ELSEVIER
Theoretical Computer Science 199 (1998) 5–24
Theoretical Computer Science

Algebraic structures in categorial grammar

Wojciech Buszkowski*
Faculty of Mathematics and Computer Science, Adam Mickiewicz University, 60-769 Poznań, Poland

Abstract

In this paper we discuss different algebraic structures which are natural algebraic frames for categorial grammars. First, absolutely free algebras of functor-argument structures and phrase structures together with power-set algebras of types are used to characterize structure languages of Basic categorial grammars and to provide algorithms for equivalence problems and related questions. Second, unification applied to the above frames is employed to develop learning procedures for basic categorial grammars. Third, residuated algebras are used to model language hierarchies of Lambek categorial grammars. The paper focuses on the author's research in this area with references to related works in logic and linguistics. © 1998—Elsevier Science B.V. All rights reserved

Keywords: Categorial grammar; Type logics; Type algebras; Lambek calculus; Residuated semigroup

0. Introduction

Categorial grammars are formal grammars based upon some fundamental ideas of mathematical logic: the theory of types and deductive systems. Expressions of a language are divided in syntactic categories denoted by logical types; usually, the hierarchy of syntactic categories is closely related to a hierarchy of semantic categories determined by logical types of semantic denotations of these expressions. Instead of production rules, characteristic of generative grammars, categorial grammars employ certain deductive systems of type change. Basic categorial grammars use a system whose only inference rule is Modus Ponens (no axioms), while different kinds of categorial grammar of current interest are based on much richer systems with special introduction and elimination rules for logical constants. From the modern perspective, the latter systems belong to so-called substructural logics whose most famous representatives are linear logics of Girard [22]. Due to the interplay of logical and linguistic issues, the theory of categorial grammars enjoys nowadays a growing significance in (semantically

* E-mail: buszko@math.amu.edu.pl.

0304-3975/98/$19.00
PII S0304-3975(97)00266-1

oriented) computational linguistics, logical foundations of computer science and logical philosophy of language (see [7]).

A standard mathematical framework in the theory of deductive systems is algebraic models; see e.g. [37]. Deductive systems of categorial grammars can be modelled by algebraic structures of different kind. Pure MP deductions induce functor-argument structures of expressions, and these structures form an absolutely free algebra (term algebra). Substructural logics used by modern categorial grammars, e.g. the Lambek calculus, refer to residuated algebras, in particular residuated (noncommutative or commutative) groupoids and semigroups. Many problems of both logical and linguistic interest can be solved by studying fine properties of these algebraic structures.

In this paper, we outline some basic notions and methods connected with algebraic structures of categorial grammars: absolutely free algebras of syntactic structures and residuated algebras. The former are especially significant for the theory of basic categorial grammars (however, they also play a role for richer systems; every categorial grammar can be regarded as a basic categorial grammar with an infinite set of lexical assumptions [15]). In Section 1 we define several notions relevant to functor-argument structures (tree languages, congruences, type algebras, quotient-algebras) and show their role in studying various mathematical properties of basic categorial grammars (characterization of tree languages generated by these grammars, relation to context-free grammars, decidability problems). In Section 2 we describe an algorithm for determining a minimal categorial grammar whose structure language contains a given finite set of functor-argument structures. This algorithm involves unification of finite families of finite sets of type-schemes. We discuss different versions of this algorithm and its connections with problems of learnability of the class of basic categorial grammars. In Section 3 we focus on residuated algebras corresponding to the Lambek calculus and related systems; we show different methods to prove representation theorems for these algebras and completeness theorems for deductive systems and point out some further applications.

1. Algebras of basic categorial grammars

Basic categorial grammars (BCGs) were introduced by Bar-Hillel [3, 4] with origins in Ajdukiewicz [1], who had followed some logical ideas of Frege, Russell and Leśniewski. Expressions are assigned types; first, *lexical assumptions* assign types to atomic expressions (symbols, words), and second, Hilbert style deductions based on *Modus Ponens* are used to derive types of complex expressions.

Types can be represented as purely conditional formulas, applying two conditionals $\rightarrow$ and $\leftarrow$. Atomic types are simply (sentential) variables (one may also think of them as constants). The type $A \rightarrow B$ (resp. $B \leftarrow A$) is assigned to *functors* from type A to type B which take their arguments on the left (resp. right). Thus, if S, PN and N are atomic types of Sentence, Proper Noun and Common Noun, respectively, then $PN \rightarrow S$ is the type VP, of Verb Phrase, $S \leftarrow VP$ is the type NP, of full Noun Phrase, and

$NP \leftarrow N$ is the type D, of Determiner. A BCG is defined by a finite set of lexical assumptions $v : A$ such that v is an atomic expression and A is a type.. By V_G we denote *the lexicon* of the grammar G, i.e. the set of atomic expressions appearing in lexical assumptions. Given a string $v_1 \dots v_n$, $v_i \in V_G$, the grammar G assigns type A to this string, if there are assumptions $v_i : A_i$, $i = 1, \dots, n$, such that there is a deduction tree with the root A and the yield $A_1 \dots A_n$. Actually, this definition can be preserved for all kinds of categorial grammar, only, inference rules admissible in deductions are different for different kinds. BCGs admit two MP-rules:

$$(\mathrm{MP}\rightarrow) \quad \frac{A;\ A \rightarrow B}{B},$$

$$(\mathrm{MP}\leftarrow) \quad \frac{B \leftarrow A;\ A}{B}.$$

For example, the grammar defined by lexical assumptions:

every, the: $D = NP \leftarrow N$,

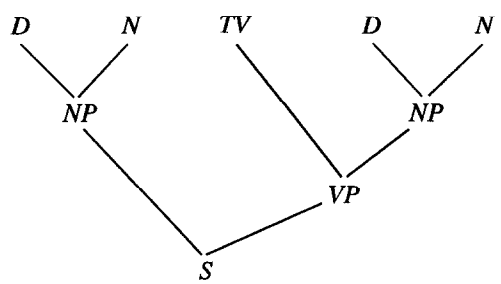
student, test: N,

passed: $TV = VP \leftarrow NP$,

where $NP = S \leftarrow VP$, assigns type S to the expression:

every student passed the test

on the basis of the following deduction:



This deduction can also be represented as the bracketed term:

$[[D,N]_1,[TV,[D,N]_1]_1]_1$

which yields the *functor-argument structure* of the given expression

[[every, student]$_1$,[passed, [the test]$_1$]$_1$]$_1$.

Here, the numerical subscript i means that the ith constituent takes the part of the functor. In the example above, always the first constituents are functors, but it is not the case for the structure:

[John, [likes, Mary]$_1$]$_2$

which is provided by the grammar:

John, Mary: PN,

likes: $(PN \rightarrow S) \leftarrow PN$.

The set $FS(V)$, of functor-argument structures over the lexicon V, is defined by the recursive clauses:

(a) $V \subseteq FS(V)$,

(b) if $X, Y \in FS(V)$ then $[X, Y]_i \in FS(V)$, for $i = 1, 2$.

X is the functor and Y is the argument in $[X, Y]_1$, and the opposite holds for $[X, Y]_2$. Clearly, $FS(V)$ can be treated as an absolutely free algebra with the set V of free generators and two binary operations $[\ldots,\ldots]_1$ and $[\ldots,\ldots]_2$. Dropping numerical subscripts changes functor-argument structures into *phrase structures*. $PS(V)$ denotes the set of phrase structures over V. Again, $PS(V)$ is an absolutely free algebra with one binary operation $[\ldots,\ldots]$.

We assume that each BCG G distinguishes an atomic type, say S, as *the basic type*, to be denoted S_G. We write $v_1 \ldots v_n \Rightarrow_G A$, if G assigns type A to string $v_1 \ldots v_n$. For $X \in FS(V_G)$, we write $X \Rightarrow_G A$, if G assigns type A to the yield of X by a deduction which determines the structure X. We define

$$L(G) = \{a \in V_G^+ : a \Rightarrow_G S_G\},$$

$$FL(G) = \{X \in FS(V_G) : X \Rightarrow_G S_G\}.$$

$L(G)$ and $FL(G)$ are called *the language* and *the F-language* of G, respectively. $PL(G)$ denotes *the P-language* of G: it arises from $FL(G)$, after one has dropped all numerical subscripts. Grammars G and G' are said to be *equivalent* (resp. *F-equivalent*, *P-equivalent*), if $L(G) = L(G')$ (resp. $FL(G) = FL(G')$, $PL(G) = PL(G')$).

Surely, the notions defined above are quite crucial for the theory of BCGs, as they (or their variants) are for other branches of mathematical linguistics. Below we outline main algebraic methods which are needed to establish fundamental properties of BCGs. Actually, one needs universal algebra and combinatorics of trees.

It is useful to see that types can be treated as functor-argument structures over the set P, of atomic types; just write $[A, B]_2$ for $A \rightarrow B$ and $[B, A]_1$ for $B \leftarrow A$. So, $Tp = FS(P)$ is the set of types. A *path* (resp. an *F-path*) in structure $X \in FS(V)$ is a sequence $X_0, \ldots, X_n$, of substructures of X, such that X_{i+1} is a constituent (resp. the functor) of X_i, for all $i < n$; n is *the length* of this path. *The height* of X ($h(X)$) is the maximal length of paths in X. *The F-degree* of X ($Fd(X)$) is the maximal length of F-paths in X. *The depth* of structure X ($d(X)$) is the minimal length of paths beginning with X and ending with an atom, and *the degree* of X ($\deg(X)$) is the maximal depth of substructures of X. Clearly, all F-free notions can be defined for phrase structures, as well. For $L \subseteq FS(V)$, we set

$$Fd(L) = \sup\{Fd(X) : X \in L\},$$

$$\deg(L) = \sup\{\deg(X) : X \in L\}.$$

The latter notion is also defined for $L \subseteq PS(V)$.

$X[v:Y]$ denotes the substitution of Y for atom v in X. Given a lexicon V, we set $V_x = V \cup \{x\}$, where x is a variable not in V. Each language $L \subseteq FS(V)$ determines *the basic congruence* $\sim_L$ in the algebra $FS(V)$, defined as follows: $X \sim_L Y$ iff, for all $Z \in FS(V_x)$,

$$Z[x:X] \in L \Leftrightarrow Z[x:Y] \in L.$$

The index of L (ind(L)) is the total number of equivalence classes of $\sim_L$. One easily shows that $\sim_L$ is the largest congruence in $FS(V)$ compatible with L, that means, for all $X, Y \in FS(V)$,

$$\text{if } X \sim_L Y \quad \text{then } X \in L \text{ iff } Y \in L.$$

The first problem which can be solved by these tools is an algebraic characterization of F-languages of BCGs. In [12], it has been proven that

(T1) For $L \subseteq FS(V)$, there is some BCG G such that $L = FL(G)$ if, and only if, both ind(L) and $Fd(L)$ are finite.

By $Tp(G)$ we denote the set of all types appearing in lexical assumptions of G. The grammar G determines the relation $\sim_G$ on $FS(V_G)$, defined as follows:

$$X \sim_G Y \text{ iff, for all } A,\ X \Rightarrow_G A \Leftrightarrow Y \Rightarrow_G A.$$

Clearly, $\sim_G$ is a congruence in $FS(V_G)$ compatible with $FL(G)$. We refer to this relation as *the basic congruence* determined by G. One easily shows the inequalities

$$\text{ind}(FL(G)) \leqslant \text{ind}(\sim_G),$$

$$Fd(FL(G)) \leqslant Fd(Tp(G)),$$

which yield the 'only if' part of (T1). For the 'if' part, one identifies atomic types with equivalence classes of $\sim_L$ and adds a new atomic type S. Languages L_A, for $A \in Tp$, are defined by induction on A:

(a) $L_p = p$, for atomic p; $L_S = L$,
(b) $L_{p \to B} = \{Y\colon (\exists X \in L_p)[X, Y]_2 \in L_B\}$,
(c) $L_{B \leftarrow p} = \{X\colon (\exists Y \in L_p)[X, Y]_1 \in L_B\}$.

Actually, we have defined L_A merely for types A of order less than 2, where *the order* of A (ord(A)) is a nonnegative integer defined as follows:

(a) $\text{ord}(p) = 0$, for atomic types p,
(b) $\text{ord}(A \to B) = \text{ord}(B \leftarrow A) = = \max(\text{ord}(B), \text{ord}(A) + 1)$.

One shows that $L_A \neq \emptyset$ only if $Fd(A) \leqslant Fd(L)$, and consequently, the BCG G defined by the lexical assumptions:

$$v : A \text{ iff } v \in L_A$$

is a finite object. This grammar G satisfies $FL(G) = L$. We also have $\text{ord}(G) \leqslant 1$, where ord($G$) is the maximal order of types in $Tp(G)$. Consequently,

(T2) Each BCG is F-equivalent to some BCG G' with $\text{ord}(G') \leqslant 1$.

To compare BCGs with context-free grammars (CFGs), it is useful to characterize P-languages of BCGs. Using (T1), one shows [13]:

(T3) For $L \subseteq PS(V)$, there is some BCG G such that $L = PL(G)$ if, and only if, both $\mathrm{ind}(L)$ and $\deg(L)$ are finite.

Accordingly, P-languages of BCGs is a narrower class than P-languages of CFGs: by the result of Thatcher [38], the latter are precisely those $L \subseteq PS(V)$ for which $\mathrm{ind}(L)$ is finite. For instance, the CFG defined by production rules $S \Rightarrow SS$, $S \Rightarrow a$, $S \Rightarrow b$ generates the total P-language $PS(a,b)$ whose degree is infinite; thus, it is not P-equivalent to any BCG. The BCG defined by lexical assumptions:

$$a, b : S,\ S \to S,$$

generates the same string language but not the same P-language (the degree of the latter equals 1).

In [4], there is proven the Gaifman theorem: BCGs are equivalent to CFGs, that means, both kinds of grammar yield the same class of string languages. This theorem is closely related to the Greibach normal form theorem for CFGs. Original proofs of both theorems use purely combinatorial transformations of one grammar into another one. An algebraic proof of the Gaifman theorem, based on the algebra $FS(V)$, is given in [16, 15]. First, observe that, for any BCG G, $\deg(PL(G)) \leqslant Fd(FL(G))$ and $\mathrm{ind}(PL(G))$ is finite. That yields the first part of the Gaifman theorem: each BCG is P-equivalent, hence also equivalent, to some CFG. For the second part, fix a CFG $\mathscr{G}$ in the Chomsky normal form. Then, $PL(\mathscr{G})$ is of finite index (by Thatcher's theorem). By the convention 'brackets associated to the left', each phrase structure can uniquely be represented in the form $vX_1 \ldots X_n$, where v is an atom and $n \geqslant 0$. The following transformation:

$$(vX_1 \ldots X_n)^T = [v, [X_1^T, [X_2^T, \ldots, [X_{n-1}^T, X_n^T] \ldots]]]$$

sends each phrase structure X to a phrase structure X^T such that $Fd(X^T) \leqslant 2$ and the yield of X^T equals that of X (here we identify phrase structures with functor-argument structures whose numerical subscripts always equal 1). We set

$$L = PL(\mathscr{G})^T = \{X^T :\ X \in PL(\mathscr{G})\}.$$

The language $L \subseteq FS(V_{\mathscr{G}})$ satisfies $Fd(L) \leqslant 2$. It requires some calculation to show that the finiteness of $\mathrm{ind}(PL(\mathscr{G}))$ entails the finiteness of $\mathrm{ind}(L)$. Then, by (T1), $L = FL(G)$, for some BCG G. Clearly, $L(G) = L(\mathscr{G})$, hence G is equivalent to $\mathscr{G}$.

The global equivalence problem for BCGs is the question if $L(G) = L(G')$, for arbitrary BCGs G and G'. Since the original proof of the Gaifman theorem yields an effective construction of a BCG G equivalent to any given CFG $\mathscr{G}$, then the global equivalence problem for CFGs (which is undecidable [23]) is effectively reducible to that for BCGs. Consequently, the global equivalence problem for BCGs is undecidable, and so is the global inclusion problem $L(G) \subseteq L(G')$. As shown in [14], a refinement of

algebraic notions sketched above provides algorithms for solving global F-equivalence and F-inclusion problems for BCGs and many related problems.

The quotient algebra $FS(V_G)/\sim_G$ is denoted ALG(G) and called *the basic algebra* of the BCG G. The quotient operations f_1, f_2 in the algebra ALG(G) are defined in the standard way:

$$f_i(X/\sim_G, Y/\sim_G) = [X, Y]_i/\sim_G,$$

for $i = 1, 2$. Here $X/\sim_G$ denotes the equivalence class of X with respect to $\sim_G$. Since $\sim_G$ is of finite index, then ALG(G) is finite. For any BCG G, one can effectively construct a powerset algebra over the set of types which is isomorphic to ALG(G).

By $T(G)$ we denote the set of all subtypes of types from $Tp(G)$. In the powerset $P(T(G))$, we define operations

$$F_1(T_1, T_2) = \{B\colon\ (\exists A \in T_2)(B \leftarrow A) \in T_1\},$$

$$F_2(T_1, T_2) = \{B\colon\ (\exists A \in T_1)(A \rightarrow B) \in T_2\}$$

for $T_1, T_2 \subseteq T(G)$. Thus, F_1 (resp. F_2) yields the results of all possible applications of rule (MP$\leftarrow$) (resp. (MP$\rightarrow$)) whose left premise is in T_1 and right premise is in T_2. The mapping

$$h_G(X/\sim_G) = \{A \in T(G)\colon\ X \Rightarrow_G A\}$$

is well defined, and h_G is a monomorphism of the algebra ALG(G) into the algebra $(P(T(G)), F_1, F_2)$. The image h_G(ALG(G)) is a subalgebra of the latter algebra; this subalgebra will be called *the type algebra* of G and denoted TP(G). Clearly, h_G is an isomorphism of ALG(G) onto TP(G).

For any BCG G, the algebra TP(G) can effectively be constructed. For TP(G) is the subalgebra of the (finite and effectively given) algebra $P(T(G))$ generated by the set of all sets

$$T_G(v) = \{A \in T(G)\colon\ v \Rightarrow_G A\},$$

for $v \in V_G$. Observe that $v \Rightarrow_G A$ holds if, and only if, $v : A$ is a lexical assumption of G. Now, many properties of BCGs can be expressed as properties of their type algebras, and the latter can effectively be verified.

The inclusion $\sim_G \subseteq \sim_{FL(G)}$ holds for every BCG G. We say that G is *well-formed* if $\sim_G = \sim_{FL(G)}$. For well-formed grammars G, syntactic categories defined as inter-substitutability classes with respect to the F-language of G are the same as natural equivalence classes determined by the type assignment of G.

(T4) The problem of whether G is well formed is decidable.

To prove (T4) we define

$$fl(G) = \{T \in \mathrm{TP}(G)\colon\ S_G \in T\}.$$

One easily shows

$$fl(G) = \{h_G(X/\sim_G)\colon X \in FL(G)\}.$$

Thus, $fl(G)$ represents the F-language $FL(G)$ in the algebra $\mathrm{TP}(G)$. Now, $\sim_G = \sim_{FL(G)}$ holds true if, and only if, the identity is the only congruence in $\mathrm{TP}(G)$ compatible with $fl(G)$, and the latter condition admits an effective verification.

In a similar way we prove

(T5) The global F-equivalence problem for BCGs is decidable.

Fix two BCGs G_1 and G_2. We assume $V_{G_1} = V_{G_2} = V$ (otherwise we add new atoms to the lexicons but no new lexical assumptions). Denote $L_i = FL(G_i)$, for $i = 1, 2$. First, observe that $L_1 = L_2$ if, and only if, there is an isomorphism g from the quotient algebra $FS(V)/\sim_{L_1}$ to the quotient algebra $FS(V)/\sim_{L_2}$ such that

$$g(v/\sim_{L_1}) = v/\sim_{L_2}, \quad \text{for all } v \in V,$$

$$\{g(X/\sim_{L_1})\colon X \in L_1\} = \{X/\sim_{L_2}\colon X \in L_2\}.$$

For the 'only if' direction, take the identity mapping for g. For the 'if' direction, assume there is an isomorphism g fulfilling the above equalities. By the first equality, we infer

$$g(X/\sim_{L_1}) = X/\sim_{L_2},$$

for all $X \in FS(V)$ (g is a homomorphism). It follows that $\sim_{L_1} \subseteq \sim_{L_2}$, since g is a function, and the converse inclusion is also true, since g is a bijective mapping. Consequently, $\sim_{L_1} = \sim_{L_2}$, and g is the identity mapping. The second equality and the fact that L_i is a union of equivalence classes $X/\sim_{L_i}$, for $X \in L_i$ yield $L_1 = L_2$. Now, the isomorphism condition can be copied in type algebras. Let $\sim_i$ be the largest congruence in $\mathrm{TP}(G_i)$ compatible with $fl(G_i)$, for $i = 1, 2$. Then, $FL(G_1) = FL(G_2)$ if, and only if, there is an isomorphism g' from the quotient algebra $\mathrm{TP}(G_1)/\sim_1$ to the quotient algebra $\mathrm{TP}(G_2)/\sim_2$ such that

$$g'(T_{G_1}(v)/\sim_1) = T_{G_2}(v)/\sim_2, \quad \text{for all } v \in V,$$

$$\{g'(T/\sim_1)\colon T \in fl(G_1)\} = \{T/\sim_2\colon T \in fl(G_2)\}.$$

Clearly, the latter isomorphism condition can effectively be verified.

Analogous methods can be used to find algorithms for many other problems concerning BCGs, for instance, the global P-equivalence problem, the F-inclusion and P-inclusion problems (use an effective construction of a BCG G such that $FL(G) = FL(G_1) \cup FL(G_2)$, and similarly for phrase languages), a construction of a rigid or well-formed grammar F-equivalent to a given grammar, a construction of a (restricted) complementation grammar for a given grammar, and so on. The reader is referred to [14] for details.

In Section 3 we consider categorial grammars based on stronger systems of types, as e.g. the Lambek calculus. For categorial grammars based on the Lambek calculus and related (associative!) systems, $FL(G)$ consists of all possible functor-argument structures which can be defined on strings from $L(G)$, and consequently, F-equivalence and P-equivalence are the same as (weak) equivalence [15].

2. Grammars determined by unification

The method of unification, extensively used in logic programming and unification systems in computational linguistics (see [36, 21]), can be applied to functor-argument structures and types in order to develop quite natural learning procedures for BCGs. The basic learning algorithms of that kind were described in [14, 19], and Kanazawa [25, 24] elaborated a Gold style learning theory for BCGs, essentially involving these algorithms (van Benthem [6] studies a closely related problem of solving 'categorial equations').

In this section, we briefly outline these algorithms and apply them to study the fine algebraic structure of F-languages, generated by BCGs.

Let us recall some basic notions concerning unification. We consider terms (of a first-order language) built from constants, variables and function symbols. *A substitution* is an assignment of terms to variables, and it naturally extends to a mapping from the set of terms to itself. A substitution σ is *a unifier* of a set T, of terms, if $\sigma(s)=\sigma(t)$, for all $s,t\in T$. σ is a unifier of a family $\{T_1,\ldots,T_n\}$, of sets of terms, if it is a unifier of each T_i, for $i=1,\ldots,n$. *A most general unifier* (mgu) of a family of sets of terms is a unifier σ of this family such that, for every unifier α of this family, there is a substitution β, such that $\alpha=\beta\sigma$. It is well known that, for any finite family of finite sets of types, one can effectively construct an mgu of this family (if exists) or prove the nonexistence of any unifier of it (see [30, 19]). Notice that two mgu's of the same family must be equal up to alphabetic variants.

We assume that atomic types are variables and constants. Thus, the set Tp, of all types, can be treated as a set of terms in the above sense. We describe an effective procedure which takes a finite set $L\subset FS(V)$ as an input, and returns a 'most general' BCG G such that $L\subseteq FL(G)$.

Fix a nonempty, finite set $L\subset FS(V)$. We assign type S to all structures from L, and to each occurrence of an argument substructure in a structure from L we assign a different variable. Then, types are assigned to occurrences of functor substructures of structures from L by the following rules:

$$(\mathrm{F}{\rightarrow})\quad \frac{[X,Y]_2:B;\ X:A}{Y:A\rightarrow B},$$

$$(\mathrm{F}{\leftarrow})\quad \frac{[X,Y]_1:B;\ Y:A}{X:B\leftarrow A}.$$

Now, we define the so-called *general form* of L ($GF(L)$) as the BCG determined by all assumptions $v:A$ such that $v\in V$ and A has been assigned to v by the above procedure.

We have

$$FL(GF(L)) = L.$$

For any BCG G and substitution σ, let $\sigma(G)$ denote the BCG determined by the assumptions $v : \sigma(A)$, for all assumptions $v : A$ of G. One easily shows

$$FL(G) \subseteq FL(\sigma(G)), \quad \text{for every BCG } G.$$

We also write $G_1 \subseteq G_2$, if each lexical assumption of G_1 is an assumption of G_2. The basic property of $GF(L)$ is: for every BCG G, $L \subseteq FL(G)$ if, and only if, there is a substitution σ such that $\sigma(GF(L)) \subseteq G$.

For each $v \in V$, let $T_L(v)$ be the set of all types A such that $v : A$ is a lexical assumption of $GF(L)$. We set

$$\mathcal{T}_L = \{T_L(v) : v \in V\}.$$

We assume that all sets $T_L(v)$ are nonempty (otherwise, drop the redundant atoms from V). Recall that a BCG G is *rigid*, if, for any $v \in V$, there is at most one type A such that $v : A$ is an assumption of G. We define $RG(L) = \sigma(GF(L))$, where σ is an mgu of $\mathcal{T}_L$. Thus, $RG(L)$ is a rigid BCG, and $L \subseteq FL(RG(L))$. The following theorems, proven in [19], show that $RG(L)$ is a most general rigid BCG G such that $L \subseteq FL(G)$.

(T6) For any nonempty, finite $L \subset FS(V)$, the following conditions are equivalent: (a) $L \subseteq FL(G)$, for some rigid BCG G, (b) the family $\mathcal{T}_L$ is unifiable.

(T7) For any rigid BCG G, $L \subseteq FL(G)$ if, and only if, there is a substitution α such that $\alpha(RG(L)) \subseteq G$.

To give a simple example, we consider the set L consisting of two structures:

- [Joan, smiles]$_2$,
- [Joan, [smiles, charmingly]$_2$]$_2$.

According to the procedure described above, we assign type S to these two structures, variables x, y to the first and the second occurrence of 'Joan', respectively, and variable z to the second occurrence of 'smiles'. By rule (F$\rightarrow$), we derive

- smiles: $x \rightarrow S$,
- [smiles, charmingly]$_2$: $y \rightarrow S$,
- charmingly: $z \rightarrow (y \rightarrow S)$.

So, $GF(L)$ is defined by the following assumptions:

- Joan: x, y, smiles: $z, x \rightarrow S$,
- charmingly: $z \rightarrow (y \rightarrow S)$.

The family $\mathcal{T}_L$ is unifiable; $\sigma(y) = x$, $\sigma(z) = x \rightarrow S$ is an mgu. Consequently, $RG(L)$ exists and is given by:

- Joan: x, smiles: $x \rightarrow S$,
- charmingly: $(x \rightarrow S) \rightarrow (x \rightarrow S)$.

It is natural to interpret $x = PN$, hence 'smiles' is of type VP, and 'charmingly' is of type $VP \to VP$, of Adverb. $RG(L)$ generates the infinite F-language which consists of structures:

- [Joan, smiles]$_2$,
- [Joan, [smiles, charmingly]$_2$]$_2$,
- [Joan, [[smiles, charmingly]$_2$, charmingly]$_2$]$_2$] etc.

It follows from (T7) that the latter F-language is contained in $FL(G)$, for every rigid BCG G such that $L \subseteq FL(G)$.

An F-language $L \subseteq FS(V)$ is said to be *rigid*, if $L = FL(G)$, for some rigid BCG G. Kanazawa [25, 24] proves that the class of rigid F-languages possesses finite elasticity: for all infinite sequences (A_n), $A_n \in FS(V)$, and all infinite sequences (L_n), of rigid F-languages $L_n \subseteq FS(V)$, there is an integer n such that either $A_n \in L_n$, or $\{A_0, \ldots, A_n\} \not\subseteq L_{n+1}$. Consequently, there exists a learning function for this class (this notion is not defined here; the reader is referred to [25, 24]). The same holds for the class of string languages, generated by rigid BCGs, and the class of string languages, generated by BCGs which assign at most k types to each lexical atom, for any $k \geqslant 1$. A computable learning function can be defined by an adaptation of the above algorithm. To prove the finite elasticity of the class of rigid F-languages, Kanazawa establishes the ascending chain condition (ACC) for this class: there is no infinite chain $L_0 \subset L_1 \subset L_2 \ldots$, of rigid F-languages over a finite lexicon.

These results enable us to say more about the class of rigid F-languages $L \subseteq FS(V)$. Let $\mathscr{R}$ denote the class consisting of the latter F-languages and the total F-language $FS(V)$ (which is not rigid, by (T1)). For any set $L \subseteq FS(V)$, we define

$$C(L) = \bigcap \{L' \in \mathscr{R} \colon L \subseteq L'\}.$$

We prove

(T8) For any set $L \subseteq FS(V)$, $C(L) \in \mathscr{R}$. Further, the operator C satisfies Tarski's conditions:
  (i) $L \subseteq C(L)$,
  (ii) if $L_1 \subseteq L_2$, then $C(L_1) \subseteq C(L_2)$,
  (iii) $C(C(L)) = C(L)$,
  (iv) for any $L \subseteq FS(V)$, there is a finite set $L' \subseteq L$ such that $C(L) = C(L')$.

We prove the first part of (T8). Fix a set $L \subseteq FS(V)$. If there is no $L' \in \mathscr{R}$ such that $L \subseteq L'$, then $C(L) = FS(V) \in \mathscr{R}$. Otherwise, fix $L' \in \mathscr{R}$ such that $L \subseteq L'$. If $L \neq \emptyset$ is finite, then $RG(L)$ exists (use (T6)), and $FL(RG(L))$ is the least rigid F-language containing L (use (T7)), hence $C(L) = FL(RG(L)) \in \mathscr{R}$. If $L = \emptyset$, then $C(L) = \emptyset \in \mathscr{R}$. So, assume L is infinite. Then, L is the join of an ascending chain $L_0 \subset L_1 \subset \ldots$, of finite F-languages. By (T6), $RG(L_n)$ exists, for all $n \geqslant 0$; we denote $L'_n = FL(RG(L_n))$. By (T7), we obtain

$$L'_0 \subseteq L'_1 \subseteq L'_2 \subseteq \ldots,$$

hence, by (ACC), there is an integer $k \geqslant 0$ such that $L'_n = L'_k$, for all $k \geqslant n$. Clearly $C(L) = L'_k \in \mathscr{R}$.

Conditions (i)–(iii) are obvious. To prove (iv), fix $L \subseteq FS(V)$. We may assume that L be infinite. If $L \subseteq L'$, for some rigid L', then we proceed as above; we have $C(L) = L'_k = C(L_k)$. So, assume there is no rigid F-language L' such that $L \subseteq L'$. Then, $C(L) = FS(V)$. We must show that $C(L^{\sim}) = FS(V)$, for some finite $L^{\sim} \subseteq L$. Suppose the contrary. Then, for every finite $L^{\sim} \subseteq L$, there is a rigid L'' such that $L^{\sim} \subseteq L''$. Consequently, $RG(L^{\sim})$ exists. As above, choose an ascending chain $L_0 \subset L_1 \subset \ldots$, of finite sets, whose join equals L. We define L'_n, for all $n \geqslant 0$, as above. By (ACC), there is $k \geqslant 0$ such that $L'_n = L'_k$, for all $n \geqslant k$. Then, $L \subseteq L'_k$, which contradicts the assumption.

The operator C is a natural grammatical consequence operator. The deductively closed sets $L = C(L)$ are precisely the rigid F-languages and the total F-language $FS(V)$. It would be interesting to find an axiomatization of this operator by means of 'inference rules' defined on functor-argument structures. One of these rules could be

$$\frac{A[B];\ A[C];\ D[B]}{D[C]};$$

here, $A[B]$ stands for a functor-argument structure A with a designated occurrence of a substructure B, and $A[C]$ results from substituting C for B in A, and similarly for $D[B]$ and $D[C]$. This rule can be justified as follows. Since $A[B]$ and $A[C]$ belong to $FL(G)$, where G is rigid, then B and C are assigned the same type; since also $D[B]$ belongs to $FL(G)$, then $D[C]$ must belong to $FL(G)$, because $\sim_G \subseteq \sim_{FL(G)}$.

Rigid BCGs and rigid F-languages are typical for artificial languages of formal logic and mathematics (but they are also useful technical tools for studying general properties of categorial grammars). Syntactic and semantic ambiguities of natural language are reflected by nonrigid grammars. A nonrigid version of the algorithm described above has been elaborated in [19].

The key notion is optimal unification. Let $\mathscr{T}$ be a family of sets of terms and σ be a substitution. $\mathrm{Ker}(\sigma)$ is the relation which holds between terms s and t iff $\sigma(s) = \sigma(t)$. Let $[t]$ denote the equivalence class of term t with respect to $\mathrm{Ker}(\sigma)$. We define

$$T/\sigma = \{[t] \cap T\colon\ t \in T\}, \quad \text{for } T \in \mathscr{T},$$

$$\mathscr{T}/\sigma = \bigcup \{T/\sigma\colon\ T \in \mathscr{T}\}.$$

σ is called an *optimal unifier* (ou) of the family $\mathscr{T}$, if it satisfies the conditions

(OU.1) σ is an mgu of $\mathscr{T}/\sigma$,

(OU.2) for all $T \in \mathscr{T}$, $s, t \in T$, if $\sigma(s) \neq \sigma(t)$, then the set $\{\sigma(s), \sigma(t)\}$ is not unifiable.

Intuitively, an ou for $\mathscr{T}$ is a most general substitution which unifies the family $\mathscr{T}$ as far as possible. For every nonempty family $\mathscr{T}$, of nonempty, finite sets of terms, one can effectively find finitely many ou's of $\mathscr{T}$ (they are all ou's of $\mathscr{T}$ up to alphabetic variants). We write $\sigma \leqslant_{\mathscr{T}} \sigma'$ if, for every $T \in \mathscr{T}$, the cardinality of T/σ is not greater than that of T/σ'. A *minimal unifier* (mu) for $\mathscr{T}$ is an ou for $\mathscr{T}$ which is $\leqslant$-minimal in the set of all ou's for $\mathscr{T}$. By the above, for every nonempty, finite family $\mathscr{T}$, of

nonempty, finite sets of terms, one can effectively find all mu's for $\mathscr{T}$ (up to alphabetic variants).

Let $\mathscr{G}(L)$ be the family of all BCGs $\sigma(GF(L))$ such that σ is an mu for $\mathscr{T}_L$. The following theorem, proven in [19], shows that $\mathscr{G}(L)$ contains precisely the 'most general' minimal BCGs G such that $L \subseteq FL(G)$. Here, 'minimality' is defined with respect to the following relation: $G \leqslant G'$ iff, for all $v \in V$, the cardinality of $T_G(v)$ is not greater than that of $T_{G'}(v)$.

(T9) Let $L \subset FS(V)$ be a finite set. The following conditions are equivalent:
  (a) G is minimal in the class of grammars G' such that $L \subseteq FL(G')$,
  (b) there are $G' \in \mathscr{G}(L)$ and a substitution α such that $G = \alpha(G')$.

Clearly, if $\mathscr{T}_L$ is unifiable, then $RG(L)$ is the only member of $\mathscr{G}(L)$, and minimal grammars G such that $L \subseteq FL(G)$ are rigid. Therefore, the nonrigid procedure is a 'conservative' generalization of the rigid one. Unlike the latter, the former always yields an outcome grammar.

Marciniec [31] studies more general versions of the above procedures in which input data can be of the form $X_i : A_i$, $i = 1, \ldots, m$ (positive data) as well as non-$Y_j : B_j$, $j = 1, \ldots, n$ (negative data). Roughly, negative data restrict the class of admissible substitutions, and the 'positive' procedures, described above, are relativized to this restricted class. Thus, the role of negative data is not merely to sieve out 'wrong' outcomes of the positive procedure, but they essentially influence the positive procedure.

3. Residuated algebras

In the last section we pass from absolutely free algebras of structure trees to algebras corresponding to stronger deductive systems applied in categorial grammars. The central notion is residuation.

Types are formed out of atomic types by means of two conditionals $\rightarrow, \leftarrow$ and product $\circ$. Γ, Δ will denote finite strings of types. Let $\Gamma \vdash A$ mean: there is a deduction tree of A with yield Γ. The deduction system of BCGs can be defined as a sequential system with axioms:

(Id) $A \vdash A$,

and inference rules

(E$\rightarrow$) if $\Gamma \vdash A$ and $\Delta \vdash A \rightarrow B$, then $\Gamma, \Delta \vdash B$,

(E$\leftarrow$) if $\Gamma \vdash B \leftarrow A$ and $\Delta \vdash A$, then $\Gamma, \Delta \vdash B$.

The Lambek calculus **L** results from completing the above elimination rules by introduction rules:

(I$\rightarrow$) if $A, \Gamma \vdash B$, then $\Gamma \vdash A \rightarrow B$,

(I$\leftarrow$) if $\Gamma, A \vdash B$, then $\Gamma \vdash B \leftarrow A$,

where $\Gamma \neq \Lambda$ (dropping this constraint leads to the stronger system **L1**). The rules for product are

$(E\circ)$ if $\Delta \vdash A \circ B$ and $\Gamma, A, B, \Gamma' \vdash C$, then $\Gamma, \Delta, \Gamma' \vdash C$,

$(I\circ)$ if $\Gamma \vdash A$ and $\Delta \vdash B$, then $\Gamma, \Delta \vdash A \circ B$.

Both **L** and **L1** are closed under the cut rule:

(CUT) if $\Gamma, A, \Gamma' \vdash B$ and $\Delta \vdash A$, then $\Gamma, \Delta, \Gamma' \vdash B$.

Intuitively, rules $(I\rightarrow)$, $(I\leftarrow)$ enable us to employ hypothetical reasoning in categorial grammars. For instance, given the lexical assumption 'John: PN' and the derivable pattern

$$PN, PN \rightarrow S \vdash S,$$

one infers the Montague type raising principle,

$$PN \vdash S \leftarrow (PN \rightarrow S) = NP,$$

which lifts up proper nouns to the type of noun phrase. As usual for natural deduction, the latter derivation can be represented by the lambda term

$$\lambda x_{PN \rightarrow S}.(x_{PN \rightarrow S}\, \mathbf{j}_{PN}),$$

which means that the constant **j** (standing for 'John') is transformed into the family of predicates holding for it. The reader is referred to [5, 7] for a thorough account of natural deduction and the lambda calculus in categorial semantics, and to [18] for a formal analysis of different deduction systems connected with linguistically relevant fragments of the lambda calculus (see also [39] for a version of the lambda calculus appropriate for directional types).

Here we focus on algebraic structures naturally modelling natural deduction systems. *A residuated semigroup* (r. semigroup) is a structure $\mathcal{M} = (M, \circ, \Rightarrow, \Leftarrow, \leqslant)$ such that $(M, \circ)$ is a semigroup, $\leqslant$ is a partial ordering on M, and $\Rightarrow, \Leftarrow$ are binary operations on M, satisfying the equivalences

$$b \leqslant a \Rightarrow c \text{ iff } a \circ b \leqslant c \text{ iff } a \leqslant c \Leftarrow b,$$

for all $a, b, c \in M$.

We describe a general powerset construction of residuated semigroups. Let $\mathcal{A} = (A, \cdot)$ be a semigroup. In the powerset $P(A)$ we define operations $\circ, \rightarrow, \leftarrow$ in the following way:

$$X \circ Y = \{a \cdot b\colon\ a \in X,\ b \in Y\},$$

$$X \Rightarrow Y = \{c \in A\colon\ (\forall a \in X)\, a \cdot c \in Y\},$$

$$X \Leftarrow Y = \{c \in A\colon\ (\forall a \in Y)\, c \cdot a \in X\},$$

for $X, Y \subseteq A$. $(P(A), \circ, \Rightarrow, \Leftarrow, \subseteq)$ is a residuated semigroup; we refer to it as *the powerset r. semigroup over the semigroup* $\mathscr{A}$. If $A = V^+$ is the free semigroup of nonempty finite strings over the lexicon V, then $P(A)$ is the r. semigroup of languages over V.

More general structures are based on frames (U, R) such that U is a nonempty set and R is a ternary relation on U. In $P(U)$ one defines operations $\circ, \Rightarrow, \Leftarrow$ as follows:

$$X \circ Y = \{c\colon (\exists a \in X)(\exists b \in Y)\, R(a,b,c)\},$$

$$X \Rightarrow Y = \{b\colon (\forall a,c)(a \in X,\ R(a,b,c) \text{ imply } c \in Y)\},$$

$$X \Leftarrow Y = \{a\colon (\forall b,c)(b \in Y,\ R(a,b,c) \text{ imply } c \in X)\},$$

for $X, Y \subseteq U$. Following Dunn [20], we call the r. semigroup $(P(U), \circ, \Rightarrow, \Leftarrow, \subseteq)$ *the concrete r. semigroup over the frame* (U, R). Clearly, the powerset r. semigroup over $(A, \cdot)$ is the concrete r. semigroup over (A, R), where $R(a,b,c)$ holds iff $a \cdot b = c$. While ternary frames are characteristic of Kripke style semantics for relevant logics, powerset models are more in the style of universal algebra and naturally related to the algebra of languages.

An assignment of types in an r. semigroup is defined in the standard way. By *a model* we mean a pair $(\mathscr{M}, \mu)$ such that $\mathscr{M}$ is an r. semigroup and μ is an assignment of types in $\mathscr{M}$. The sequent $A_1, \ldots, A_n \vdash A$ is *true* in this model, if

$$\mu(A_1) \circ \cdots \circ \mu(A_n) \leqslant \mu(A),$$

and it is *valid* in $\mathscr{M}$, if it is true in all models $(\mathscr{M}, \mu)$. If the underlying semigroup is a monoid $(M, \circ, 1)$, then the sequent $\vdash A$ is *true* in $(\mathscr{M}, \mu)$, if $1 \leqslant \mu(A)$, and the remaining notions are defined, as above. The powerset r. monoid over the monoid $(A, \cdot, 1)$ is $(P(A), \circ, \Rightarrow, \Leftarrow, \{1\}, \subseteq)$.

Let Σ be a set of sequents. $\mathbf{L}(\Sigma)$ (resp. $\mathbf{L1}(\Sigma)$) denotes the system $\mathbf{L}$ (resp. $\mathbf{L1}$) with (CUT), enlarged with all sequents from Σ as new axioms. Basic completeness theorems for the Lambek calculus are the following:

(T10) Sequents derivable in $\mathbf{L}(\Sigma)$ are precisely those which are valid in all powerset r. semigroups over arbitrary semigroups.

(T11) Sequents derivable in $\mathbf{L1}(\Sigma)$ are precisely those which are valid in all powerset r. monoids over arbitrary monoids.

(T12) If Σ consists of product-free sequents, then product-free sequents derivable in $\mathbf{L}(\Sigma)$ are precisely those which are valid in all powerset r. semigroups over free semigroups.

(T13) If Σ consists of product-free sequents, then product-free sequents derivable in $\mathbf{L1}(\Sigma)$ are precisely those which are valid in all powerset r. monoids over free monoids.

(T14) Sequents derivable in $\mathbf{L}$ (resp. $\mathbf{L1}$) are precisely those which are valid in all powerset r. semigroups (resp. monoids) over free semigroups (resp. monoids).

(T15) Product-free sequents derivable in **L** (resp. **L1**) are precisely those which are valid in all powerset r. semigroups (resp. monoids) over finite semigroups (resp. monoids).

For each of these theorems, the 'only if' direction (soundness) is easy: axioms (Id) are true in all models, and inference rules preserve the truth. Theorems (T12), (T13) were first proven in [8]. One constructs the canonical model $P(\Sigma^+)$, where Σ is the set of types. The canonical assignment f is defined as follows:

$$f(p)=\{\Gamma\colon \Gamma\vdash_{\mathbf{L}} p\}.$$

By induction on type A, one proves

$$f(A)=\{\Gamma\colon \Gamma\vdash_{\mathbf{L}} A\},$$

for all product-free types A. That yields

$$A_1,\ldots,A_n\vdash_{\mathbf{L}} A \text{ iff } f(A_1)\circ\cdots\circ f(A_n)\subseteq f(A).$$

The 'only if' direction holds by soundness, and the 'if' direction by (Id) and the latter equality. For **L1**, the argument is similar.

Theorems (T10) and (T11) were proven in [11]. They require a more sophisticated construction. Roughly, one affixes to **L** new rules of the form

(D) if $\Gamma\vdash A\circ B$, then $(\Gamma,1,A\circ B)\vdash A$ and $(\Gamma,2,A\circ B)\vdash B$,

which enable us to decompose each Γ such that $\Gamma\vdash A\circ B$ is derivable into two terms

$$\Gamma=(\Gamma,1,A\circ B)\cdot(\Gamma,2,A\circ B)$$

such that $(\Gamma,1,A\circ B)\vdash A$ and $(\Gamma,2,A\circ B)\vdash B$ are derivable. Of course, the language of **L** must be extended to admit terms of the above form. For the extended system, one constructs the canonical model, following the lines above, but now the underlying semigroup is not a free semigroup.

Theorem (T14) has been proven by Pentus [35] by an approximation of a model with partial models; the proof uses many combinatory tools. Theorem (T15), proven in [9, 17], establishes the finite model property for Lambek systems. One uses generalized powerset models over free semigroups. For a set $K\subseteq\Sigma^+$ such that $K\neq\emptyset$ is closed under nonempty subintervals, one defines the relativized powerset operations $X\circ Y$, $X\Rightarrow Y$, $X\Leftarrow Y$, for $X,Y\subseteq K$,

$$X\circ Y=\{ab\in K\colon a\in X,\ b\in Y\},$$

$$X\Rightarrow Y=\{c\in K\colon (\forall a\in X)(ac\notin K\vee ac\in Y)\},$$

$$X\Leftarrow Y=\{c\in K\colon (\forall a\in Y)(ca\notin K\vee ca\in X)\}.$$

Then, $(P(K), \circ, \Rightarrow, \Leftarrow, \subseteq)$ is an r. semigroup which is isomorphic to the powerset r. semigoup $P(K')$, where $K' = K \cup \{x\}$, x being a new element which takes the part of all strings not in K. One shows that **L** (resp. **L1**) is complete with respect to all models of that kind with a finite K. Details of the proof are somewhat cumbersome.

If powerset models are replaced with concrete models, then analogues of theorems (T10) and (T11) can be obtained by a modification of the well known Stone representation theorem for Boolean algebras. One shows:

(T16) Each residuated semigroup is embeddable into a concrete residuated semigroup.

This representation theorem has been proven in [20] in the following way. Let $\mathcal{M}$ be an r. semigroup. A set $\nabla \subseteq M$ is called *a cone*, if $a \in \nabla$ and $a \leqslant b$ entails $b \in \nabla$. Take U equal to the set of cones, and define a ternary relation R on U:

$$R(\nabla_1, \nabla_2, \nabla_3) \text{ iff } (\forall a, b)(a \in \nabla_1,\ a \Rightarrow b \in \nabla_2 \text{ imply } b \in \nabla_3).$$

Then, the mapping $h(a) = \{\nabla\colon a \in \nabla\}$ is a monomorphism of the r. semigroup $\mathcal{M}$ into the concrete r. semigroup $P(U)$.

Interestingly, if R is replaced with the binary operation

$$\nabla_1 \cdot \nabla_2 = \{b\colon (\exists a \in \nabla_1) a \Rightarrow b \in \nabla_2\},$$

then $(U, \cdot)$ is a semigroup, and one can consider the powerset r. semigroup $P(U)$ instead of the concrete r. semigroup. Yet, the mapping h, defined above, is merely a homomorphism with respect to $\Rightarrow, \Leftarrow$ and $\leqslant$, but not $\circ$. Thus, using the method above, one can show that each $\Rightarrow, \Leftarrow$-reduct of an r. semigroup is embeddable into a powerset r. semigroup. To prove the full representation theorem:

(T17) Each residuated semigroup is embeddable into a powerset residuated semigroup,

one needs decomposition methods of the proof of theorems (T10) and (T11).

Other representation and completeness theorems for Lambek systems have been obtained in [2, 33, 29] with respect to algebras of binary relations. These algebras fit an interesting interpretation of Lambek systems as logics of programs (procedures).

We pass to categorial grammars. *Lambek categorial grammars* (LCGs) result from enriching BCGs with the full strength of the Lambek calculus (here, we restrict ourselves to product-free types). For any type A, the LCG G determines the language $L_A(G) \subseteq V_G^+$ which consists of all strings on V_G, being assigned type A by G. On the other hand, given languages $L_p(G)$, for atomic types p, languages $L^A(G)$ are uniquely determined by powerset operations in the algebra $P(V_G^+)$:

$$L^p(G) = L_p(G), \quad \text{for atomic } p,$$

$$L^{A \to B}(G) = L^A(G) \Rightarrow L^B(G),$$

$$L^{B \leftarrow A}(G) = L^B(G) \Leftarrow L^A(G).$$

There arises a natural problem of compatibility of languages $L_A(G)$ and $L^A(G)$. As shown in [8], no Lambek grammar G is *complete*, that means, it cannot fulfil $L^A(G) = L_A(G)$, for all types A. A weaker condition is *correctness*: $L_A(G) \subseteq L^A(G)$, for all types A. It is easy to show that G is correct, if $v : A$ in the sense of G entails $v \in L^A(G)$, for all lexical atoms v and types A. Each Lambek grammar can be extended to a correct Lambek grammar by introducing new lexical atoms: for any type B which appears in lexical assumptions as a subtype, one introduces a new atom v_B and a new assumption $v_B : B$. The new grammar G' is conservative with respect to the initial grammar G:

$$L_A(G) = L_A(G') \cap V_G^+.$$

Correct Lambek grammars are precisely those whose language family $L_p(G)$, for atomic types p, is a minimal solution of the lexical postulates $v : A$. Thus, for correct Lambek grammars, languages generated by the grammar can be characterized in Algol-like style, as the minimal languages satisfying a system of postulates.

Problems of the relation of LCGs (and categorial grammars based on other systems of that kind, e.g. nonassociative and/or commutative) to the Chomsky hierarchy were considered by several authors [5, 10, 13, 26, 27, 34]. In [34], LCGs are shown to be weakly equivalent to CFGs. Methods of proofs are of proof-theoretic rather than typically algebraic character, hence we do not discuss them here (actually, in [13, 26, 27], the characterization of P-languages of BCGs, stated in (T3), has been employed to prove the P-equivalence of Nonassociative Lambek Grammars and BCGs). An extensive account of proof-theoretic methods in categorial grammar (in connection with algebraic structures) can be found in [18].

Most questions discussed above can be extended toward abstract algebras $(A, \mathscr{F})$, where $\mathscr{F}$ is a set of operations in the universe A. With each operation f, of arity $n \geqslant 1$, we associate residuation operations f/i, $i = 1, \ldots, n$, satisfying the equivalence

$$f(\ldots, a_i, \ldots) \leqslant a \text{ iff } a_i \leqslant (f/i)(\ldots, a, \ldots),$$

where $\leqslant$ is a partial ordering on A. If the residuation operations exist, for each $f \in \mathscr{F}$, then the structure $(A, \mathscr{F}, \leqslant)$ is called *a residuated algebra*. Given an algebra $(A, \mathscr{F})$, *the powerset residuated algebra* over this algebra is defined as the set $P(A)$ with $\subseteq$ taking the part of partial ordering, and operations f and f/i given by

$$f(X_1, \ldots, X_n) = \{f(a_1, \ldots, a_n) : (\forall i) a_i \in X_i\},$$

$$(f/i)(X_1, \ldots, X_n) = \{a_i : (\forall j \neq i)(\forall a_j \in X_j) f(a_1, \ldots, a_n) \in X_i\}.$$

Residuated algebras as general frames for Lambek style categorial grammars have been proposed in [15, 18]. They correspond to minimal multimodal systems considered in [32, 29] which account for different composition modes in natural language. *The Generalized Lambek Calculus* **GL** (in the natural deduction form) is based on axioms

(Id) and rules:

(Ef) if $\Delta \vdash f(A_1,\ldots,A_n)$, and $\Gamma[f[A_1,\ldots,A_n]] \vdash C$, then $\Gamma[\Delta] \vdash C$,

(If) if $\Gamma_i \vdash A_i$, $i=1,\ldots,n$, then $f[\Gamma_1,\ldots,\Gamma_n] \vdash f(A_1,\ldots,A_n)$,

(Ef/i) if $\Gamma_i \vdash (f/i)(A_1,\ldots,A_n)$ and $\Gamma_j \vdash A_j$, for all $j \neq i$, then $f[\Gamma_1,\ldots,\Gamma_n] \vdash A_i$,

(If/i) if $f[A_1,\ldots,\Gamma,\ldots,A_n] \vdash A_i$, then $\Gamma \vdash (f/i)(A_1,\ldots,A_n)$,

(CUT) if $\Gamma[A] \vdash B$ and $\Delta \vdash A$, then $\Gamma[\Delta] \vdash B$.

Again, the rule (CUT) can be eliminated from the pure system **GL**. Now, **GL** is complete with respect to residuated algebras (that is an easy application of Lindenbaum algebras), and it is also complete with respect to powerset residuated algebras. The latter theorem has been proven in [28] by means of the following representation theorem: each residuated algebra is embeddable into a powerset residuated algebra. The proof goes by a generalization of the proof of (T10) in [11].

References

[1] K. Ajdukiewicz, Die syntaktische Konnexität, Studia Philos. 1 (1935) 1–27.

[2] H. Andréka, S. Mikulás, Lambek calculus and its relational semantics, J. Logic Language Inform. 3 (1994) 1–38.

[3] Y. Bar-Hillel, A quasi-arithmetical notation for syntactic description, Language 29 (1953) 47–58.

[4] Y. Bar-Hillel, C. Gaifman, E. Shamir, On categorial and phrase structure grammars, Bull. Res. Council Israel F9 (1960) 155–166.

[5] J. van Benthem, Essays in Logical Semantics, Reidel, Dordrecht, 1986.

[6] J. van Benthem, Categorial equations, in: E. Klein, J. van Benthem (Eds.), Categories, Polymorphism and Unification, University of Amsterdam, 1987.

[7] J. van Benthem, Language in Action. Categories, Lambdas and Dynamic Logic, North-Holland, Amsterdam, 1991.

[8] W. Buszkowski, Compatibility of a categorial grammar with an associated category system, Z. Math. Logik Grundlagen Math. 28 (1982) 229–238.

[9] W. Buszkowski, Some decision problems in the theory of syntactic categories, Z. Math. Logik Grundlagen Math. 28 (1982) 539–548.

[10] W. Buszkowski, The equivalence of unidirectional Lambek categorial grammars and context-free grammars, Z. Math. Logik Grundlagen Math. 31 (1985) 369–384.

[11] W. Buszkowski, Completeness results for Lambek syntactic calculus, Z. Math. Logik Grundlagen Math. 32 (1986) 13–28.

[12] W. Buszkowski, Typed functorial languages, Bull. Polish Academy Sci. Math. 34 (1986) 495–505.

[13] W. Buszkowski, Generative capacity of nonassociative Lambek calculus, Bull. Polish Academy Sci. Math. 34 (1986) 507–516.

[14] W. Buszkowski, Solvable problems for classical categorial grammars, Bull. Polish Academy Sci. Math. 35 (1987) 373–382.

[15] W. Buszkowski, Generative power of categorial grammars, in: R.T. Oehrle, E. Bach, D. Wheeler (Eds.), Categorial Grammars and Natural Language Structures, Reidel, Dordrecht, 1988, pp. 69–94.

[16] W. Buszkowski, Gaifman's theorem on categorial grammars revisited, Studia Logica 47 (1988) 23–33.

[17] W. Buszkowski, The finite model property for BCI and related systems, Studia Logica 57 (1996) 303–323.

[18] W. Buszkowski, Mathematical linguistics and proof theory, in: J. van Benthem, A. ter Meulen (Eds.), Handbook of Logic and Language, Elsevier, Amsterdam, 1997, pp. 683–736.

[19] W. Buszkowski, G. Penn, Categorial grammars determined from linguistic data by unification, Studia Logica 49 (1990) 431–454.
[20] J.M. Dunn, Partial gaggles applied to logics with restricted structural rules, in: K. Došen, P. Schroeder-Heister (Eds.), Substructural Logics, Clarendon Press, Oxford, 1993, pp. 63–108.
[21] G. Gazdar, C. Mellish, Natural Language Processing in PROLOG, Addison-Wesley, Reading, MA, 1989.
[22] J.Y. Girard, Linear logic, Theoret. Comput. Sci. 50 (1987) 1–102.
[23] J. Hopcroft, J. Ullman, Introduction to Automata Theory, Languages and Computation, Addison-Wesley, Reading, MA, 1979.
[24] M. Kanazawa, Learnable classes of categorial grammars, Ph.D. Thesis, Dept. of Linguistics, University of Stanford, 1994.
[25] M. Kanazawa, Identification in the limit of categorial grammars, J. Logic Language Inform. 5 (1996) 115–155.
[26] M. Kandulski, The equivalence of nonassociative Lambek categorial grammars and context-free grammars, Z. Math. Logik Grundlagen Math. 34 (1988) 41–52.
[27] M. Kandulski, Normal form of derivations in the nonassociative and commutative Lambek calculus with product, Math. Logic Quart. 39 (1993) 103–114.
[28] M. Kołowska-Gawiejnowicz, Powerset residuated algebras and generalized Lambek calculus, Math. Logic Quart. 43 (1997) 60–72.
[29] N. Kurtonina, Frames and labels. A modal analysis of categorial inference, Ph.D. Thesis, Dept. of Linguistics, University of Utrecht, 1995.
[30] J.W. Lloyd, Foundations of Logic Programming, Springer, Berlin, 1987.
[31] J. Marciniec, Learning categorial grammars by unification with negative constraints, J. Appl. Non-Classical Logics 4 (1994) 181–200.
[32] M. Moortgat, Categorial type logics, in: J. van Benthem, A. ter Meulen (Eds.), Handbook of Logic and Language, Elsevier, Amsterdam, 1997, pp. 93–177.
[33] N. Pankratiev, On the completeness of the Lambek calculus with respect to relativized relational semantics, J. Logic, Language Inform. 3 (1994) 293–246.
[34] M. Pentus, Lambek grammars are context-free, in: Proc. 8th Ann. IEEE Symp. on Logic in Computer Science, Montreal, 1993.
[35] M. Pentus, Lambek calculus is L-complete, Report LP 93-14, Institute for Logic, Language and Computation, University of Amsterdam, 1993.
[36] F. Pereira, S. Shieber, Prolog and Natural Language Analysis, CSLI Lecture Notes, vol. 10, Menlo Park, 1987.
[37] H. Rasiowa, An algebraic approach to non-classical logics, North Holland and Polish Scientific Publishers, Amsterdam, Warszawa, 1974.
[38] J.W. Thatcher, Characterizing derivation trees of context-free grammars through a generalization of finite automata theory, J. Comput. Systems Sci. 1 (1967) 317–322.
[39] H. Wansing, The logic of information structures, Ph.D. Thesis, Dept. of Mathematics and Computer Science, University of Amsterdam, 1992.