$ (here $\leq$ is the designated partial ordering in $\mathcal{M}$). It is *valid* in $\mathcal{M}$, if it is true in $(\mathcal{M}, \mu)$, for any valuation μ. It is valid in a class of algebras C, if it is valid in all algebras from C. A set of sequents Φ *emtails* a sequent S with respect to C, if S is true in all models $(\mathcal{M}, \mu)$ such that $\mathcal{M} \in C$ and all sequents from Φ are true in $(\mathcal{M}, \mu)$.

NL is *strongly complete* with respect to the class of residuated groupoids $\mathcal{RG}$: for any set of sequents Φ and any sequent S, $\Phi \vdash_{\mathbf{NL}} S$ if and only if Φ entails S with respect to $\mathcal{RG}$. Consequently, **NL** is *weakly complete* with respect to $\mathcal{RG}$: the sequents provable in **NL** are precisely the sequents valid in $\mathcal{RG}$. The 'only if' part of the equivalence (soundness) is easy to prove: the axioms of **NL** are valid in $\mathcal{RG}$, and every inference rule preserves the truth in $(\mathcal{M}, \mu)$, for $\mathcal{M} \in \mathcal{RG}$. The 'if' part (completeness) is proved by a routine construction of the Lindenbaum-Tarski algebra. **L** is strongly complete with respect to the class of residuated semigroups $\mathcal{RS}$. Analogous completeness theorems are true for other systems, considered in this paper, and their routine proofs are omitted.

In linguistics, standard frames are *algebras of languages*. Let Σ be an alphabet (not necessarily finite). Σ^* (resp. Σ^+) denotes the set of all (resp. nonempty) finite strings on Σ. The empty string is denoted by ϵ. The powerset $P(\Sigma^*)$ (resp. $P(\Sigma^+)$) consists of all (resp. $\epsilon-$free) *languages* on Σ. For $L_1, L_2 \subseteq \Sigma^+$, one defines the operations $\cdot, \backslash, /$ as follows:

$$L_1 \cdot L_2 = \{ab : a \in L_1,\, b \in L_2\}\,,$$

$$L_1 \backslash L_2 = \{c \in \Sigma^+ : L_1 \cdot \{c\} \subseteq L_2\}\,,\ L_1 / L_2 = \{c \in \Sigma^+ : \{c\} \cdot L_2 \subseteq L_1\}\,.$$

Here ab denotes the concatenation of strings a and b. It is easy to show that $(P(\Sigma^+), \subseteq, \cdot, \backslash, /)$ is a residuated semigroup. We refer to it as the residuated semigroup $P(\Sigma^+)$. A similar construction yields

the residuated monoid $P(\Sigma^*)$; one writes $c \in \Sigma^*$ in definitions of $\backslash, /$. The set $\{\epsilon\}$ is the unit element for $\cdot$.

In general, *a monoid* is an algebra $(M, \cdot, 1)$ such that $(M, \cdot)$ is a semigroup, and $1 \in M$ satisfies: $1 \cdot a = a = a \cdot 1$, for all $a \in M$, and *a residuated monoid* is an algebra $(M, \leq, \cdot, \backslash, /, 1)$ such that $(M, \cdot, 1)$ is a monoid and $(M, \leq, \cdot, \backslash, /)$ is a residuated semigroup.

In algebraic terms, Σ^+ with concatenation (resp. Σ^* with concatenation and ϵ) is the free semigroup (resp. the free monoid) generated by Σ. The constructions of $P(\Sigma^+)$, $P(\Sigma^*)$ can be generalized: given an arbitrary semigroup (resp. monoid) $\mathcal{M}$, the residuated semigroup (resp. monoid) $P(\mathcal{M})$ consists of all subsets of M, ordered by inclusion, with operations defined as above, for $L_1, L_2 \subseteq M$ (write $c \in M$ in definitions of $\backslash, /$); for the case of monoids, $\{1\}$ is the unit element of $P(\mathcal{M})$, where 1 is the unit element of $\mathcal{M}$. If one starts from a groupoid $\mathcal{M} = (M, \cdot)$ (resp. a unital groupoid $\mathcal{M} = (M, \cdot, 1)$), then she obtains the residuated groupoid (resp. residuated unital groupoid) $P(\mathcal{M})$.

The free groupoid generated by Σ consists of bracketed strings on Σ, i.e. the smallest set containing Σ and being closed under the bracketed concatenation $X, Y \mapsto (X, Y)$. Bracketed strings on Σ can be imagined as finite binary trees whose leaves are labeled by symbols from Σ; so we denote the set of all such strings by Σ^T. Admitting the empty tree Λ, we obtain the free unital groupoid generated by Σ, denoted Σ^{T*}. One assumes $(\Lambda, X) = X = (X, \Lambda)$, for any $X \in \Sigma^{T*}$. As above, one constructs the residuated groupoid $P(\Sigma^T)$ and the residuated unital groupoid $P(\Sigma^{T*})$; clearly $\{\Lambda\}$ is the unit element of the latter.

By the completeness theorems, discussed above, all sequents provable in **L** are valid in residuated semigroups $P(\Sigma^+)$, and all sequents provable in **NL** are valid in residuated groupoids $P(\Sigma^T)$. Interestingly, Pentus (1995) shows that **L** yields all sequents valid in the frames $P(\Sigma^+)$ (even restricted to finite alphabets); in other words, **L** is weakly complete with respect to the class of frames $P(\Sigma^+)$ such that Σ is finite. The strong completeness fails: the assumption $p \Rightarrow p \cdot p$ entails $p \Rightarrow q$ in the class of frames $P(\Sigma^+)$, since $\mu(p) = \emptyset$, for any valuation μ, fulfilling $\mu(p) \subseteq \mu(p) \cdot \mu(p)$. On the other hand, $p \Rightarrow q$ is not provable from $p \Rightarrow p \cdot p$ in **L**, since the former is not entailed by the latter with respect to $\mathcal{RS}$ (consider frames $P(\Sigma^*)$). **L** is strongly complete with respect to powerset algebras $P(\mathcal{M})$ such that $\mathcal{M}$ is a semigroup (Buszkowski, 1986a).

NL is not weakly complete with respect to the class of frames $P(\Sigma^T)$; the sequent $((p \cdot q)/r) \cdot r \Rightarrow p \cdot r$ is valid in this class, but it is not provable in **NL** (it is not valid in $\mathcal{RG}$; see the next section for another proof).

NL1 and **L1** are extensions of **NL** and **L**, respectively, by the constant 1 and additional axioms:

$$\text{(Ax1)}\ 1 \cdot A \Rightarrow A\,,\ A \Rightarrow 1 \cdot A\,,\ A \cdot 1 \Rightarrow A\,,\ 1 \Rightarrow A \cdot 1\,.$$

NL1 (resp. **L1**) is strongly complete with respect to residuated unital groupoids (resp. residuated monoids).

To abbreviate notation we use $A \Leftrightarrow B$ as a shortening of two sequents $A \Rightarrow B$ and $B \Rightarrow A$. (Ax1) can be expressed by $1 \cdot A \Leftrightarrow A$ and $A \cdot 1 \Leftrightarrow A$. The following sequents are provable in **L1** ((L7)-(L9) also in **NL1**).

$$\text{(L7)}\ 1 \Rightarrow A\backslash A\,,\ 1 \Rightarrow A/A\,,$$

$$\text{(L8)}\ A \Leftrightarrow 1\backslash A\,,\ A \Leftrightarrow A/1\,,$$

$$\text{(L9)}\ A \Leftrightarrow (A/A)\backslash A\,,\ A \Leftrightarrow A/(A\backslash A)\,,$$

$$\text{(L10)}\ A\backslash A \Leftrightarrow (A\backslash A)\backslash(A\backslash A)\,,\ A/A \Leftrightarrow (A/A)/(A/A)\,.$$

NL1 and **L1** are more similar to standard logical calculi than **NL** and **L**, since the former admit provable formulas: types A such that $1 \Rightarrow A$ is provable, which is not the case for the latter. Some authors regard systems with 1 as linguistically less adequate than their 1-free companions. If adjectives are assigned type N/N, then 'very' can be assigned type $(N/N)/(N/N)$. These two types are equivalent in **L1**, whence 'very' is sent to N/N as well, which does not work. Remember, however, that the empty string ϵ is of type 1 (in frames $P(\Sigma^*)$), so 'very' does not belong to the category $(N/N)/(N/N)$ (it does not form an adjectival phrase, if applied to ϵ). A correct typing would be more cumbersome than for the 1-free case.

At the end of this section we briefly describe algebras, corresponding to different extensions of Lambek calculi.

A bilinear algebra is a residuated monoid with an element 0, satisfying $a = (0/a)\backslash 0 = 0/(a\backslash 0)$, for any element a. Bilinear algebras are models of Bilinear Logic **BL**. One defines two negations $a^r = a\backslash 0$, $a^l = 0/a$; they satisfy $a = (a^l)^r = (a^r)^l$, a version of the double negation law. A bilinear algebra is said to be

cyclic, if $a\backslash 0 = 0/a$, for any element a; so $a^r = a^l$, and this element is denoted by $\neg a$. The double negation law $\neg\neg a = a$ holds in cyclic bilinear algebras. Bilinear Logic amounts to Noncommutative Multiplicative Linear Logic (Noncommutative **MLL**). The equation $(a^l b^l)^r = (a^r b^r)^l$ is valid in bilinear algebras; this element is denoted by $b \oplus a$, and the operation $\oplus$ is called *par*. We use a notation close to (Lambek, 1995); the community of Linear Logic prefers another notation; in particular, their $\oplus$ amounts to our $\vee$, and they use $\otimes$ for our product. Residuals can be defined in terms of 'par' and negations: $a\backslash b = a^r \oplus b$, $b/a = b \oplus a^l$. Conversely, product and par can be defined in terms of $\backslash, /, 0$:

$$a \otimes b = (0/a \cdot b)\backslash 0 = ((0/b)/a)\backslash 0\,,\ a \oplus b = (0/a)\backslash b\,.$$

Besides multiplicative operations $\cdot, \oplus, \backslash, /$, one also admits additive operations $\wedge, \vee$, interpreted as lattice meet and join. *A residuated lattice* is a residuated monoid $\mathcal{M}$ such that $(M, \leq)$ is a lattice (one can also consider more general classes of lattice-ordered residuated semigroups, groupoids and unital groupoids). Noncommutative Multiplicative-Additive Linear Logic (Noncommutative **MALL**) is the logic of residuated lattices with 0, being bilinear algebras; this logic has been introduced and studied by Abrusci (1991). **MLL** corresponds to commutative bilinear algebras, and **MALL** to commutative bilinear algebras with lattice operations. (Commutativity means the commutative law $ab = ba$ for product; then $a\backslash b = b/a$, and this element is denoted by $a \to b$.) Propositional Linear Logic **PLL** of Girard (1987) is **MALL** with exponentials ! and ?. In algebras, $!a$ (*ofcourse a*) is the greatest element $x \leq a$ such that $xx = x$, and $?a$ (*whynot a*) is the least element $x \geq a$ such that $x \oplus x = x$. **L** (resp. **NL**) with $\wedge, \vee$ is called Full (resp. Nonassociative) Lambek Calculus and denoted **FL** (resp. **FNL**); some authors assume that these systems contain 1 and 0; see e.g. (Galatos *et al.*, 2007). Recently, Moortgat (2009) studies symmetric algebras, with $\otimes, \oplus$ and their residuals, but without negations; the corresponding logics are referred to as Lambek-Grishin calculi.

Types with $\wedge$ were used by Lambek (1961) in order to replace a finite set of types, assigned to a word by a type grammar, with a single type. In (Kanazawa, 1992) syntactic types are augmented with 'features'; e.g. 'walks' is assigned type $(\mathrm{np} \wedge \mathrm{sing})\backslash \mathrm{s}$, 'walk' type $(\mathrm{np} \wedge \mathrm{pl})\backslash \mathrm{s}$, and 'became' type $(\mathrm{np}\backslash \mathrm{s})/(\mathrm{np} \vee \mathrm{ap})$, where np is the type of noun phrase, s of sentence, ap of adjectival phrase, and sing, pl

represent 'singular' and 'plural'. Subtyping also shows some possible application of $\vee$; if s_1, s_2 are types of sentence in present tense and past tense, respectively, then one can define $s = s_1 \vee s_2$, whence $s_1 \Rightarrow s$ and $s_2 \Rightarrow s$ become provable.

Bilinear algebras in which $\otimes$ equals $\oplus$ are called pregroups (a simpler definition is given below). Pregroups are models of Compact Bilinear Logic (**CBL**), underlying pregroup grammars of Lambek Lambek (1999).

A pregroup can be defined as an algebra $(M, \leq, \cdot, ^l, ^r, 1)$ such that $(M, \leq, \cdot, 1)$ is a partially ordered monoid, and $^l, ^r$ are unary operations on M, satisfying the adjoint laws:

$$\text{(Ad-l) } a^l a \leq 1 \leq a a^l\,, \quad \text{(Ad-r) } a a^r \leq 1 \leq a^r a\,,$$

for all $a \in M$. The element a^l (resp. a^r) is called *the left* (resp. *right*) *adjoint* of a. In any pregroup one defines $a\backslash b = a^r b$, $b/a = b a^l$, and easily proves (RES). Hence any pregroup is a residuated monoid. However, the powerset frame $P(\mathcal{M})$, where $\mathcal{M}$ is a monid, cannot be expanded to a pregroup, in general. Concrete pregroups can be constructed as sets of order-preserving functions over a poset; every pregroup is isomorphic to a pregroup of this form Buszkowski (2001).

For any element a of a pregroup and any integer n, one defines $a^{(n)} = a^{r\ldots r}$, if $n > 0$, and $a^{(n)} = a^{l\ldots l}$, if $n < 0$, where the adjoint operation is iterated $|n|$ times; also $a^{(0)} = a$. The following laws are valid in pregroups: $1^{(n)} = 1$, $a^{(n)} a^{(n+1)} \leq 1 \leq a^{(n+1)} a^{(n)}$, $(ab)^{(n)} = a^{(n)} b^{(n)}$, if n is even, $(ab)^{(n)} = b^{(n)} a^{(n)}$, if n is odd. Also $a \leq b$ iff $a^{(n)} \leq b^{(n)}$, if n is even, and $a \leq b$ iff $b^{(n)} \leq a^{(n)}$, if n is odd.

Pregroups grammars of Lambek (1999) are based on the calculus of free pregroups, referred to as **CBL**. Let $(P, \leq)$ be a finite poset. Elements of P are treated as atomic types. *Terms* are expressions $p^{(n)}$ such that $p \in P$ and n is an integer. *Types* are finite strings of terms. One defines a relation $\Rightarrow$ on the set of types as the reflexive and transitive closure of the relation defined by the clauses:

(CON) $\Gamma, p^{(n)}, p^{(n+1)}, \Gamma' \Rightarrow \Gamma, \Gamma'$,

(EXP) $\Gamma, \Gamma' \Rightarrow \Gamma, p^{(n+1)}, p^{(n)}, \Gamma'$,

(IND) $\Gamma, p^{(n)}, \Gamma' \Rightarrow \Gamma, q^{(n)}, \Gamma'$, if either n is even and $p \leq q$, or n is odd and $q \leq p$,

called Contraction, Expansion and Induced Step, respectively. So $\Gamma_1 \Rightarrow \Gamma_2$ holds if and only if Γ_1 reduces to Γ_2 by a finite number of applications of (CON), (EXP) and (IND). The completeness theorem holds: $\Gamma \Rightarrow \Delta$ if and only if $f(\Gamma) \leq f(\Delta)$, for any pregroup $\mathcal{M}$ and any order preserving mapping from $(P, \leq)$ to $\mathcal{M}$ (f can uniquely be extended to a homomorphism of the algebra of types into $\mathcal{M}$ in an obvious way; one sets $f(\epsilon) = 1$).

In pregroup grammars, one assigns pregroup types to expressions. These types are usually direct translations of **L**-types, applying the above definition of $\backslash$, / in pregroups. For instance, type $n\backslash s$ is translated into $n^r s$ (equal to $n^{(1)}s$), and $s/(n\backslash s)$ into $s(n^r s)^l$, equal to $ss^l n$, i.e. $ss^{(-1)}n$. Actually, all linguistic phenomena, described by Lambek and his collaborators in terms of pregroup types, can also be described in terms of **L**-types modulo this translation. The advantage of pregroup grammars is their low computational complexity. **CBL** is polynomial, while **L** is NP-complete. We return to this question in Section 4. Their disadvantage seems to be the lack of any natural relation to type-theoretic semantics and other substructural logics. **CBL** is stronger than **L1**, e.g. $(p \cdot q)/r \Rightarrow p \cdot (q/r)$, and $(p/((q/q)/p))/p \Rightarrow p$ are valid in pregroups, but not in residuated monoids, whence they are provable in **CBL**, but not in **L1**. It is not clear how this extra-power can influence syntactic analysis. A rich collection of pregroup types of English words with a fine analysis of syntax can be found in (Lambek, 2008). Some extended pregroup grammars have also been studied; see e.g. (Francez and Kaminski, 2007; Kiślak-Malinowska, 2007).

We omit the algebraic forms of **FL**, **FNL**, **MLL** and related systems; sequential systems of some of them will be given in the next section.

3. Sequential systems

Gentzen-style sequential systems of **L** and **NL** were proposed in (Lambek, 1958, 1961). First, we present the system of **L**.

Greek capitals $\Gamma, \Delta, \Phi, \ldots$ denote finite sequences of types (but we exclude Σ; it is reserved for alphabets). *Sequents* are of the form $\Gamma \Rightarrow A$. In the antecedents of sequents we write $A_1, \ldots, A_n$ for $(A_1, \ldots, A_n)$ and Γ, Δ for the concatenation of Γ and Δ (but we write $\Gamma\Delta$, if the concatenation does not appear in the antecedent of

a sequent).

L operates on sequents $\Gamma \Rightarrow A$ with Γ nonempty. The axioms are (Id), and the inference rules are:

$$(\mathrm{L}\cdot)\ \frac{\Gamma, A, B, \Delta \Rightarrow C}{\Gamma, A \cdot B, \Delta \Rightarrow C}\,,\quad (\mathrm{R}\cdot)\ \frac{\Gamma \Rightarrow A\,;\ \Delta \Rightarrow B}{\Gamma, \Delta \Rightarrow A \cdot B}\,,$$

$$(\mathrm{L}\backslash)\ \frac{\Gamma, B, \Delta \Rightarrow C\,;\ \Phi \Rightarrow A}{\Gamma, \Phi, A\backslash B, \Delta \Rightarrow C}\,,\quad (\mathrm{R}\backslash)\ \frac{A, \Gamma \Rightarrow B}{\Gamma \Rightarrow A\backslash B}\,,$$

$$(\mathrm{L}/)\ \frac{\Gamma, A, \Delta \Rightarrow C\,;\ \Phi \Rightarrow B}{\Gamma, A/B, \Phi, \Delta \Rightarrow C}\,,\quad (\mathrm{R}/)\ \frac{\Gamma, B \Rightarrow A}{\Gamma \Rightarrow A/B}\,,$$

$$(\mathrm{CUT})\ \frac{\Gamma, A, \Gamma' \Rightarrow B\,;\ \Delta \Rightarrow A}{\Gamma, \Delta, \Gamma' \Rightarrow B}\,,$$

where Γ is nonempty in rules (R$\backslash$) and (R/). Dropping this constraint yields a stronger system **L***.

The both axiomatization of **L** are equivalent. Let $F(\Gamma)$ denote the type arising from Γ after one has replaced each comma by $\cdot$ and introduced parentheses (their order is not essential, by associativity). Then, $\Gamma \Rightarrow A$ is provable in the sequential system of **L** if and only if $F(\Gamma) \Rightarrow A$ is provable in the algebraic system of **L**. The 'only if' part can be proved by induction on derivations in the sequential system (using monotonicity rules, derivable in the algebraic system). For the 'if' part, notice that $\Gamma \Rightarrow F(\Gamma)$ is provable in the sequential system, by (R$\cdot$). Then, $\Gamma \Rightarrow A$ is provable in the sequential system if and only if $F(\Gamma) \Rightarrow A$ is so (use (CUT) and (L$\cdot$)); actually, $\Gamma \Rightarrow A$ is derivable from $F(\Gamma) \Rightarrow A$ in this system, and conversely. Now, by induction on derivations in the algebraic system, one shows that every simple sequent provable in the algebraic system is also provable in the sequential system, which yields our thesis.

Hereafter sequential systems will be our standard presentations of type logics. They can easily be related to algebraic models. An assignment μ in $\mathcal{M}$ can be extended for nonempty sequences Γ, by setting $\mu(\Gamma) = \mu(F(\Gamma))$. Then, $\Gamma \Rightarrow A$ is true in $\mathcal{M}$ under μ if, and only if, $\mu(\Gamma) \leq \mu(A)$. It follows that the sequential system of **L** is (strongly) complete with respect to residuated semigroups.

It is expedient to enrich **L*** by the constant 1 with one new axiom and one new rule:

$$(\mathrm{L1})\ \frac{\Gamma, \Delta \Rightarrow A}{\Gamma, 1, \Delta \Rightarrow A}\,,\quad (\mathrm{R1})\ \Rightarrow 1\,,$$

which yields the sequential system of **L1**. We set: $\mu(\epsilon) = \mu(1) = 1$, for any assignment μ in a residuated monoid. We also set $F(\epsilon) = 1$. As above, one can show that $\Gamma \Rightarrow A$ is provable in the sequential system of **L1** if, and only if, $F(\Gamma) \Rightarrow A$ is provable in the algebraic system of **L1**. We show below that **L*** and **L1** prove the same 1-free sequents; so **L*** is a *conservative* fragment of **L1**, and consequently it is complete with respect to residuated monoids (the strong completeness also holds, but the proof is more involved).

To prove that **L*** is a conservative fragment of **L1**, we need the cut elimination theorem for **L1**. Let **S** be a sequential system with (CUT). By $\mathbf{S}^-$ we denote its subsystem without (CUT). We say that **S** *admits cut elimination*, if every sequent provable in **S** is already provable in $\mathbf{S}^-$ (hence both systems yield the same provable sequents). The following theorem is due to Lambek (1958).

Theorem 3.1. **L** *admits cut elimination.*

It suffices to prove that (CUT) is *admissible* in $\mathbf{L}^-$, it means: if the premises are provable in this system, then the conclusion is so. A direct syntactic proof proceeds by triple induction: (1) on the cut-formula A, (2) on the derivation of the left premise, (3) on the derivation of the right premise. Algebraic proofs can be found in (Galatos *et al.*, 2007). Systems **L*** and **L1** also admit cut elimination.

Every instance of an introduction rule fulfills the following conditions: (PS1) all formulas appearing in the premises are subformulas of formulas appearing in the conclusion, (PS2) the complexity of the conclusion is greater than the complexity of any premise. The complexity of a formula (sequent) can be defined as the total number of occurrences of $\cdot, \backslash, /, 1$ in this formula (sequent). The cut elimination theorem for **L**, **L***, **L1** has two important consequences:

Subformula Property. Every sequent provable in the system has a (cut-free) proof which involves only subformulas of formulas appearing in this sequent.

Decidability. The provability problems for **L**, **L***, **L1** are decidable.

By Subformula Property **L*** is a conservative subsystem of **L1**. Similarly, the product-free **L** is a conservative fragment of the full **L**, and so for **L***, **L1**.

As an illustration of the proof-search procedure, based on cut elimination, we show that $p/(q/q) \Rightarrow p$ is not provable in **L** (it is

provable in **L***). This sequent is not an axiom and cannot be inferred by any introduction rule; the only candidate is (L/), but the right premise must have the empty antecedent, which is impossible.

The sequential system of **NL**, due to Lambek (1961), operates with sequents $X \Rightarrow A$ such that X is a tree on the set of types, and A is a type. By $X[-]$ we denote a tree in which one leaf is labeled by a special symbol '$-$'; $X[Y]$ denotes the tree obtained from $X[-]$ by inserting Y as the subtree rising from the vertex, labeled by '$-$' in $X[-]$. The axioms of **NL** are (Id), and the inference rules are:

$$(\mathrm{L}\cdot\mathrm{N})\ \frac{X[(A,B)] \Rightarrow C}{X[A\cdot B] \Rightarrow C}\,,\quad (\mathrm{R}\cdot\mathrm{N})\ \frac{X \Rightarrow A\,;\ Y \Rightarrow B}{(X,Y) \Rightarrow A\cdot B}\,,$$

$$(\mathrm{L}\setminus\mathrm{N})\ \frac{X[B] \Rightarrow C\,;\ Y \Rightarrow A}{X[(Y,A\setminus B)] \Rightarrow C}\,,\quad (\mathrm{R}\setminus\mathrm{N})\ \frac{(A,X) \Rightarrow B}{X \Rightarrow A\setminus B}\,,$$

$$(\mathrm{L}/\mathrm{N})\ \frac{X[B] \Rightarrow C\,;\ Y \Rightarrow A}{X[(B/A,Y)] \Rightarrow C}\,,\quad (\mathrm{R}/\mathrm{N})\ \frac{(X,A) \Rightarrow B}{X \Rightarrow B/A}\,,$$

$$(\text{CUT-N})\ \frac{X[A] \Rightarrow B\,;\ Y \Rightarrow A}{X[Y] \Rightarrow B}\,.$$

$F(X)$ is the type arising from the tree X, after one has replaced each comma by $\cdot$ (parentheses are saved). Again, $X \Rightarrow A$ is derivable from $F(X) \Rightarrow A$ in the latter system, and conversely. The sequential system of **NL** is equivalent to the algebraic system of **NL**: $X \Rightarrow A$ is provable in the former if, and only if, $F(X) \Rightarrow A$ is provable in the latter. In models, any assignment μ is extended for trees, by setting: $\mu(X) = \mu(F(X))$. $X \Rightarrow A$ is true in $(\mathcal{M},\mu)$ if, and only if, $\mu(X) \leq \mu(A)$. Thus, the sequential system of **NL** is strongly complete with respect to residuated groupoids.

Lambek (1961) proved Theorem 3.1 for **NL**. Consequently, **NL** possesses the subformula property and is decidable.

As an illustration, we show that $((p\cdot q)/r, r) \Rightarrow p\cdot r$ is not provable in **NL**. This sequent is not an axiom. It can be the conclusion of two rules: (1) (R·N) with premises $(p\cdot q)/r \Rightarrow p$ and $r \Rightarrow r$, (2) (L/N) with premises $p\cdot q \Rightarrow p\cdot r$ and $r \Rightarrow r$. For (1), the first premise is not an axiom and cannot be inferred by any rule. Then, this branch of proof-search fails. For (2), the first premise is not an axiom; it can be inferred by (L·N) with the premise $(p,q) \Rightarrow p\cdot r$. The latter sequent can be inferred by (R·N) with premises $p \Rightarrow p$ and $q \Rightarrow r$. But $q \Rightarrow r$ is not an axiom and cannot be inferred by any rule. Our proof search fails.

We have shown that **NL** is not complete with respect to algebras $P(\Sigma^T)$. It is not known whether the set of sequents valid in these algebras is recursively enumerable.

NL* admits sequents of the form $\Lambda \Rightarrow A$, written $\Rightarrow A$, and the axioms and rules of **NL**, extended to empty trees X, Y. In particular, from $A \Rightarrow B$ one can infer $\Rightarrow A\backslash B$, by (R\N), since A can be represented as (A, Λ). The sequential system of **NL1** is **NL***, enriched with the constant 1, one new rule and one new axiom:

$$\text{(L1N)}\ \frac{X[\Lambda] \Rightarrow A}{X[1] \Rightarrow A}\,,\ \text{(R1N)}\ \Rightarrow 1\,.$$

(L1N) can be replaced by two rules:

$$\text{(L1Nl)}\ \frac{X[Y] \Rightarrow A}{X[(1, Y)] \Rightarrow A}\,,\ \text{(L1Nr)}\ \frac{X[Y] \Rightarrow A}{X[(Y, 1)] \Rightarrow A}\,.$$

These systems admit cut elimination and are decidable. The same holds for the sequential systems of **FL** and **FNL**, which arise from those of **L** and **NL**, respectively, extended for formulas with $\wedge, \vee$, by affixing the following rules:

$$\text{(L}\wedge\text{)}\ \frac{\Gamma, A_i, \Delta \Rightarrow B}{\Gamma, A_1 \wedge A_2, \Delta \Rightarrow B}\,,\ \text{(R}\wedge\text{)}\ \frac{\Gamma \Rightarrow A\,;\ \Gamma \Rightarrow B}{\Gamma \Rightarrow A \wedge B}\,,$$

$$\text{(L}\vee\text{)}\ \frac{\Gamma, A, \Delta \Rightarrow C\,;\ \Gamma, B, \Delta \Rightarrow C}{\Gamma, A \vee B, \Delta \Rightarrow C}\,,\ \text{(R}\vee\text{)}\ \frac{\Gamma \Rightarrow A_i}{\Gamma \Rightarrow A_1 \vee A_2}\,,$$

where $i \in \{1, 2\}$.

$$\text{(L}\wedge\text{N)}\ \frac{X[A_i] \Rightarrow B}{X[A_1 \wedge A_2] \Rightarrow B}\,,\ \text{(R}\wedge\text{N)}\ \frac{X \Rightarrow A\,;\ X \Rightarrow B}{X \Rightarrow A \wedge B}\,,$$

$$\text{(L}\vee\text{N)}\ \frac{X[A] \Rightarrow C\,;\ X[B] \Rightarrow C}{X[A \vee B] \Rightarrow C}\,,\ \text{(R}\vee\text{N)}\ \frac{X \Rightarrow A_i}{X \Rightarrow A_1 \vee A_2}\,.$$

FL (resp. **FNL**) is strongly complete with respect to lattice-ordered residuated semigroups (resp. groupoids). Admitting empty antecedents and affixing 1 with (L 1), (R 1), for the associative case, and (L 1N), (R 1N), for the nonassociative case, one obtains systems **FL1** and **FNL1** strongly complete with respect to residuated lattices and lattice-ordered residuated unital groupoids, respectively.

One can also add constants $\bot, \top$, interpreted as the lower bound and the upper bound of the lattice. The corresponding axioms (for the associative case) are:

$$\text{(Ax}\bot\text{)}\ \Gamma, \bot, \Gamma' \Rightarrow A\,,\ \text{(Ax}\top\text{)}\ \Gamma \Rightarrow \top\,,$$

and the nonassociative versions are similar.

The Lambek-van Benthem Calculus is **L** enriched with the exchange rule:

$$(\mathrm{EXC})\ \frac{\Gamma, A, B, \Gamma' \Rightarrow C}{\Gamma, B, A, \Gamma' \Rightarrow C}\,.$$

We denote this system by $\mathbf{L}_e$, and similarly for other systems with (EXC). Actually, van Benthem (1986, 1988b) offered the product-free $\mathbf{L}_e$ as a basic logic of semantic types. The following sequents:

$$A \cdot B \Leftrightarrow B \cdot A\,,\ A\backslash B \Leftrightarrow B/A$$

are provable in this system. Therefore $\backslash, /$ can be collapsed into one operation $\rightarrow$ (implication), and van Benthem writes (a, b) for $a \rightarrow b$ (following a standard type-theoretic notation). $\mathbf{L}_e$ is strongly complete with respect to commutative residuated semigroups.

Type logics can also be axiomatized as *natural deduction systems* (ND-systems). For every logical operation (constant), one admits two inference rules: one for introduction and one for elimination. We only present ND-systems for the $(/)$−fragment of **L**. Again sequents are of the form $\Gamma \Rightarrow A$ with $\Gamma \neq \epsilon$. The axioms are (Id), the introduction rule equals (R/), but the elimination rule (E/) is different from (L/).

$$(\mathrm{E}/)\ \frac{\Gamma \Rightarrow B/A\,;\ \Delta \Rightarrow A}{\Gamma, \Delta \Rightarrow B}\,.$$

The rule (CUT) is admissible in this system. Using this fact, one easily shows that this system is equivalent to the sequential system. The proof of cut elimination is simpler, since inference rules do not introduce any new formula in the antecedents of sequents. The price of simplification is less strength. The formula A disappears in the conclusion of (E/), whence cut elimination does not directly yield Subformula Property (it holds for ND-systems, which follows from their equivalence with sequential systems, studied carefully).

Proofs in ND-systems are closely related to typed lambda-terms by Curry-Howard isomorphism. It will not be discussed in this paper. We only note that this isomorphism underlies interesting studies on semantic readings of sentences, parsed by means of Lambek grammars (van Benthem, 1986, 1988b,a, 1991; Moortgat, 1988; Buszkowski, 1997).

Inference rules of ND-systems never introduce any logical operation or constant in the antecedent; they only perform some algebraic operation on sequences of types, e.g. concatenation in (E/). This property is crucial for *labeled deductive systems* (LDSs) of Gabbay (1996), which generalize ND-systems, in a sense. LDSs are quite natural as a machinery for formal grammars. Given typing $x : A$ and $y : A\backslash B$, one can deduce $xy : B$, which resembles the Curry-style typing in lambda calculus (here the labels x, y are strings of words). We do not discuss this subject in more detail; see e.g. (Buszkowski, 1997).

Using sequential systems or ND-systems for the product-free **L**, **L*** and their versions with $\wedge$ (also with Exchange, Weakening and other structural rules), one can prove their strong completeness with respect to standard frames $P(\Sigma^+)$, $P(\Sigma^*)$ ($\wedge$ is interpreted as the intersection of sets of strings). Given a set of simple sequents Φ, one defines a frame $\mathcal{M}(\Phi)$ whose universe is the set of product-free types, and $\mu(p) = \{\Gamma : \Phi \vdash_{\mathcal{L}} \Gamma \Rightarrow p\}$, for any atomic type p. By induction on A, one proves $\mu(A) = \{\Gamma : \Phi \vdash_{\mathcal{L}} \Gamma \Rightarrow A\}$, for any type A (remind that the logical operations in A are $\backslash, /, \wedge$ or less). Consequently, all sequents from Φ are true in $(\mathcal{M}, \mu)$. If $\Gamma \Rightarrow A$ is not provable from Φ in $\mathcal{L}$, then $\Gamma \Rightarrow A$ is not true in $(\mathcal{M}, \mu)$, which yields the strong completeness; see (Buszkowski, 1986a)

Interestingly, the same construction and, essentially, the same argument yield the strong completeness of (resp. Noncommutative) **MALL** with respect to (resp. noncommutative) phase space models, first proved in (Girard, 1987) and (Abrusci, 1991). Studied more carefully, these methods also yield Finite Model Property (FMP) of Lambek calculi and exponential-free linear logics. Precisely, FMP of **MALL** directly follows from FMP of the product-free $\mathbf{L}_e^*$ with $\wedge$ and $\top$, since the former is faithfully interpretable in the latter, and similar results hold for Cyclic Noncommutative **MALL** and **L*** with 0 and the new rule: from $\Gamma, \Gamma' \Rightarrow 0$ infer $\Gamma', \Gamma \Rightarrow 0$; see (Buszkowski, 2002, 2008).

A faithful interpretation of **MALL** in the product-free $\mathbf{L}_e^*$ with $\wedge, \top$ follows from the completeness of **MALL** with respect to phase space models and the completeness of $\mathbf{L}_e^*$ with $\wedge, \top$ with respect to frames $P(\Sigma^*)$. *A phase space* is the powerset frame $P(\mathcal{M})$ over a commutative monoid $\mathcal{M}$ with a distinguished subset $0^{\mathcal{M}} \subseteq M$. A set $U \subseteq M$ is called *a fact*, if $U = V \rightarrow 0^M$, for some $V \subseteq M$. The family of all facts forms a lattice-ordered commutative bilin-

ear algebra. **MALL** is (strongly) complete with respect to algebras, constructed in this way (the operations are defined in terms of $\rightarrow$ and negation; see Section 2). It follows that $I(p) = p \rightarrow 0$, where 0 is a distinguished atomic type, together with the appropriate definitions of linear operations and constants, yields the desired interpretation. The same method yields a faithful interpretation of **MLL** in the product-free $\mathbf{L}^*_e$ (the latter system is equivalen to the logic **BCI**).

At the end of this section, we discuss a sequential systems of **CBL**. Sequents are of the form $\Gamma \Rightarrow \Delta$, where Γ and Δ are pregroup types, i.e. finite sequences of terms. The axioms are:

$$\text{(Id-P)}\ p^{(n)} \Rightarrow p^{(n)}\,,$$

and the inference rules, clearly corresponding to (CON), (EXP) and (IND), respectively, are:

$$\text{(CON-R)}\ \frac{\Gamma, \Gamma' \Rightarrow \Delta}{\Gamma, p^{(n)}, p^{(n+1)}, \Gamma' \Rightarrow \Delta}\,,\ \text{(EXP-R)}\ \frac{\Gamma \Rightarrow \Delta, \Delta'}{\Gamma \Rightarrow \Delta, p^{(n+1)}, p^{(n)}, \Delta'}\,,$$

$$\text{(IND-R1)}\ \frac{\Gamma, q^{(n)}, \Gamma' \Rightarrow \Delta}{\Gamma, p^{(n)}, \Gamma' \Rightarrow \Delta}\,,\ \text{(IND-R2)}\ \frac{\Gamma \Rightarrow \Delta, p^{(n)}, \Delta'}{\Gamma \Rightarrow \Delta, q^{(n)}, \Delta'}\,,$$

where either n is even and $p \leq q$, or n is odd and $q \leq p$. The appropriate version of the cut rule is:

$$\text{(TRAN)}\ \frac{\Gamma \Rightarrow \Delta\,;\ \Delta \Rightarrow \Gamma'}{\Gamma \Rightarrow \Gamma'}.$$

As shown in (Buszkowski, 2003a), this system admits cut elimination: (TRAN) is admissible in the system, devoid of this rule; also see (Buszkowski, 2007b), where a more general system is studied. The cut-elimination theorem for the sequential system of **CBL** is equivalent to Lambek Switching Lemma (or: Lambek Normalization Theorem), first proved in (Lambek, 1999):

Theorem 3.2. *If* $\Gamma \Rightarrow \Delta$ *is provable in* **CBL***, then there exists a type* Γ' *such that* $\Gamma \Rightarrow \Gamma'$ *can be proved by without (EXP), and* $\Gamma' \Rightarrow \Delta$ *can be proved without (CON).*

Actually, Lambek's result is a bit stronger, but the original formulation needs auxiliary rules of Generalized Contraction and Generalized Expansion, which we omit here. Theorem 3.2 directly follows from cut-elimination for the above sequential system: since

every rule has only one premise, then a cut-free proof of $\Gamma \Rightarrow \Delta$ can apply all left rules before all right rules. Conversely, the cut-elimination theorem directly follows from Theorem 3.2.

By Theorem 3.2, if $\Gamma \Rightarrow t$ is provable in **CBL** and t is a term or the empty sequence, then $\Gamma \Rightarrow t$ can be proved without (EXP) (without (EXP-R)). This yields the polynomial time complexity of **CBL**.

Although **BL** can be presented as a similar sequential system, the interpretation of $\Gamma \Rightarrow \Delta$ is different. While Γ is interpreted as above, i.e. commas represent product, commas in Δ represent the dual operation 'par'. Accordingly, (TRAN) is not a correct rule of **BL**. The cut-elimination theorem for **BL** and its versions with additives can be proved for one-side sequential systems only; see (Abrusci, 1991).

4. Categorial grammars

Let $\mathcal{L}$ be a type logic (in the form of a sequential system). *A categorial grammar* based on $\mathcal{L}$ (shortly: an $\mathcal{L}$–grammar) is a tuple $G = (\Sigma, I, A, \mathcal{L})$ such that Σ is a nonempty finite alphabet, I is a mapping which assigns finite sets of types to elements of Σ, and A is a designated type, called *the principal type* of G. Usually A is an atomic type, often denoted by s. The mapping I is called *the initial type assignment* of G (or: the type lexicon of G). Sometimes one also specifies a finite set P of atomic types and assumes that all types under consideration are formed out of atoms from P. In examples we write '$a : A$ in G' for $A \in I(a)$. We also denote by Σ_G, I_G, and A_G the alphabet (lexicon), the initial type assignment and the principal type of G, respectively.

Let G be an $\mathcal{L}$–grammar, and let $x \in (\Sigma_G)^+$, $x = a_1 \dots a_n$. We say that G assigns type A to x (write: $x \mapsto_G A$), if there exist types $A_i \in I(a_i)$, $i = 1, \dots, n$, such that $A_1, \dots, A_n \Rightarrow A$ is provable in $\mathcal{L}$. By $L_G(A)$ we denote the set of all $x \in (\Sigma_G)^+$ such that $x \mapsto_G A$; it is called *the category* of type A determined by G. $L_G(A_G)$ is called *the language* of G and denoted $L(G)$. Two grammars G_1, G_2 are said to be *equivalent*, if $L(G_1) = L(G_2)$ (this notion can be applied to grammars from different classes, not necessarily categorial grammars). For a class of grammars $\mathcal{G}$, by $L(\mathcal{G})$ we denote the family of all $L(G)$ such that $G \in \mathcal{G}$. One says that $\mathcal{G}_1$ and $\mathcal{G}_2$ are equivalent,

if $L(\mathcal{G}_1) = L(\mathcal{G}_2)$.

In the above definitions we assume that types are assigned to strings. Often it is natural to assign types to tree structures. For instance, in nonassociative systems, e.g. **NL**, **FNL**, sequents are of the form $X \Rightarrow A$, where X is a tree on the set of types. If X is an ordered tree, then $s(X)$ denotes *the yield* of X, i.e. the string of all (labels of) leaves of X, ordered from the left to the right. By '$A_1, \ldots, A_n \Rightarrow A$ is provable in **NL**' we mean that there exists a sequent $X \Rightarrow A$ such that $\vdash_{\mathbf{NL}} X \Rightarrow A$ and $s(X) = A_1 \ldots A_n$, and similarly for other nonassociative systems. We can also assign types directly to trees on Σ. For an **NL**-grammar G and $Y \in (\Sigma_G)^T$, we write $Y \mapsto_G A$, if there exists a sequent $X \Rightarrow A$ such that $\vdash_{\mathbf{NL}} X \Rightarrow A$ and X arises from Y by replacing each leaf label a by some type $A \in I_G(a)$. $L^T_G(A)$ denotes the set of all Y such that $Y \mapsto_G A$. The set $L^T_G(A_G)$ is called *the tree language* of G and denoted $L^T(G)$. Two grammars G_1, G_2 are called $T-$equivalent, if $L^T(G_1) = L^T(G_2)$. The $T-$equivalence of classes of grammars is defined similarly as above.

Let G be a categorial grammar. P_G denotes the set of atomic types appearing in A_G and I_G (sometimes P_G is explicitly given in the declaration of G). $T(G)$ denotes the set of all types on P_G, T_G the set of all types appearing in I_G, and T^s_G the set of all subtypes of types from T_G and A_G. (Usually A_G is a subtype of a type appearing in I_G, but we also admit abnormal options.)

If Subformula Property holds for $\mathcal{L}$ (it is the case for all systems, considered above), then any deduction confirming $x \mapsto_G A$, where G is an $\mathcal{L}-$grammar, employs types from T^s_G and subtypes of A only. Thus, deductive parsing procedures are restricted to finitely many types, effectively computed from G and A.

All standard type logics are decidable. Hence the languages and the tree languages of standard categorial grammars are recursive. In most cases the string languages are context-free. Below we discuss this matter in more detail.

Classical categorial grammars, introduced in (Bar-Hillel *et al.*, 1960), are restricted to $(\backslash, /)-$types and use a simple reduction procedure, based on reduction laws:

$$\text{(AP-1) } A, A\backslash B \Rightarrow B\,, \quad \text{(AP-2) } B/A, A \Rightarrow B\,;$$

these sequents correspond to application laws (L1). A string of types Γ reduces to A, if A results from Γ by applying (AP-1), (AP-2)

finitely many times, always to adjacent types. This rewriting system is equivalent to the fragment of **L**, restricted to (Id) (for product-free types) and rules (L\), (L/) ((CUT) is admissible). The resulting logic is denoted **AB** after K. Ajdukiewicz and Y. Bar-Hillel. So classical categorial grammars are also referred to as **AB**-grammars. The following theorem was proved in (Bar-Hillel *et al.*, 1960).

Theorem 4.1. **AB**-*grammars are equivalent to* ϵ*−free context-free grammars.*

The restriction to ϵ−free languages is characteristic for categorial grammars: neither $L(G)$ contains ϵ, nor $L^T(G)$ contains Λ. It seems unnatural to assume that $\epsilon \mapsto_G A$ if and only if $\Rightarrow A$ is provable in the type logic. Adopting this definition, one must choose a principal type A such that $\Rightarrow A$ is provable in order to obtain a language containing ϵ, e.g. A of the form B/B, but such a choice influences other strings in the language; for an **L**-grammar, we obtain $xx \in L(G)$, for any $x \in L(G)$ (by (L2)), which may be undesirable.

In the proof of Theorem 4.1, it is easy to show that every **AB**-grammar is equivalent to a CFG. Actually, **AB**-grammars can be treated as certain CFGs in Chomsky Normal Form: we identify types from T_G^s with nonterminals, take A_G as the start symbol, reverse arrows in (AP-1), (AP-2) (writing $\mapsto$ for $\Rightarrow$) and add lexical rules $A \mapsto a$ for $A \in I(a)$. It is more difficult to show that any ϵ−free CFG is equivalent to an **AB-**-grammar. Bar-Hillel *et al.* (1960) prove: every ϵ−free CFG G_1 can be transformed into an equivalent **AB**-grammar G_2 such that all types in T_{G_2} are of the form $p, p/q, (p/q)/r$ (p, q, r are atomic) and $s_G \in P_G$.

This result is equivalent to the Greibach Normal Form theorem for CFGs. Precisely, it is equivalent to the statement that every ϵ−free CFG G_1 can be transformed to a CFG G_2 in 2-GNF. Given an **AB**-grammar G_2 with types restricted as above, one obtains an equivalent CFG in 2-GNF, admitting production rules: $p \mapsto arq$ for $a : (p/q)/r$ in G_2, $p \mapsto aq$ for $a : p/q$ in G_2, and $p \mapsto a$ for $a : p$ in G_2. Conversely, given a CFG in 2-GNF, one obtains an equivalent **AB**-grammar by an obvious reversal of the preceding transformation. The both grammars yield the same language, which can be show by an easy induction on the length of strings.

AB-grammars naturally provide some tree structures on strings. *Functor-argument structures* (shortly: FA-structures) on Σ are recursively defined as follows: (i) all elements of Σ are (atomic) FA-

structures, (ii) if X, Y are FA-structures, then $(X, Y)_1$ and $(X, Y)_2$ are FA-structures. In $(X, Y)_1$ (resp. $(X, Y)_2$), the substructure X (resp. Y) is *the functor*, and the other substructure is *the argument*. Σ^F denotes the set of all FA-structures on Σ. **AB**-reductions assign types to FA-structures on Σ_G, according to the following rules:

(FA1) if $a : A$ in G, then $a \mapsto_G A$,

(FA2) if $X \mapsto_G A$ and $Y \mapsto_G A\backslash B$, then $(X, Y)_2 \mapsto_G B$,

(FA3) if $X \mapsto_G B/A$ and $Y \mapsto_G A$, then $(X, Y)_1 \mapsto_G B$.

We define $L^F_G(A) = \{X \in \Sigma^F : X \mapsto_G A\}$ and $L^F(G) = L^F_G(A_G)$; $L^F(G)$ is called *the FA-language* of G. *The tree language* of G, denoted by $L^T(G)$, arises from $L^F(G)$ by dropping functor indices in FA-structures.

Every CFG in Chomsky Normal Form generates a tree language: it consists of all derivation trees of the generated terminal strings (to obtain trees in Σ^T one omits all labels of internal nodes). A tree language $L \subseteq \Sigma^T$ determines a congruence $\sim_L$ on Σ^T: $X \sim_L Y$ iff, for any context $Z[-]$, $Z[X] \in L$ iff $Z[Y] \in L$; an analogous relation can be defined for $L \subseteq \Sigma^F$. It is well known that a tree language $L \subseteq \Sigma^T$ is generated by some CFG (in Chomsky Normal Form) with alphabet Σ if and only if $\sim_L$ is of finite index. The tree languages of finite index are precisely the *regular* tree languages.

A similar characterization of FA-languages and tree languages of **AB**-grammars was obtained in (Buszkowski, 1986c). An FA-language $L \subseteq \Sigma^F$ is the FA-language of some **AB**-grammar if and only if it satisfies the following conditions: (i) $\sim_L$ is of finite index, (ii) all functor paths in FA-structures from L are of a bounded length. A tree language $L \subseteq \Sigma^T$ is the tree language of some **AB**-grammar if, and only if, it satisfies the following conditions: (i) $\sim_L$ is of finite index, (ii) all shortest paths from an internal node to a leaf in trees from L are of a bounded length. Consequently, although **AB**-grammars are equivalent to $\epsilon-$free CFGs, the T-equivalence does not hold; the tree languages of **AB**-grammars are a proper subclass of the tree languages of CFGs.

The following theorem was proved in (Buszkowski, 1986b) for $(\backslash, /)-$types and (Kandulski, 1988a,b) for types with product.

Theorem 4.2. **NL**-*grammars are T-equivalent to* **AB**-*grammars, whence they are equivalent to* $\epsilon-$*free CFGs.*

The proof employs a transformation of **NL**-derivations to a normal form: for any provable sequent $A \Rightarrow B$, there exists a sequence $(A_0, \ldots, A_n)$ such that $A_0 = A$, $A_n = B$ and $A_{i-1} \Rightarrow A_i$, $i = 1, \ldots, n$, are certain basic provable sequents; furthermore, after a number of sequents, reducing the size of types, there come sequents which expand the size of types.. Accordingly, given an **NL**-grammar G, one obtains an equivalent **AB**-grammar by applying all possible reducing transformations to the type lexicon.

Later, another proof of the equivalence of **NL**-grammars and CFGs was presented in (Jäger, 2004). The latter proof employs some kind of interpolation of trees (on the set of types) by single types, which has been refined in (Buszkowski, 2005) to prove the polynomial time complexity and the context-freeness of **NL** with assumptions (and other related systems). In the next section we discuss this method in more detail.

Another kind of interpolation was studied in (Roorda, 1991) for **L** and other associative systems. Let $|\Gamma|_p$ denote the number of occurrences of variable p in Γ (as a subtype of a type in Γ). Roorda (1991) proves the following interpolation lemma: if $\vdash_{\mathbf{L}} \Gamma, \Delta, \Gamma' \Rightarrow A$, then there exists type D such that $\vdash_{\mathbf{L}} \Delta \Rightarrow D$, $\vdash_{\mathbf{L}} \Gamma, D, \Gamma' \Rightarrow A$, and for any variable p, $|D|_p$ is not greater than the minimum of $|\Delta|_p$ and $|\Gamma\Gamma' A|_p$. In this context, D is referred to as *an interpolant* of Δ in $\Gamma, \Delta, \Gamma' \Rightarrow A$. Roorda's lemma yields more than standard interpolation lemmas for classical logic which only claim that any variable occurring in D occurs in both Δ and $\Gamma\Gamma' A$.

This interpolation lemma plays a crucial role in the proof of the following result, due to Pentus (1993).

Theorem 4.3. **L**-*grammars are equivalent to* ϵ*–free CFGs.*

This theorem confirms a conjecture, posed as early as (Bar-Hillel *et al.*, 1960) and attacked without success by several authors.

Pentus's proof deeply penetrates the structure of **L**-derivations. Since **L** is a multiplicative fragment of **PLL**, then two different branches of a proof tree are completely independent. In particular, in any proof tree one can rename the (Id)-leaves in such a way that each variable has zero or two occurrences in the tree. Consequently, every provable sequent is a substitution instance of a provable sequent in which each variable has precisely two occurrences (if any); a provable sequent satisfying this condition is said to be *thin*, if every type in this sequent contains at most one occurrence of any vari-

able (such types are said to be *thin*). Roorda's lemma applied to a thin sequent yields a thin interpolant. This lemma also implies that any provable sequent $A_1, \ldots, A_n \Rightarrow A_{n+1}$ such that each variable has zero or two occurrences in the sequent can be transformed into a thin sequent $B_1, \ldots, B_n \Rightarrow B_{n+1}$ such that B_i is an interpolant of A_i, for $i = 1, \ldots, n$, and B_{n+1} is an interpolant of $A_1 \ldots A_n$. Let m be the maximal number of occurrences of variables in a type B_i. Using a combinatorial argument, one shows that, for $n \geq 2$, there exists $1 \leq i \leq n-1$ such that $B_i B_{i+1}$ has an interpolant D whose size is not greater than m. Accordingly, if $B_1, \ldots, B_n \Rightarrow B_{n+1}$ is a thin sequent with $n \geq 2$, then one can reduce the antecedent to B_{n+1} using **L**-provable sequents of the form $A, B \Rightarrow C$ such that the size of each type A, B, C is bounded by m. Applying substitution, the same can be shown for any provable sequent $A_1, \ldots, A_n \Rightarrow A_{n+1}$. Reductions of this form can easily be simulated by a CFG, which yields one direction of Theorem 4.3. The other direction is an easy consequence of Theorem 4.1; by cut elimination, if $\vdash_{\mathbf{L}} A_1, \ldots, A_n \Rightarrow s$ and all types A_i are as in Theorem 4.1, then any cut-free proof of this sequent is actually a proof in **AB**. Consequently, any **L**-grammar of the form restricted as in Theorem 4.1 is trivially equivalent to an **AB**-grammar (the latter has the same type lexicon as the former). An analogous theorem holds for **L***-grammars.

(Bulińska, 2005) shows that **L**-grammars with finitely many assumptions of the form $p \Rightarrow q$ are equivalent to CFGs. It is not known whether more general assumptions of the form $p_1, \ldots, p_n \Rightarrow p$ preserve context-freeness. As early as Buszkowski (1982), it was shown that the $(/)-$fragment of **L** with finitely many assumptions of the form $p \Rightarrow q/r$ and $p/q \Rightarrow r$ can generate arbitrary $\epsilon-$free recursively enumerable languages. This result is easier, if types with product are admitted; see (Buszkowski, 2005).

Kanazawa (1992) shows that **FL** can generate languages, which are not context-free, e.g. the intersection of two context-free languages; it suffices to work with **L** with $\wedge$. Interestingly, **NL** with $\wedge$ and **FNL** with distribution (also with boolean or Heyting operations) remain context-free, even supplied with finitely many assumptions. It can be understood better, if one remembers that regular tree languages are closed under intersection and join.

It is well known that the problem $x \in L(G)$ (*the membership problem*) for CFGs in Chomsky Normal Form can be solved in time $O(kn^3)$, where $n = |x|$ and k is the number of production rules of G;

a standard algorithm CYK computes all nonterminals A such that $A \rightarrow_G y$ for any nonempty interval y of x. Since **AB**-grammars can be regarded as special CFGs in Chomsky Normal Form, the same algorithm works for **AB**-grammars; now k equals the cardinality of $T^s(G)$. This algorithm, however, cannot be applied to **NL**-grammars, **L**-grammars, and their variants. The proof of Theorem 4.2 yields, actually, an exponential transformation of the given **NL**-grammar into an equivalent **AB**-grammar. A polynomial transformation can also be provided, whence the membership problem for **NL**-grammars is polynomial (see Section 5). It is not the case for **L**-grammars, if P ≠ NP; Pentus (2006) shows the NP-completeness of **L**.

Pregroup grammars are equivalent to CFGs , and the transformation is polynomial (Buszkowski, 2001; Buszkowski and Moroz, 2008). By Theorem 3.2, $\Gamma \Rightarrow t$ can be proved, using non-expanding steps only, which can be simulated by a CFG. $\Gamma \Rightarrow \Delta$ holds in **CBL** if and only if $\Gamma, \Delta^r \Rightarrow \epsilon$ holds in **CBL**, which yields the polynomial time decidability of **CBL**. The membership problem for pregroup grammars is polynomial (Oehrle, 2004; Moroz, 2010). Preller (2007) provides linear parsing algorithms for some restricted pregroup grammars. Pregroup grammars enriched with partial commutation go beyond the context-free world (Francez and Kaminski, 2007).

5. Substructural logics

In this section we briefly discuss some recent results on substructural logics which are based on methods elaborated for Lambek calculi and categorial grammars.

We have already mentioned the fact that FMP for **MALL** and Cyclic Noncommutative **MALL** directly follow from FMP for the product-free $\mathbf{L}^*_e$ and **L*** with 0 and one new rule.

A stronger version of FMP is Finite Embeddability Property (FEP). A class of algebras $\mathcal{C}$ has FEP, if every finite partial subalgebra of an algebra from $\mathcal{C}$ can be embedded in a finite algebra from $\mathcal{C}$. FEP of $\mathcal{C}$ is equivalent to FMP of the universal theory of $\mathcal{C}$. If $\mathcal{C}$ is closed under finite products, then FEP of $\mathcal{C}$ is equivalent to Strong Finite Model Property of $\mathcal{C}$, i.e. FMP of the Horn theory of $\mathcal{C}$.

The above notions refer to first-order logic. One consider the first-order language corresponding to the given class of algebras (it is determined by their similarity type). For algebras, considered in this paper, the only designated relation symbols are $\leq$ and $=$. Function symbols correspond to logical operations, e.g. product, left residual, right residual, join, meet. In this setting, types (formulas of type logics) correspond to terms, and sequents correspond to atomic formulas $s \leq t$ (s, t are terms). Clearly $s = t$ can be represented as the conjunction of $s \leq t$ and $t \leq s$. If $\vee$ or $\wedge$ are accessible, then $s \leq t$ can be defined as $s \vee t = t$, and similarly with $\wedge$.

The equational theory of $\mathcal{C}$ is the set of equations valid in $\mathcal{C}$; without lattice operations, one considers the atomic theory of $\mathcal{C}$, i.e. the set of inequations $s \leq t$, valid in $\mathcal{C}$. One says that $\mathcal{C}$ has FMP, if the equational (atomic) theory of $\mathcal{C}$ has FMP, it means: every equation (atomic formula) not valid in $\mathcal{C}$ is falsified in a finite algebra from $\mathcal{C}$. A Horn formula is any formula $\psi_1 \& \cdots \& \psi_n \Rightarrow \chi$ such that ψ_i and χ are equations (atomic formulas). The Horn theory of $\mathcal{C}$ is the set of all Horn formulas valid in $\mathcal{C}$. SFMP of $\mathcal{C}$ is FMP of the Horn theory of $\mathcal{C}$. A universal sentence is of the form $\forall x_1 \ldots \forall x_n \varphi$, where φ is a quantifier-free formula. The universal theory of $\mathcal{C}$ is the set of all universal sentences valid in $\mathcal{C}$.

Type logics, discussed in this paper, naturally correspond to some classes of algebras; it was already discussed in Section 2. The completeness of **L** with respect to the class of residuated semigroups means, actually, that **L** is an axiomatization of the atomic theory of this class. The strong completeness of **L** with respect to this class means that **L** (with (CUT)) provides an axiomatization of the Horn theory of this class. We know that the consequence relation $\Phi \vdash_{\mathbf{L}} \Gamma \Rightarrow A$ (for finite sets Φ) is undecidable; this yields the undecidability of the Horn theory of residuated semigroups (even their reducts with / only (Buszkowski, 1982)). Similarly, the Horn theories of residuated monoids, residuated lattices, bilinear algebras and lattice-ordered bilinear algebras are undecidable; also see (Galatos *et al.*, 2007). Since FMP of a finitely axiomatizable class entails the decidability of the corresponding theory, then SFMP (FEP) cannot hold for any of this classes.

The decidability of the Horn theory of commutative bilinear algebras, i.e. algebras corresponding to **MLL**, remains open; it is equivalent to the decidability of the consequence relation of **BCI** (use the faithful interpretation of **MLL** in **BCI**, discussed in Sec-

tion 3). Interestingly, this problem is closely related to the still open problem of the decidability of **MELL**, i.e. **PLL** without additives, and to the problem $L(G) = \emptyset$ (the emptiness problem) for categorial grammars based on the Lambek-van Benthem calculus. It is known that these grammars generate at least all permutation closures of context-free languages, but their precise power is not known. If the emptiness problem for them is undecidable, then the consequence relation of **BCI** is undecidable, whence **MELL** is undecidable. We use a deduction theorem for **BCI**: $\{\psi_1, \ldots, \psi_n\} \vdash_{\mathbf{BCI}} \chi$ iff $\vdash_{\mathbf{BCI}} \Gamma \Rightarrow \chi$, for some multiset Γ built from formulas $\psi_1, \ldots, \psi_n$. We also employ a many-valued interpretation of **BCI** in the Lambek-van Benthem calculus, provided in (Buszkowski, 1996).

In opposition to the associative case, the consequence relations of nonassociative Lambek calculi **NL**, **NL***, **NL1** are decidable; their complexity is polynomial. The corresponding classes of algebras possess FEP. Also the class of distributive lattice-ordered residuated groupoids posesses FEP, which yields the decidability of the consequence relation of **FNL** with distribution (of $\wedge$ over $\vee$, and conversely). Categorial grammars based on these systems generate context-free languages.

These results, solving some open problems in substructural logics, have been proved by a refinement of the interpolation method of Jäger (2004). Let $X \Rightarrow A$ be a sequent of **NL**. Let Φ be a finite set of simple sequents (assumptions). Let T be a finite set of types, containing all types appearing in $X \Rightarrow A$ and Φ and being closed under subtypes. An interpolation lemma proved in (Buszkowski, 2005) states: if $\Phi \vdash_{\mathbf{NL}} X[Y] \Rightarrow A$, then there exists a type $D \in T$ such that $\Phi \vdash_{\mathbf{NL}} X[D] \Rightarrow A$ and $\Phi \vdash_{\mathbf{NL}} Y \Rightarrow D$ (here Y is a subtree of X). Jäger proves a slightly weaker form of this lemma for the pure **NL** only.

This lemma directly implies that **NL**-grammars with assumptions generate context-free languages: any provable sequent $X \Rightarrow A$ can be proved by (N-CUT) from provable basic sequents $B \Rightarrow C$ and $(B, C) \Rightarrow D$ such that $B, C, D \in T$. This proof, however, is non-constructive, since we do not know at this moment whether the consequence relation is decidable (it is constructive for the pure **NL**). Fortunately, one can construct all basic sequents (with types from T), provable from Φ, in polynomial time (Buszkowski, 2005), which yields the polynomial complexity and a constructive proof of the context-freeness of **NL** with assumptions. The context-freeness

can also be proved for nonassociative systems with $\wedge$ and $\wedge, \vee$ (assuming distribution); see (Buszkowski and Farulewski, 2009).

To prove SFMP of the class of residuated groupoids, one uses nuclei, i.e. some closure operators on powerset algebras; models are families of all closed sets. The basic closed sets are of the form $\{Y : \Phi \vdash_{\mathbf{NL}} X[Y] \Rightarrow A\}$ such that A and all types in $X[Y]$ belong to T. The closed sets are intersections of families of basic closed sets. Using the interpolation lemma, stated above, one shows that the family of basic closed sets is finite, whence the family of all closed sets is finite. It yields a finite counter-model for any sequent not provable from Φ. Similar arguments show FEP for other classes of nonassociative algebras, mentioned above, and for the class of boolean-ordered residuated groupoids and Heyting-ordered residuated groupoids (Buszkowski and Farulewski, 2009). These results can be extended for multi-modal residuated algebras; actually, they provide a new way of proving FMP for a wide class of modal logics.

Pratt (1991) introduced action algebras, defined as residuated monoids with $\vee, \bot$ and Kleene star *. For an element a, a^* is the least reflexive and transitive element b such that $a \leq b$ (b is reflexive, if $1 \leq b$, and transitive, if $bb \leq b$). The $(\cdot, \vee, ^*, \bot)$−reduct of an action algebra is a Kleene algebra. Interesting examples of action algebras are relation algebras and algebras of languages $P(\Sigma^*)$. It is not known whether the equational theory of action algebras is decidable. Action algebras with $\wedge$ are called action lattices.

Relation algebras and algebras of languages are *-continuous, it means: $a^* = \sup\{a^n : n \in \omega\}$, for any element a. The complete logic of *-continuous action algebras can be presented as the $\wedge$−free fragment of **FL**, supplied with * and the following rules:

$$(*\mathrm{L})\ \frac{\{\Gamma, A^n, \Gamma' \Rightarrow B : n \in \omega\}}{\Gamma, A^*, \Gamma' \Rightarrow B}\ ,\quad (*\mathrm{R})\ \frac{\Gamma_1 \Rightarrow A\,; \ldots ;\ \Gamma_n \Rightarrow A}{\Gamma_1, \ldots, \Gamma_n \Rightarrow A^*}\ .$$

(*L) is an infinitary rule; it has infinitely many premises $\Gamma, A^n, \Gamma' \Rightarrow B$ (here A^n stands for the sequence of n copies of A). (*R) is an infinite collection of finitary rules, one for each $n \in \omega$ (for $n = 0$, it is an axiom $\Rightarrow A^*$). We denote this system by **ACT***.

ACT* admits cut-elimination. The complexity of **ACT*** is Π^0_1. It follows from a theorem on the elimination of negative occurrences of *: $\Gamma \Rightarrow B$ is provable if, and only if, for all $n \in \omega$, $I_n(\Gamma \Rightarrow B)$ is provable, where $I_n(\Gamma \Rightarrow B)$ arises from $\Gamma \Rightarrow B$, after one has replaced each negative occurrence of A^* by A^n (here

A^n stands for the product of n copies of A). Since (*L) introduces negative occurrences of *, $I_n(\Gamma \Rightarrow B)$ is provable if and only if it is provable without (*L), whence it is provable in a decidable finitary logic; this yields the upper bound of complexity (Buszkowski and Palka, 2008). In (Buszkowski, 2007a), the problem has been shown Π^0_1–hard. Interestingly, the proof essentially uses Theorem 4.1; by this theorem, the problem $L(G) = \Sigma^+$, for **AB**-grammars G in the restricted form, described in the theorem, is Π^0_1–complete (since an analogous problem for CFGs is so). Using some basic properties of **FL**, one shows that the latter problem reduces to the provability problem for **ACT***. Consequently, this problem is Π^0_1–complete. Similar results have been obtained for **ACT*** with $\wedge$, which is the complete logic of *-continuous action lattices and for different theories of some classes of relation algebras.

It follows from these results that the equational theory of *-continuous action algebras (lattices) is essentially stronger than that of all action algebras (lattices); for Kleene algebras, both theories are equal. Also, one cannot find a decidable dynamic logic, complete with respect to relational frames, which constructs programs by means of residual operations $\backslash, /$ and regular operations $\vee, \cdot, ^*$ (such logics have been proposed by some authors to express the weakest precondition and the strongest postcondition of a program).

Lambek calculi with Kleene star can be applied in linguistics in order to supply categorial grammars with regular expressions; see (Buszkowski and Palka, 2008) for examples. For effectiveness, some restrictions are necessary; for instance, one may exclude sequents with negative occurrences of Kleene star.

Works Cited

1. Abrusci, Vito Michele. 1991. Phase semantics and sequent calculus for pure noncommutative classical linear logic. *Journal of Symbolic Logic* 56: 1403–1451.
2. Ajdukiewicz, Kazimierz. 1935. Die syntaktische Konnexität. *Studia Philosophica* 1: 1–27.
3. Bar-Hillel, Yehoshua, Chaim Gaifman, and Eliahu Shamir. 1960. On categorial and phrase structure grammars. *Bulletin Res. Council Israel* F(9): 155–166.

4. van Benthem, Johan. 1986. *Essays in Logical Semantics*. Dordrecht: D. Reidel.
5. —. 1988a. The Lambek Calculus. In Oehrle *et al.* (1988), 35–68.
6. —. 1988b. The semantics of variety in categorial grammar. In Buszkowski *et al.* (1988), 37–55.
7. —. 1991. *Language in Action. Categories, Lambdas and Dynamic Logic*, *Studies in Logic and The Foundations of Mathematics*, vol. 130. North Holland.
8. —. 2005. The Categorial Fine-Structure of Natural Language. In Casadio *et al.* (2005), 3–29.
9. Bulińska, Maria. 2005. The Pentus Theorem for Lambek Calculus with Simple Nonlogical Axioms. *Studia Logica* 81(1): 43–59.
10. Buszkowski, W., W. Marciszewski, and J. van Benthem, eds. 1988. *Categorial Grammar*. Amsterdam: John Benjamins.
11. Buszkowski, Wojciech. 1982. Some decision problems for the theory of syntactic categories. *Zeitschrift f. mathematische Logik und Grundlagen der Mathematik* 28: 539–548.
12. —. 1986a. Completeness results for Lambek Syntactic Calculus. *Zeitschrift f. mathematische Logik und Grundlagen der Mathematik* 32: 13–28.
13. —. 1986b. Generative capacity of nonassociative Lambek calculus. *Bulletin Polish Acad. Scie. Math.* 34: 507–516.
14. —. 1986c. Typed functorial languages. *Bulletin Polish Acad. Scie. Math.* 34: 495–505.
15. —. 1996. Extending Lambek Grammars to Basic Categorial Grammars. *Journal of Logic, Language and Information* 5(3/4): 279–295.
16. —. 1997. Mathematical Linguistics and Proof Theory. In *Handbook of Logic and Language*, edited by J. van Benthem and A. ter Meulen, 683–736. Amsterdam: Elsevier.
17. —. 2001. Lambek Grammars Based on Pregroups. In *Logical Aspects of Computational Linguistics*, edited by P. de Groote, G. Morrill, and C. Retoré, *Lecture Notes in Artificial Intelligence*, vol. 2099, 95–109. Berlin: Springer.
18. —. 2002. Finite Models of Some Substructural Logics. *Mathematical Logic Quarterly* 48(1): 63–72.
19. —. 2003a. Sequent systems for compact bilinear logic. *Mathematical Logic Quarterly* 49(5): 467–474.

20. —. 2003b. Type Logics in Grammar. In *Trends in Logic. 50 years of Studia Logica*, edited by V. H. Hendriks and J. Malinowski, Trends in Logic, 337–382. Dordrecht: Kluwer.
21. —. 2005. Lambek Calculus with Nonlogical Axioms. In Casadio *et al.* (2005), 77–93.
22. —. 2007a. On Action Logic: Equational Theories of Action Algebras. *Journal of Logic and Computation* 17(1): 199–217.
23. —. 2007b. Type Logics and Pregroups. *Studia Logica* 87(2/3): 145–169.
24. —. 2008. On the Complexity of Some Substructural Logics. *Reports on Mathematical Logic* 43: 5–24.
25. Buszkowski, Wojciech and Maciej Farulewski. 2009. Nonassociative Lambek Calculus with Additives and Context Free Languages. In *Languages: From Formal to Natural. Esays Dedicated to Nissim Francez*, edited by O. Grumberg, M. Kaminski, S. Katz, and S. Wintner, *Lecture Notes in Computer Science*, vol. 5533, 45–58. Berlin: Springer.
26. Buszkowski, Wojciech and Katarzyna Moroz. 2008. Pregroup grammars and context-free grammars. In *Computational Algebraic Approaches to Natural Language*, edited by C. Casadio and J. Lambek, 1–21. Monza: Polimetrica.
27. Buszkowski, Wojciech and Ewa Palka. 2008. Infinitary Action Logic: Complexity, Models and Grammars. *Studia Logica* 89(1): 1–18.
28. Casadio, C., P. J. Scott, and R.A. G. Seely, eds. 2005. *Language and Grammar. Studies in Mathematical Linguistics and Natural Language*, *CSLI Lecture Notes*, vol. 168. Stanford: CSLI Publications.
29. Francez, Nissim and Michael Kaminski. 2007. Commutation-Augmented Pregroup Grammars and Mildly Context-Sensitive Languages. *Studia Logica* 87(2/3): 295–321.
30. Gabbay, Dov. 1996. *Labelled Deductive Systems*. Oxford: Oxford University Press.
31. Galatos, Nikolaos, Tomasz Kowalski, Peter Jipsen, and Hiroakira Ono. 2007. *Residuated Lattices: an Algebraic Glimpse at Substructural Logics*. Studies in Logic and The Foundations of Mathematics. Amsterdam: Elsevier.
32. Girard, Jean Yves. 1987. Linear logic. *Theoretical Computer Science* 50: 1–102.
33. de Groote, Philippe. 2001. Towards abstract categorial gram-

mars. In *Proceedings of 39th Annual Meeting of the Association of Computational Linguistics*, 252–259. Association of Computational Linguistics.

34. Jäger, Gerhard. 2004. Residuation, structural rules and context-freeness. *Journal of Logic, Language and Information* 13(1): 47–59.
35. Kanazawa, Makoto. 1992. The Lambek Calculus Enriched with Additional Connectives. *Journal of Logic, Language and Information* 1(2): 141–171.
36. Kandulski, Maciej. 1988a. The equivalence of nonassociative Lambek categorial grammars and context-free grammars. *Zeitschrift f. mathematische Logik und Grundlagen der Mathematik* 34(1): 41–52.
37. —. 1988b. Phrase structure languages generated by categorial grammars with product. *Zeitschrift f. mathematische Logik und Grundlagen der Mathematik* 34(4): 373–383.
38. Kiślak-Malinowska, Aleksandra. 2007. On the Logic of beta-pregroups. *Studia Logica* 87(2/3): 323–342.
39. Lambek, Joachim. 1958. The mathematics of sentence structure. *The American Mathematical Monthly* 65: 154–170.
40. —. 1961. On the calculus of syntactic types. In *Structure of Language and Its Mathematical Aspects*, edited by R. Jakobson, 166–178. Providence: American Mathematical Society.
41. —. 1995. Bilinear logic in algebra and linguistics. In *Advances in Linear Logic*, edited by J.-Y. Girard, Y. Lafont, and L. Regnier, 43–60. Cambridge: Cambridge University Press.
42. —. 1999. Type grammars revisited. In *Logical Aspects of Computational Linguistics*, edited by A. Lecomte, F. Lamarche, and G. Perrier, *Lecture Notes in Artificial Intelligence*, vol. 1582, 1–27. Berlin: Springer.
43. —. 2008. *From Word to Sentence: a computational algebraic approach to grammar*. Monza: Polimetrica.
44. Moortgat, Michael. 1988. *Categorial Investigations. Logical and Linguistic Aspects of the Lambek Calculus*. Dordrecht: Foris.
45. —. 1996. Multimodal linguistic inference. *Journal of Logic, Language and Information* 5(3/4): 349–385.
46. —. 1997. Categorial Type Logics. In *Handbook of Logic and Language*, edited by J. van Benthem and A. ter Meulen, 93–177. Amsterdam: Elsevier.

47. —. 2009. Symmetric Categorial Grammar. *Journal of Philosophical Logic* 38(6): 681–710.
48. Moroz, Katarzyna. 2010. A Savateev-style parsing algorithm for pregroup grammars. In *Proceedings of Formal Grammar 14*, *Lecture Notes in Computer Science*, vol. 5591.
49. Morrill, Glyn. 1994. *Type Logical Grammar. Categorial Logic of Signs*. Dordrecht: Kluwer.
50. Oehrle, R. T., E. Bach, and D. Wheeler, eds. 1988. *Categorial Grammars and Natural Language Structures*. Dordrecht: D. Reidel.
51. Oehrle, Richard T. 2004. A parsing algorithm for pregroup grammars. In *Categorial Grammars: an Efficient Tool for Natural Language Processing*, 59–75. The University of Monpellier.
52. Ono, Hiroakira and Yoko Komori. 1985. Logics without the contraction rule. *Journal of Symbolic Logic* 50: 169–201.
53. Pentus, Mati. 1993. Lambek grammars are context-free. In *Proceedings of the 8th IEEE Symposium on Logic in Computer Science*, 429–433.
54. —. 1995. Models for the Lambek calculus. *Annals of Pure and Applied Logic* 75: 179–213.
55. —. 2006. Lambek calculus is NP-complete. *Theoretical Computer Science* 357: 186–201.
56. Pratt, Vaughan. 1991. Action logic and pure induction. In *Logics in AI*, edited by J. van Eijck, *Lecture Notes in Artificial Intelligence*, vol. 478, 97–120. Berlin: Springer.
57. Preller, Anne. 2007. Linear Processing with Pregroups. *Studia Logica* 87(2/3): 171–197.
58. Preller, Anne and Joachim Lambek. 2007. Free compact 2-categories. *Mathematical Structures for Computer Sciences* 17(1): 1–32.
59. Roorda, Dirk. 1991. *Resource Logics: Proof-theoretical Investigations*. Ph.D. thesis, University of Amsterdam.
60. Schroeder-Heister, P. and K. Dosen, eds. 1993. *Substructural Logics*. Oxford: Clarendon Press.