W. Buszkowski

Type Logics in Grammar

Introduction

Type logics are logics whose formulas are interpreted as types. For instance, $A \to B$ is a type of functions (procedures) which send inputs of type A to outputs of type B, and $A \otimes B$ is a type of pairs (f, g) such that f is of type A and g is of type B. The scope of possible realizations is huge: from constructivism in mathematics to logics of computation, from combinators and lambda calculus to linear logics, from type theories of Russell and Church to theories of syntax, proposed by Leśniewski, Ajdukiewicz, Chomsky, Curry and Lambek.

In this paper I try to illuminate the current state of the latter theories, often characterized as logical grammar or, more generally, formal grammar. Not every logician would be inclined to attach this region to *logic proper*. Fortunately, numerous contributions of last thirty years have lifted up the logic of grammar to a vivid research branch with a well-defined conceptual framework and promising connections with linguistics and computer science (also philosophy, cognitive science, semantics). From the standpoint of logic, formal systems appearing in this discipline belong to substructural logics, since their sequential formats abandon structural rules (Weakening, Contraction, Exchange); see [69].

Hiż [34] dubbed the term *grammar logicism* for the thesis: grammar reduces to logic. Not discussing all subtleties of this standpoint, I believe Hiż soundly characterizes the real content of so-called formal linguistics with its standard syntactic devices (automata, grammars, trees, transformations) which are conceptually and methodologically akin to logical entities (routines of computation, theories, proofs, deductive systems). It seems to be important that grammatical structures influence the creation and evolution of new logical formalisms, as e.g. the Lambek calculus and noncommutative linear logics.

Reduction of grammar to logic need not be restricted to the dimension of

---

V. F. Hendricks and *J. Malinowski* (eds.),
Trends in Logic: 50 Years of Studia Logica

Trends in Logic **21**: 321–366, 2003.

syntax. Since Montague [57], semantics of natural language has extensively been studied within philosophical logic. Semantical problems seem to be more sophisticated and solutions less definitive. I am inclined to remove any strict borderline between syntax and semantics; the former is a representation of the latter, the latter must be anchored in the former. Type grammars are especially close to the description of syntax according to general semantic paradigms. Connections of type grammars with Montague semantics are based on the Curry-Howard isomorphism between proofs in Natural Deduction format and lambda terms. For Lambek logics, main results are due to van Benthem [76, 78], Moortgat [58, 59] and their collaborators. In this survey, semantical aspects are briefly mentioned at the beginning of section 3.

Grammars based on logics of types are traditionally called *categorial grammars* (after Bar-Hillel et al. [5]). From the modern perspective, the term *type grammars* seems to be more adequate: all formal grammars classify language expressions into categories, but a characteristic feature of type grammars is the usage of (logical) types as carriers of grammatical information. The present paper mainly focuses on type grammars and their logics, which evidently reflects the author's predilection and experience. Hopingly, it shows a more general significance of some central ideas.

We begin with the simplest formalization of type grammar: Classical Categorial Grammar (CCG) of Ajdukiewicz [3] and Bar-Hillel [5]. Ajdukiewicz was strongly influenced by logical syntax, designed by Leśniewski [54], and the idea of universal grammar, due to Husserl [36]. The logic of CCG is a purely implicational propositional logic with modus ponens and no axioms; in the bidirectional format, stemming from Lambek [49] and Bar-Hillel [5], it admits two conditionals $A \to B$ and $B \leftarrow A$ and two MP-schemes. We discuss three major topics concerning CCG: (i) syntactic unambiguity (related to Polish notation), (ii) connections with Chomsky grammars and tree automata, (iii) learnability. The latter exhibits a new, promising direction of formal learning theory, employing logical unification to design successful algorithms for language acquisition.

Next, we consider full-fledged type logics, originating in Syntactic Calculus of Lambek [49]. They are closely connected with other logical formalisms, as e.g. linear logics, lambda calculus and combinators, labelled deductive systems. First, we present different versions of type logics and discuss their proof-theoretic and computational aspects. Second, we sketch a landscape of models of these logics. We also regard central problems related to type grammars based on these logics.

This survey is complementary to the author's [14]. Certain topics, largely

discussed there, e.g. algebra of syntactic structures and modelling type logics in the lambda calculus, are almost completely omitted here. On the other hand, we present in more detail some special matters, characteristic of current research, as e.g. learning theory and computational complexity. In section 3, we mix algebraic models and computational problems, because they are mutually dependent. Algebraic structures influence the shape of formal systems and are useful for illuminating proof-theoretic properties, i.e. the subformula property.

We report results of many people, including the author, who apologizes for the great number of self-citations in comparing with others. The reason is obvious: the author focuses on problems he has been involved in and informs the reader where she may find proofs abandoned here. In compensation, let me emphasize a particular role of two persons. Joachim Lambek has originated some central ideas, and Johan van Benthem has shown their broader significance for logic.

Classical Categorial Grammar

Pr denotes a set of *primitive types* (propositional variables). *Types* (formulas) are formed out of two conditionals $\rightarrow$ (right arrow) and $\leftarrow$ (left arrow). We denote types by A, B, C, D. Γ, Δ denote finite sequences (strings) of types, and $\Gamma\Delta$ denotes the concatenation of Γ and Δ. *Sequents* are expressions of the form $\Gamma \vdash A$. The empty string is denoted by ϵ.

The logic AB (after Ajdukiewicz and Bar-Hillel) can be presented as a structured consequence relation (in the sense of Gabbay [28]), which is determined by MP-schemes:

$$\text{(MP-R) } A(A \rightarrow B) \vdash B, \text{ (MP-L) } (B \leftarrow A)A \vdash B,$$

and the following structural rules:

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

$$\text{(CUT) } \frac{\Gamma A \Gamma' \vdash B;\ \Delta \vdash A}{\Gamma \Delta \Gamma' \vdash B}.$$

Equivalently, AB can be presented as a rewriting system, based on reduction rules:

$$A, A \rightarrow B \Rightarrow B \text{ and } B \leftarrow A, A \Rightarrow B.$$

Then, $\Gamma \vdash_{AB} A$ iff Γ reduces to A by a finite number of applications of reduction rules. We write $\Gamma \vdash_{AB} A$ for: $\Gamma \vdash A$ is provable in AB, and similarly for other systems. Clearly, if $\Gamma \vdash_{AB} A$ then $\Gamma \neq \epsilon$.

Ajdukiewicz [3] introduced the $(\leftarrow)-$fragment of this logic; precisely, he regarded multiple-argument types $B \leftarrow A_1, \ldots, A_n$ with reduction rules:

$$(B \leftarrow A_1, \ldots, A_n), A_1, \ldots, A_n \Rightarrow B.$$

In Natural Deduction (ND) format, AB is axiomatized by (Id) and two conditional elimination rules:

$$(\rightarrow\text{E})\ \frac{\Gamma \vdash A;\ \Delta \vdash A \rightarrow B}{\Gamma\Delta \vdash B},\ (\leftarrow\text{E})\ \frac{\Gamma \vdash B \leftarrow A;\ \Delta \vdash A}{\Gamma\Delta \vdash B}.$$

Proofs in the ND-format determine the so-called *functor-argument structures* (fa-structures) on antecedents of provable sequents. Types are atomic fa-structures. If X, Y are fa-structures, then $(XY)_1$ and $(XY)_2$ are fa-structures. X is the functor, and Y is the argument of $(XY)_1$, and the converse holds for $(XY)_2$. The logic ABs (i.e. AB in structure form) arises from AB (in ND-format) by replacing $\Gamma$with X, Δ with Y in premises of rules, and $\Gamma\Delta$ with $(XY)_2$ (resp. $(XY)_1$) in the conclusion of $(\rightarrow\text{E})$ (resp. $(\leftarrow\text{E})$). Every proof of $\Gamma \vdash A$ in AB determines a unique fa-structure X on Γ such that $X \vdash_{ABs} A$. The same could be done for the original version of AB.

Syntactic unambiguity

It is well known that the Polish notation is unambigous, that means, the structure of a formula is uniquely determined by the string of symbols. This property is actually a property of the logic AB, restricted to one conditional. Since this a rather untypical property of a logical system, and it is in focus of Ajdukiewicz [3], we have decided to discuss the problem in some detail.

For any structure X, there is at most one type A such that $X \vdash_{ABs} A$. One computes this type by a successive replacement of every substructure of X of the form $(A(A \rightarrow B))_2$ or $((B \leftarrow A)A)_1$ with type B. This property will be referred to as *type unambiguity* of ABs. Also, every provable sequent $X \vdash A$ has a unique proof tree in the ND-format of ABs (*proof unambiguity*).

AB is both type and proof ambiguous. $(A \leftarrow A)A(A \rightarrow A) \vdash A$ has two proofs in AB which determine structures $(C(AD)_2)_1$ and $((CA)_1D)_2$, for $C = A \leftarrow A$, $D = A \rightarrow A$. For $\Gamma = ((A \leftarrow D) \leftarrow A)AD$, with D as above, we have both $\Gamma \vdash_{AB} A \leftarrow D$ and $\Gamma \vdash_{AB} A$.

One-directional fragments of AB are both type and proof unambigous, which is a nonstandard proof-theoretic property, clearly related to (generalized) Polish notation: a string of symbols supplied with one-directional types admits a unique syntactic analysis (see [13] for a discussion and the

proof for the general case of multiple-argument types). We sketch the proof for the $(\leftarrow)-$fragment of AB.

In types $B \leftarrow A_1 \leftarrow \cdots \leftarrow A_n$ parentheses are grouped to the left. If D is of the above form, with B primitive, then $C < D$ means that C equals $B \leftarrow A_1 \leftarrow \cdots \leftarrow A_i$, for some $0 \leq i < n$. The relation $<$ is irreflexive, transitive and quasi-linear: if $A < C$ and $B < C$, then $A < B$ or $A = B$ or $B < A$. The following lemmas are crucial.

(L1) If $A\Gamma \vdash_{AB} B$ and $\Gamma \neq \epsilon$ then $B < A$.

(L2) If $B < A$ then $A\Gamma \vdash_{AB} B$, for some $\Gamma \neq \epsilon$.

(L3) If $\Gamma \vdash_{AB} A$ then there is no $\Delta \neq \epsilon$ such that $\Gamma\Delta \vdash_{AB} A$.

We prove (L3). We use the following property of the restricted AB: if $\Gamma \vdash_{AB} A$ and Γ has a length greater than 1, then $\Gamma = (A_2 \leftarrow A_1)\Gamma_1\Gamma_2$ with $\Gamma_1 \vdash_{AB} A_1$ and $A_2\Gamma_2 \vdash_{AB} A$. The proof of (L3) proceeds by induction on the length of Γ.

Let Γ be of length 1. Assume $\Gamma \vdash_{AB} A$ and $\Gamma\Delta \vdash_{AB} A$, for $\Delta \neq \epsilon$. Then, $\Gamma = A$, and consequently, $A < A$, by (L1). Contradiction.

Let Γ be of length greater than 1. Assume the same as above. Using the property, mentioned above, we get types A_1, A_2 and nonempty strings Γ_i, Δ_i, for $i = 1, 2$, such that:

1. $\Gamma = (A_2 \leftarrow A_1)\Gamma_1\Gamma_2$,
2. $\Gamma\Delta = (A_2 \leftarrow A_1)\Delta_1\Delta_2$,
3. $\Gamma_1 \vdash_{AB} A_1$, $\Delta_1 \vdash_{AB} A_1$,
4. $A_2\Gamma_2 \vdash_{AB} A$, $A_2\Delta_2 \vdash_{AB} A$.

Clearly, the length of Γ_1 is less than the length of Γ, and similarly for Γ_2. By the induction hypothesis, neither Γ_1 is a proper initial segment of Δ_1, nor Δ_1 is a proper initial segment of Γ_1. Consequently, $\Gamma_1 = \Delta_1$. Then, Γ_2 is a proper initial segment of Δ_2, hence $A_2\Gamma_2$ is a proper initial segment of $A_2\Delta_2$, and its length is less than that of Γ. This contradicts the induction hypothesis.

We prove type unambiguity. Assume $\Gamma \vdash_{AB} A$ and $\Gamma \vdash_{AB} B$, with $A \neq B$. Let C be the first type in Γ. Since $A \neq B$, then Γ is of length greater than 1. By (L1), $A < C$ and $B < C$. Consequently, $A < B$ or $B < A$. If $A < B$ then there exists $\Delta \neq \epsilon$ such that $B\Delta \vdash_{AB} A$, by (L2),

which yields $\Gamma\Delta \vdash_{AB} A$, by (CUT). This contradicts (L3). The case $B < A$ is symmetrical.

We prove proof unambiguity. Precisely, we show that every provable sequent admits a unique ND proof tree in the restricted AB. Let d_1, d_2 be proofs of $\Gamma \vdash A$. We prove $d_1 = d_2$, by induction on the length of Γ.

Assume Γ is of length 1. Then, $\Gamma = A$, and d_1, d_2 consist of the only application of (Id). Assume Γ is of length greater than 1. Then, both d_1 and d_2 use ($\leftarrow$E). We consider the last application of this rule in either proof. For d_1, let the premises be $\Gamma_1 \vdash A \leftarrow C$, $\Delta_1 \vdash C$, and for d_2, let them be $\Gamma_2 \vdash A \leftarrow D$, $\Delta_2 \vdash D$. We show $C = D$. Assume $C \neq D$. Then, Γ_1 or Γ_2 is of length greater than 1, hence $A \leftarrow C < A \leftarrow D$ or $A \leftarrow D < A \leftarrow C$, by (L1) and properties of $<$. This is impossible, by the definition of $<$. Now, Γ_1 is an initial segment of Γ_2 or conversely. Neither of them can be a proper initial segment of the other, by (L3), which yields $\Gamma_1 = \Gamma_2$, and consequently, $\Delta_1 = \Delta_2$. By the induction hypothesis, subproofs of d_1, d_2 leading to the left premise are equal, and similarly for the right premise. Consequently, $d_1 = d_2$.

Ajdukiewicz [3] has designed a simple parsing procedure. Given a string Γ, find the left-most occurrence of $(B \leftarrow A)A$ and replace it with B. Then, $\Gamma \vdash_{AB} C$ iff Γ reduces to C by finitely many applications of left-most reduction. As shown in [13], this is not true for multiple-argument types (considered by Ajdukiewicz) but is true for one-argument types, considered here. The Ajdukiewicz procedure is fully deterministic and shows, in fact, that languages generated by rigid type grammars based on this fragment of AB are deterministic context-free languages, i.e. they can be recognized by deterministic push-down automata (rigid grammars are defined below).

CCG versus other formal grammars

A CCG can formally be defined as a triple $G = (V_G, I_G, S_G)$ such that V_G is a nonempty finite lexicon (alphabet), I_G is a finite relation between elements of V_G and types, and $S_G \in$Pr; V_G, I_G, S_G are called *the lexicon*, *the initial type assignment* and *the principal type*, respectively, of G. We write $G : v \mapsto A$ for $(v, A) \in I_G$. We say that G *assigns* type A to string $a = v_1 \dots v_n$, $v_i \in V_G$, if there are types A_i such that $G : v_i \mapsto A_i$, for $i = 1, \dots, n$, satisfying $A_1 \dots A_n \vdash_{AB} A$; we write $a \mapsto_G A$. *The language* of G ($L(G)$) is the set of all strings a such that $a \mapsto_G S_G$. Other type grammars, considered later on, are defined in a similar way; only AB is to be replaced by a different type logic.

The fa-structures on V_G are defined as fa-structures of types except that

atomic fa-structures are elements of V_G. By $F(V_G)$ we denote the set of fa-structures on V_G. We say that G assigns type A to structure $X \in F(V_G)$ if there is an fa-structure X' of types which arises from X by replacing each atom v by a type B such that $G : v \mapsto B$, satisfying $X' \vdash_{ABs} A$; we again write $X \mapsto_G A$. *The f-language* of G ($L^f(G)$) is the set of all $X \in F(V_G)$ such that $X \mapsto_G S_G$.

For instance, let G admit the following assignment:

$$\text{Joan} \mapsto A, \text{ works} \mapsto B, \text{ hardly} \mapsto C,$$

where A =PN (proper noun), $B = A \rightarrow$S, $C = B \rightarrow B$, and S_G =S (sentence). Then, $L^f(G)$ contains structures:

1. (Joan works)$_2$,
2. (Joan (works hardly)$_2$)$_2$,
3. (Joan ((works hardly)$_2$ hardly)$_2$)$_2$,

and so on. $L(G)$ contains the strings resulting from the above structures after one has dropped all structure markers.

The above grammar is a *rigid* CCG, i.e. it assigns at most one type to any lexical atom. Rigidness is characteristic of grammars designed for formal languages of logic and mathematics, while natural language usually requires several types to be assigned to one lexical atom. The negation 'not' is assigned type S←S, as in the sentence 'not every man works' with structure (not ((every man) works)), but also type $C \leftarrow C$ (C defined as above), as in 'John works not hardly'.

We have used a structure without functor markers; such structures are called *phrase structures* (p-structures). For $X \in F(V_G)$, by $p(X)$ we denote the p-structure resulting from X after one has dropped all functor markers. *The phrase language* of G ($L^p(G)$) consists of all $p(X)$, for $X \in L^f(G)$.

Now, let us turn to some basic notions of Chomsky linguistics. *A context-free grammar* (CF-grammar) is a quadruple $G = (V_G, N_G, R_G, S_G)$ such that V_G, N_G are disjoint finite alphabets, $S_G \in N_G$, and R_G is a finite subset of $N_G \times (V_G \cup N_G)^\star$ ($W^\star$ denotes the set of all finite strings of elements of set W). Elements of V_G, N_G, R_G are called *terminal symbols*, *nonterminal symbols* and *production rules*, respectively, of G, and S_G is called *the initial symbol* of G. Production rules are written $A \mapsto a$ instead of (A, a). We say that string b is *directly derivable* from string a in G (write $a \Rightarrow_G b$) if there exist strings c, d, e and rule $A \mapsto e$ in R_G such that $a = cAd$, $b = ced$. We say that string b is *derivable* from string a in G (write $a \Rightarrow^\star_G b$) if there exists

a sequence $(a_0, \ldots, a_n)$ such that $n \geq 0$, $a_0 = a$, $a_n = b$ and $a_{i-1} \Rightarrow_G a_i$, for all $i = 1, \ldots, n$. *The language* of G ($L(G)$) is the set of all $a \in V_G^\star$ such that $S_G \Rightarrow_G^\star a$.

Grammars G, G' are said to be (weakly) *equivalent* if $L(G) = L(G')$. It is well-known (see e.g. [72]) that every CF-grammar G such that $\epsilon \notin L(G)$ can be transformed into an equivalent CF-grammar G' in *the Chomsky normal form*: all production rules of G' are of the form $A \mapsto v$ or $A \mapsto BC$, where $A, B, C \in N_{G'}$, $v \in V_{G'}$.

The CCG exemplified above can be replaced with a CF-grammar (in the Chomsky normal form) whose production rules are as follows:

$$\mathrm{S} \mapsto AB,\ B \mapsto BC,$$

$$A \mapsto \text{Joan},\ B \mapsto \text{works},\ C \mapsto \text{hardly},$$

with $S_G = \mathrm{S}$, $N_G = \{S, A, B, C\}$, and V_G consisting of lexical atoms 'Joan', 'works' and 'hardly'. A derivation of 'Joan works hardly' is:

$$\mathrm{S} \Rightarrow AB \Rightarrow ABC \Rightarrow \cdots \Rightarrow \text{Joan works hardly}.$$

Every CF-derivation determines a unique p-structure on the derived string; the above derivation leads to structure (Joan (works hardly)). *The phrase language* of CF-grammar G ($L^p(G)$) consists of all p-structures on strings from $L(G)$ which are determined by possible derivations of these strings.

A principal difference between CCG and CF-grammar is that the former is *lexical*, that means: all particular linguistic information is put in the initial assignment of types to lexical atoms, and the derivation procedure is based on universal rules, common for all languages, whereas the latter puts the linguistic information in the production rules which underly the derivation procedure. Lexicality is characteristic of all basic kinds of type grammar: the universal rules for derivation procedures are provable sequents of some logics, being independent of the particular language. In section 3, we shall show that lexicality is, actually, a restriction: larger classes of languages can only be described by type grammars avoiding lexicality.

The Gaifman theorem [5] establishes the (weak) equivalence of CCG's and CF-grammars (for $\epsilon-$free languages). It is easy to show that every CCG is equivalent to some CF-grammar. Let G be a CCG, and let T_G denote the set of all types appearing in I_G. By $T(G)$ we denote the set of all subtypes of types from T_G. Clearly, $T(G)$ is finite and contains all types assigned by

G to any strings. A CF-grammar G' is defined by: $V_{G'} = V_G$, $N_{G'} = T(G)$, $S_{G'} = S_G$, and $R_{G'}$ consists of all rules:

$$B \mapsto A(A \to B),\ B \mapsto (B \leftarrow A)A,$$

for $(A \to B), (B \leftarrow A) \in T(G)$, and all lexical rules $A \mapsto v$, for $(v, A) \in I_G$. One easily proves:

$$A_1 \dots A_n \vdash_{AB} A \text{ iff } A \Rightarrow^{\star}_{G'} A_1 \dots A_n,$$

for all $A_i, A \in T(G)$, and consequently, $L(G) = L(G')$.

The converse direction is more sophisticated. It is easier if one assumes that the CF-grammar G is in *the Greibach normal form*, that means: all production rules are of the form $A \mapsto v$, $A \mapsto vB$ or $A \mapsto vBC$, for $A, B, C \in N_G$, $v \in V_G$ (every CF-grammar G such that $\epsilon \notin L(G)$ is equivalent to a CF-grammar in the Greibach normal form [72]). We identify nonterminal symbols of G with primitive types. A CCG G' is defined by: $V_{G'} = V_G$, $S_{G'} = S_G$, and $I_{G'}$ consists of:

1. all pairs (v, A) such that $(A \mapsto v) \in R_G$,
2. all pairs $(v, A \leftarrow B)$ such that $(A \mapsto vB) \in R_G$,
3. all pairs $(v, A \leftarrow C \leftarrow B)$ such that $(A \mapsto vBC) \in R_G$.

By induction on n, one proves:

$$A \Rightarrow^{\star}_{G} v_1 \dots v_n \text{ iff } v_1 \dots v_n \mapsto_{G'} A,$$

for all $A \in N_G$, $v_i \in V_G$, which yields $L(G) = L(G')$.

Actually, the Gaifman theorem has been proven earlier than the Greibach normal form theorem. Since the CCG, constructed by Gaifman, is precisely of the above form, then his proof can be treated as the first proof of the Greibach normal form theorem for CF-grammars (see an analysis of this proof in [10]).

CCG's are not equivalent to CF-grammars on the level of p-structures. Let $P(V)$ denote the set of all p-structures on alphabet V. Let $L \subseteq P(V)$. We say that $X \in P(V)$ is equivalent to $Y \in P(V)$ with respect to L (write $X \sim_L Y$) if, for all $Z \in P(V)$, $Z[X] \in L$ iff $Z[Y] \in L$ (as usual in logic, $Z[X]$ denotes Z with a distinguished occurrence of substructure X, and $Z[Y]$ denotes the result of replacing X with Y in Z). By the classical theorem of Thatcher, $L = L^p(G)$, for some CF-grammar G, iff the relation $\sim_L$ is of finite index. By *the external degree* of $X \in P(V)$ (think of X as a tree)

we mean the length of the shortest branch of X, and *the degree* of X is the maximal external degree of substructures of X. For instance, v is of degree 0, (vw) is of degree 1, and $((vw)(v'w'))$ is of degree 2. It has been shown in [9] that $L = L^p(G)$, for some CCG G, iff both $\sim_L$ is of finite index and all structures in L are of bounded degree. Accordingly, phrase languages of CCG's are a narrower class than those of CF-grammars. For instance, the CF-grammar given by rules S$\mapsto$SS, S$\mapsto v$ produces all possible p-structures on $\{v\}$, hence its phrase language is of unbounded degree, and consequently, it cannot be generated by any CCG.

Phrase languages generated by CCG's are regular tree languages in the sense of tree automata (see [30]); this follows from the above characterization of them as languages of finite index. Tiede [74] obtains analogous results for ND proof trees associated with CCG's and other kinds of categorial grammars. For grammars based on the nonassociative Lambek calculus, some related results have been proven by Kandulski [43].

A similar characterization can be given for functor languages generated by CCG's: for $L \subseteq F(V)$, there is a CCG G such that $L = L^f(G)$ iff both the relation $\sim_L$ is of finite index and all structures in L are of bounded functor degree (one counts the length of functor branches only; see [14]). Consequently, functor languages of CCGs are regular tree languages. Standard techniques of tree automata [30] yield the decidability of the emptiness problem, the inclusion problem and the equality problem for these languages; for a discussion, see [14]. In particular, the problem of whether $L^f(G) \subseteq L^f(G')$ is decidable, which is crucial for some learnability results in the next subsection.

Learnability

Formal learning theory, stemming from Gold [31], is a theory of language acquisition, based on formal grammars and recursion theory. It has evident links with logical theories of inductive inference and knowledge discovery. Technical results are mainly due to people working in theoretical computer science; see the survey [64].

In [20], the logical method of unification has been applied to design learning procedures for CCG's. Kanazawa [38, 39] uses these procedures to construct convergent learning functions for several classes of CCG's. Negative postulates are considered in [20, 56, 18, 26, 27]. Unification in type grammar has also been studied in e.g. [77, 75].

Due to lexicality (and the functional structure of types), type grammars seem to be especially suitable to experiment learning algorithms employing

unification. Hopingly, similar techniques can be developed for dependency grammars and minimalist grammars (C. Retore, E. Stabler).

We shall describe fundamental lines of this approach for CCG's. First, we recall some basic notions of formal learning theory.

A grammar system is a triple (Ω, E, L) such that Ω and E are countably infinite sets, and L is a function from Ω into the powerset of E. Elements of Ω are called *grammars*, elements of E are called *expressions*, and $L(G)$, for $G \in \Omega$, is called *the language* of G.

W^+ denotes the set of all nonempty strings of elements of W. *A learning function* for the grammar system is a partial function from E^+ into Ω. Intuitively, the learning function assigns grammars to finite samples of a language. Let $(s_i)_{i\in\omega}$ be an infinite sequence of expressions. One says that a learning function φ *converges* on this sequence to $G \in \Omega$ if $\varphi((s_i)_{i<n})$ is defined and equals G, for all but finitely many $n \in \omega$.

Let $\mathcal{G} \subseteq \Omega$. We denote $L(\mathcal{G}) = \{L(G) : G \in \mathcal{G}\}$. A sequence $(s_i)_{i\in\omega}$ is called *a text* for a language $L \subseteq E$ if $L = \{s_i : i \in \omega\}$. One says that a learning function φ *learns* $\mathcal{G}$ if, for every $L \in L(\mathcal{G})$ and every text for L, there is $G \in \mathcal{G}$ such that $L = L(G)$ and φ converges to G on this text. A class $\mathcal{G}$ is said to be (effectively) *learnable* if there exists a computable learning function φ that learns $\mathcal{G}$.

Informally, $\mathcal{G}$ is learnable iff there exists an algorithmic procedure which hypothesizes a grammar on the basis of a finite language sample (it need not be defined on all samples); if the inputs are successive initial segments of any text for any language generated by a grammar from this class, then, after a finite number of steps, the outputs provide a fixed grammar from $\mathcal{G}$ which generates this language.

Let $\mathcal{L}$ be a class of subsets of E (languages on E). One says that $\mathcal{L}$ *admits a limit point* if there exists a stricly ascending chain $(L_n)_{n\in\omega}$ of languages from $\mathcal{L}$ such that the join of this chain belongs to $\mathcal{L}$. It is known that if $L(\mathcal{G})$ admits a limit point then $\mathcal{G}$ is not (even uneffectively) learnable. As a consequence, if $\mathcal{G}$ generates all finite languages and at least one infinite language, then $\mathcal{G}$ is not learnable. This holds for all standard classes of formal grammars, e.g. regular grammars, CF-grammars, CCG's and so on.

Accordingly, learnability can be gained for some restricted classes only, as e.g. context-sensitive grammars with at most k production rules [71]. Kanazawa [38, 39] shows that rigid CCG's and $k-$valued CCG's are learnable.

Wright [81] has defined the following property of a class $\mathcal{L}$ of languages: $\mathcal{L}$ has *finite elasticity* if there exists no pair $((s_i)_{i\in\omega}, (L_i)_{i\in\omega})$, $s_i \in E$, $L_i \in \mathcal{L}$, such that $s_i \notin L_i$ but $s_0, \ldots, s_i \in L_{i+1}$, for all $i \in \omega$. Kapur [44] has proven

that finite elasticity entails the following condition:

(D) for every $L \in \mathcal{L}$, there exists a finite set $D_L \subseteq L$ such that L is the smallest language $L' \in \mathcal{L}$, satisfying $D_L \subseteq L'$.

We prove that finite elasticity is equivalent to a strengthening of (D):

(D') for every $L \subseteq E$, there exists a finite set $D_L \subseteq L$ such that, for every $L' \in \mathcal{L}$, if $D_L \subseteq L'$ then $L \subseteq L'$.

Assume that $\mathcal{L}$ does not satisfy (D'). Then, there exists $L \subseteq E$ such that, for any finite $D \subseteq L$, there is $L' \in \mathcal{L}$ such that $D \subseteq L'$ but $L \not\subseteq L'$. Take $D = \emptyset$. There is $L' \in \mathcal{L}$ such that $L \not\subseteq L'$. Put $s_0 \in L - L'$, $L_0 = L'$. Assume $s_i \in L$ and $L_i \in \mathcal{L}$, for $i = 0, \ldots n$, have already been defined with $s_i \notin L_i$, for $0 \leq i \leq n$, and $s_0, \ldots, s_i \in L_{i+1}$, for $0 \leq i < n$. There is $L' \in \mathcal{L}$ such that $s_0, \ldots, s_n \in L'$ but $L \not\subseteq L'$. Put $s_{n+1} \in L - L'$, $L_{n+1} = L'$. In this way, we define a sequence $(s_i)_{i \in \omega}$ and a sequence $(L_i)_{i \in \omega}$, showing that $\mathcal{L}$ does not have finite elasticity.

Assume that $\mathcal{L}$ satisfies (D'). Let (s_i) be an infinite sequence of expressions. By (D'), there is $n \in \omega$ such that, for any $L' \in \mathcal{L}$, if $s_0, \ldots, s_n \in L'$ then $s_i \in L'$, for all $i \in \omega$. Consequently, there exists no infinite sequence (L_i), of languages from $\mathcal{L}$, such that $s_0, \ldots, s_n \in L_{n+1}$ but $s_{n+1} \notin L_{n+1}$, which proves the finite elasticity of $\mathcal{L}$.

This yields Kapur's theorem, since (D) follows from (D'). If $\mathcal{L}$ is a closure system, that means: $\mathcal{L}$ is closed under arbitrary meets, then $\mathcal{L}$ satisfies (D') iff it admits no infinite ascending chains, that means: there is no infinite sequence (L_i), of languages from $\mathcal{L}$, such that $L_i \subset L_{i+1}$, for all $i \in \omega$ (this observation and the equivalence of finite elasticity with (D') seem to have been overlooked in literature). We prove this fact.

Assume $\mathcal{L}$ satisfies (D'). Let (L_i) be an infinite sequence of languages from $\mathcal{L}$ such that $L_i \subseteq L_{i+1}$, for all $i \in \omega$. Let L be the join of this sequence. Take a finite set $D_L \subseteq L$, as in (D'). Then, $D_L \subseteq L_n$, for some $n \in \omega$. Consequently, $L \subseteq L_n$, hence $L_n = L_{n+1}$. Therefore, $\mathcal{L}$ admits no infinite ascending chains.

Assume that $\mathcal{L}$ is a closure system which does not satisfy (D'). Then, there exists $L \subseteq E$ such that, for every finite $D \subseteq L$, there is $L' \in L$, satisfying $D \subseteq L'$ but $L \not\subseteq L'$. Clearly, L is infinite. Let (s_i) be an infinite sequence of all expressions from L. For every $n \in \omega$, let L_n denote the smallest language $L' \in \mathcal{L}$ such that $s_0, \ldots, s_n \in L'$; it exists, since $\mathcal{L}$ is a closure system. Clearly, $L_n \subseteq L_{n+1}$, for all $n \in \omega$. For every $n \in \omega$, there is $L' \in \mathcal{L}$ such that $s_0, \ldots, s_n \in L'$ but $L \not\subseteq L'$, and consequently, there is

$m > n$ such that $s_m \notin L_n$, which yields $L_n \subset L_m$. Then, there exists an infinite, strictly ascending chain of languages from $\mathcal{L}$.

Let $\mathcal{G} \subseteq \Omega$. We always assume that $\mathcal{G}$ is recursively enumerable and the universal membership problem $s \in L(G)$, for $s \in E$, $G \in \mathcal{G}$, is decidable. We need one more condition:

(MIN) there exists a computable partial function χ from the set of finite subsets of E into $\mathcal{G}$ such that the domain of χ is recursive and, for any finite $D \subseteq E$, the family $\{L \in L(\mathcal{G}) : D \subseteq L\}$ is nonempty iff $\chi(D)$ is defined; if so, $L(\chi(D))$ is a $\subseteq -$minimal language in this family.

We prove a basic theorem, due to Kapur [44]:

(T1) if $\mathcal{G}$ satisfies (MIN) and $L(\mathcal{G})$ has finite elasticity, then $\mathcal{G}$ is learnable.

Assume $\mathcal{G}$ satisfies (MIN), and $L(\mathcal{G})$ has finite elasticity. A learning function φ is defined as follows:

(l0) $\varphi((s_0)) = \chi(\{s_0\})$,

(l1) if $\varphi((s_0, \ldots, s_i)) = G$ is and $s_{i+1} \in L(G)$ then $\varphi((s_0, \ldots, s_{i+1})) = G$,

(l2) $\varphi((s_0, \ldots, s_{i+1})) = \chi(\{s_0, \ldots, s_{i+1}\})$, otherwise.

The relation 'φ is defined on $(s_0, \ldots, s_i)$' is recursive, since both the domain of χ and the relation $s \in L(G)$, for $G \in \mathcal{G}$, are recursive. Consequently, φ is recursive. It is easy to show the following: if $\varphi((s_0, \ldots, s_i)) = G$ is defined, then $L(G)$ is a minimal language in the family of all $L \in L(\mathcal{G})$ such that $s_0, \ldots, s_i \in L$.

Let $L \in L(\mathcal{G})$, and let (s_i) be a text for L. Clearly, $\varphi((s_0, \ldots, s_i))$ is defined, for all $i \in \omega$. Let $D_L \subseteq L$ be as in (D). There exists $n \in \omega$ such that $D_L \subseteq \{s_0, \ldots, s_n\}$. Since L is the smallest language $L' \in \mathcal{L}$ such that $D_L \subseteq L'$, then $L = L(G)$, for $G = \varphi((s_0, \ldots, s_n))$. By (l1), $G = \varphi((s_0, \ldots, s_m))$, for all $m \geq n$.

We have shown that φ learns $\mathcal{G}$. Since φ satisfies (l1), it is *conservative* in the sense of formal learning theory. If conservativity is not required, then the finite elasticity of $L(\mathcal{G})$ is a sufficient condition for the learnability of $\mathcal{G}$ [64].

Now, we apply these methods to CCG. Let us consider the grammar system (Ω, E, L) such that Ω is the class of CCG's (with a fixed lexicon V), $E = F(V)$, and $L(G) = L^f(G)$, for $G \in \Omega$. We assume that all grammars in Ω have the same principal type S. Since V and S are fixed, a grammar G can be identified with I_G.

First, we recall a unification-based learning procedure from [20]. Let $D \subseteq F(V)$ be a nonempty, finite set. We define a CCG $\mathrm{GF}(D)$, called *the general form of a grammar* for D. S is treated as a constant, whereas other primitive types are treated as variables. We assign S to all structures from D and distinct variables to all occurrences of argument substructures of these structures. Then, we assign types to functor substructures of these structures according to the rules:

(fr) if $(XY)_2$ is assigned B and X is assigned A then Y is assigned $A \to B$,

(fl) if $(XY)_1$ is assigned B and Y is assigned A then X is assigned $B \leftarrow A$.

$\mathrm{GF}(D)$ contains all assignments $v \mapsto A$ obtained in this way, for $v \in V$; the principal type of $\mathrm{GF}(D)$ is S. For instance, if D consists of structures (Joan works)$_2$ and (Joan (works hardly)$_2$)$_2$, then $\mathrm{GF}(D)$ contains the assignments:

$$\text{Joan} \mapsto x, y,\ \text{works} \mapsto z,\ x \to \mathrm{S},\ \text{hardly} \mapsto z \to (y \to \mathrm{S}).$$

A substitution is a mapping from variables to types; it is naturally extended to a mapping from types to types. If G is a CCG, and σ is a substitution, then $G\sigma$ is a (unique) CCG which assignments $v \mapsto A\sigma$, whenever $G : v \mapsto A$. We have $L^f(G) \subseteq L^f(G\sigma)$, for all G, σ. For $G, G' \in \Omega$, we write $G \subseteq G'$ if $I_G \subseteq I_{G'}$. The following lemma is crucial [20].

(L1) Let $D \subseteq F(V)$ be nonempty and finite. Then, for every $G \in \Omega$, $D \subseteq L^f(G)$ iff there exists σ such that $GF(D)\sigma \subseteq G$.

The following notions come from the logical theory of unification; we modify definitions from [20]. A substitution σ is called *a unifier* of $G \in \Omega$ if, for all $v \in V$ and types A, B such that $G : v \mapsto A$ and $G : v \mapsto B$, there holds $A\sigma = B\sigma$. It is called *a most general unifier* (m.g.u.) of G if it is a unifier of G and, for every unifier η of G, there exists a substitution γ such that $\eta = \sigma\gamma$. G is said to be *unifiable* if there exists a unifier of G. The standard algorithm of unification can be used to decide whether a given CCG G is unifiable and, if so, to produce an m.g.u. of G (it is unique up to variants). For rigid CCG's, and any nonempty, finite $D \subseteq F(V)$, there holds the following theorem [20].

(T2) There exists a rigid $G \in \Omega$ such that $D \subseteq L^f(G)$ iff $\mathrm{GF}(D)$ is unifiable. If σ is an m.g.u. of $\mathrm{GF}(D)$, then $\mathrm{GF}(D)\sigma$ is a rigid CCG from Ω whose functor language is the smallest language $L(G)$ such that $G \in \Omega$ is rigid and $D \subseteq L(G)$.

GF(D) from the above example is unifiable; an m.g.u. σ is the substitution: y/x, $z/x \to$ S, hence GF(D)σ is given by:

$$\text{Joan} \mapsto x, \text{ works} \mapsto x \to \text{S}, \text{ hardly} \mapsto (x \to \text{S}) \to (x \to \text{S}),$$

which agrees with our earlier analysis of these sentences.

We define RG(D) =GF(D)σ, if GF(D) is unifiable and σ is an m.g.u. of GF(D). Kanazawa [38, 39] proves that RG($\{s_0, \ldots, s_n\}$) is a conservative learning function for the class $\mathcal{G}_r$, of all rigid grammars from Ω. It is true if one identifies 'isomorphic' grammars. It is easier to show that, for $\chi(D)$ =RG(D), the learning function φ, defined by (l0)-(l2), learns $\mathcal{G}_r$ (now, 'isomorphic' grammars need not be identified); this function is conservative. By (T1), one needs (MIN) and finite elasticity of $L(\mathcal{G}_r)$. (MIN) holds, by (T2). To prove finite elasticity one may use Kanazawa's lemma on chains of rigid languages:

(L2) There exists no infinite sequence (G_i) of rigid CCG's from Ω such that $L^f(G_i) \subset L^f(G_{i+1})$, for all $i \in \omega$.

We omit a rather involved, combinatorial proof of (L2). Let $\mathcal{L}$ be the class $L(\mathcal{G}_r)$ enriched with the total language $F(V)$. Using (L2) and (T2), we prove that $\mathcal{L}$ is a closure system. Let $\{L_i\}_{i \in I}$ be a family of languages from $\mathcal{L}$, and let L be the meet of this family. If $I = \emptyset$, then $L = F(V)$, hence $L \in \mathcal{L}$. So, we may assume $I \neq \emptyset$ and $L \neq F(V)$; further, we may also assume $L_i \in L(\mathcal{G}_r)$, for all $i \in I$. We consider two cases. (1) L is finite. By (T2), RG(L) exists, and the functor language of RG(L) is the smallest language $L' \in L(\mathcal{G}_r)$ such that $L \subseteq L'$. Consequently, this functor language contains L and is contained in all L_i, for $i \in I$, hence it equals L. Thus, $L \in \mathcal{L}$. (2) L is infinite. Let (s_i) be a text for L. We define $D_n = \{s_0, \ldots, s_n\}$. By (T2), RG($D_n$) exists, for all $n \in \omega$. Let $L(n)$ denote the functor language of RG(D_n). Clearly, $L(n) \subseteq L(n+1)$ and $L(n) \subseteq L_i$, for all $n \in \omega$, $i \in I$. By (L2), there exists $n \in \omega$ such that $L(m) = L(n)$, for all $m \geq n$, and consequently, $L = L(n)$. Again, $L \in \mathcal{L}$.

So, $\mathcal{L}$ is a closure system which admits no infinite ascending chains. By the fact on closure systems, proven above, $\mathcal{L}$ has finite elasticity, and consequently, $L^f(\mathcal{G}_r)$ has finite elasticity (this property is preserved by subclasses). This yields the learnability of $\mathcal{G}_r$.

A CCG G is said to be $k-$valued if $\{A : (v, A) \in G\}$ is of cardinality at most k, for any $v \in V$. By $\mathcal{G}_k$ we denote the class of all $k-$valued grammars in Ω. Again, (T1) yields the learnability of $\mathcal{G}_k$.

To show the finite elasticity of $L^f(\mathcal{G}_k)$ Kanazawa uses the following lemma.

(L3) Let $R \subseteq E_1 \times E_2$ be a finite valued relation, that means, for any $s \in E_1$, there exist at most finitely many $t \in E_2$ such that $(s,t) \in R$. Let $\mathcal{L}$ be a family of languages on E_2 which has finite elasticity. Then, the family $\{R^{-1}[L] : L \in \mathcal{L}\}$ has finite elasticity.

The proof of (L3) is an application of the König lemma: every infinite, finitely branching tree has an infinite branch.

Each atom from V is associated with k copies $v^1, \ldots, v^k$. Let W denote the set of all copies of atoms from V. A mapping $f : F(W) \mapsto F(V)$ is defined by: (i) $f(v^i) = v$, (ii) $f((XY)_i) = (f(X)f(Y))_i$. We define a finite valued relation $R \subseteq F(V) \times F(W)$: $(X,Y) \in R$ iff $X = f(Y)$. For any grammar $G \in \mathcal{G}_k$, we define a rigid grammar G' with lexicon W in the following way: if $A_1, \ldots, A_i$ are all types assigned to v by G, then G' assigns A_j to v^j, for $j = 1, \ldots, i$. Clearly, $L^f(G) = R^{-1}[L^f(G')]$ and $L^f(\mathcal{G}_k) = R^{-1}[L^f(\mathcal{G}_r)]$. So, the latter class has finite elasticity.

To prove (MIN) one uses a modification of the construction of $\mathrm{RG}(D)$. Let $\mathrm{GF}_k(D)$ denote the set of all grammars G with lexicon W, satisfying the conditions: for all $v \in V$ and types A, $GF(D) : v \mapsto A$ iff, for some i, $G : v^i \mapsto A$, (ii) if $i \neq j$ then G assigns no common type to both v^i and v^j. Clearly, $\mathrm{GF}_k(D)$ is a finite set, efectively constructed from $\mathrm{GF}(D)$. Let $\mathrm{GF}^k(D)$ be defined as follows: (i) construct all grammars $G\sigma$ such that $G \in \mathrm{GF}_k(D)$, σ is an m.g.u. of G (if exists), (ii) return to lexicon V, by assigning to v all types $G\sigma$ assigns to copies of v. $\mathrm{GF}^k(D)$ is an effectively constructed set of $k-$valued grammars, fulfilling an analogue of (L1): for any $k-$valued grammar G, $D \subseteq L^f(G)$ iff there exist $G' \in \mathrm{GF}^k(D)$ and substitution σ such that $G'\sigma \subseteq G$. Consequently, among languages $L^f(G)$, $G \in \mathrm{GF}^k(D)$, there appear all minimal languages L such that $L = L^f(G)$, for some $k-$valued grammar G with $D \subseteq L^f(G)$. Since the inclusion problem $L^f(G) \subseteq L^f(G')$ is decidable, then we can effectively find a grammar $G \in \mathrm{GF}^k(D)$ such that $L^f(G)$ is minimal among all languages of grammars from $\mathrm{GF}^k(D)$, and consequently, among all languages of $k-$valued grammars G such that $D \subseteq L^f(G)$. Accordingly, a function χ satisfying (MIN) exists.

Kanazawa also considers learnability from strings. Now, the set E consists of all nonempty strings on V, and $L(G)$ is the string language of G. For $X \in F(V)$, by $s(X)$ we denote the string on V resulting from X after one has dropped all structure markers. The relation R between strings and structures, given by: $(a,X) \in R$ iff $a = s(X)$, is finite valued, and consequently, if $L^f(\mathcal{G})$ has finite elasticity, then $L(\mathcal{G})$ has finite elasticity, by (L3). So, both $L(\mathcal{G}_r)$ and $L(\mathcal{G}_k)$ have finite elasticity, hence $\mathcal{G}_r$ and $\mathcal{G}_k$ are learnable from strings, by a general theorem of formal learning theory, mentioned above.

The corresponding learning functions need not be conservative. Kanazawa defines nonconservative learning functions for these classes which use the above constructions performed on all possible structures admissible for a given string.

We have discussed learnability from positive data. As observed by Gold [31], if one admits texts consisting of all expressions from E and informing which expressions belong and which expressions do not belong to the language, then every recursively enumerable class of grammars is learnable. The learning function simply picks the first grammar G_n (in the fixed recursive sequence of all grammars from the class) which is compatible with the sample.

In [20, 56], negative data have been handled in a different way. Samples consist of positive data only, while some negative information is employed to restrict the search space of admissible grammars. In [56], there are considered classes $\mathcal{B}$, of grammars, fulfilling the following conditions:

(N1) if $G \subseteq G'$ and $G' \in \mathcal{B}$, then $G \in \mathcal{B}$,

(N2) if $G\sigma \in \mathcal{B}$, then $G \in \mathcal{B}$.

The order of type A is defined by:

$$o(A) = 0, \text{ for primitive } A,\ o(A \to B) = o(B \leftarrow A) = \max(o(B), o(A) + 1).$$

Thus, $o(p \to q) = 1$ and $o((p \to q) \to r) = 2$. The order of G is the maximal order of types in G; we denote it by $o(G)$. Always $o(G) \leq o(G\sigma)$, hence the class of all CCG's of order at most n satisfies (N1) and (N2). Another example is the following. We admit Pr contains some other constant types besides S, say PN, N etc. Let $X_1, \ldots, X_n$ be structures in $F(V)$, and let $A_1, \ldots, A_n$ be variable-free types. Then, the class of all CCG's which do not assign A_i to X_i, for $i = 1, \ldots, n$, satisfies (N1), (N2).

A set S, of substitutions, is said to be *hereditary*, if: $\sigma\eta \in S$ entails $\sigma \in S$. If $\mathcal{B}$ satisfies (N2), then, for any G, the set of all σ such that $G\sigma \in \mathcal{B}$ is hereditary. If $\mathcal{B}$ is recursive, then the latter set is recursive. We say that G is unifiable in S if there exists a unifier of G which belongs to S. If S is hereditary, then G is unifiable in S iff the m.g.u. of G exists and belongs to S.

Using these facts, all results discussed above for classes $\mathcal{G}_r$ and $\mathcal{G}_k$ can also be obtained for these classes restricted to grammars from a recursive class $\mathcal{B}$, fulfilling (N1), (N2). Many results of that kind can be found in [56, 18, 26, 27].

Kanazawa considers similar problems for type grammars based on logics stronger than AB; Moortgat [60] proposes an extension to grammars based on multimodal logics. Recently, Foret has shown that string languages of rigid grammars based on Associative Lambek Calculus (see section 3) admit a limit point, hence this class of grammars is not learnable from strings. Rigidness is not a natural constraint for Lambek grammars, since the Lambek calculus transforms each type to infinitely many types. Consequently, no Lambek grammar is esentially rigid (or $k-$valued), and grammars of that kind seem to require some different notion of learnability. On the other hand, methods discussed above can be adapted to various grammar formalisms, different from CCG but interpretable in it, as e.g. finite state automata, CF-grammars in (generalized) Greibach normal form, planar dependency grammars, and others.

Type logics

The basic logic AB can be extended in various ways. Lambek [49] has introduced Syntactic Calculus, now called Associative Lambek Calculus (AL). Types are formed out of primitive types by $\otimes, \rightarrow, \leftarrow$. The axioms are (Id) and:

$$\text{(ASS)}\ (A \otimes B) \otimes C \vdash A \otimes (B \otimes C),\ A \otimes (B \otimes C) \vdash (A \otimes B) \otimes C,$$

and inference rules are:

$$\text{(RRES)}\ A \otimes B \vdash C \text{ iff } B \vdash A \rightarrow C,$$

$$\text{(LRES)}\ A \otimes B \vdash C \text{ iff } A \vdash C \leftarrow B,$$

$$\text{(CUT1) if } A \vdash B \text{ and } B \vdash C \text{ then } A \vdash C.$$

Theorems of AL (in this format) are simple sequents $A \vdash B$, interpreted as: every expression of type A is also of type B. By (Id), (RRES) and (LRES), one gets:

$$A \otimes (A \rightarrow B) \vdash B,\ (B \leftarrow A) \otimes A \vdash B,$$

which correspond to the reduction rules of AB. AL provides many theorems not derivable in AB, as e.g.:

(1) $A \vdash B \leftarrow (A \rightarrow B)$, $A \vdash (B \leftarrow A) \rightarrow B$,

(2) $(A \rightarrow B) \otimes (B \rightarrow C) \vdash A \rightarrow C$, $(C \leftarrow B) \otimes (B \leftarrow A) \vdash C \leftarrow A$,

(3) $(A \to B) \leftarrow C \vdash A \to (B \leftarrow C)$, $A \to (B \leftarrow C) \vdash (A \to B) \leftarrow C$,

(4) $A \vdash B \to (B \otimes A)$, $A \vdash (A \otimes B) \leftarrow B$,

called expansion, composition, associativity and lifting, respectively.

According to (1), type PN (proper noun) can be expanded to NP (noun phrase) equal to S← (PN→S). As a consequence, noun phrase conjunctions of type (NP→NP)←NP can be used in contexts like 'Joan and some teacher', analysed as compound noun phrases. According to (2), the sentence negation 'not' of type S←S can be composed with a noun phrase to produce a negated noun phrase, e.g. 'not every teacher'. An application of (3) was observed by Lambek. Let NP' equal to (S←PN)→S be the type of noun phrases on the object position, as in 'likes every teacher'. (PN→S)←PN is the type of transitive verb phrases. If we assign NP' to 'him', then 'Joan likes him' gives rise to the sequent:

$$\text{NP } (\text{PN}\to\text{S})\leftarrow\text{PN } (\text{S}\leftarrow\text{PN})\to\text{S} \vdash \text{S},$$

which is derivable in AL but not in AB (in AL, use (3) to the middle type). The sequence $A_1 \dots A_n$ in the antecedent is interpreted as $A_1 \otimes \dots \otimes A_n$.

Type transformations provided by AL are justified by an algebra of types, which is assumed by Lambek but not by Ajdukiewicz and Bar-Hillel. In CCG, types of expressions are completely determined by the initial type assignment of a categorial grammar. If 'Joan' is assigned type PN, then PN is the only type of 'Joan' in all syntactic analyses. According to Lambek, 'Joan' has type PN, but also type NP, NP' and infinitely many other types A such that PN$\vdash A$ is derivable in AL. Types are interpreted as sets of nonempty strings on a lexicon V (languages on V). The logical constants $\otimes, \to, \leftarrow$ are interpreted by the following operations on languages:

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

(6) $L_1 \backslash L_2 = \{a \in V^+ : (\forall b \in L_1) ba \in L_2\}$,

(7) $L_1 / L_2 = \{a \in V^+ : (\forall b \in L_2) ab \in L_1\}$.

A language model is defined as a pair (V, μ) such that V is a nonempty finite set, and μ is a mapping from Pr into $P(V^+)$. One extends μ to a mapping from Tp (the set of types) into $P(V^+)$, by setting:

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

A sequent $A \vdash B$ is said to be true in this model, if $\mu(A) \subseteq \mu(B)$. It is easy to show that sequents provable in AL are true in all language models. Pentus

[66] proves the completeness: every sequent true in all language models is provable in AL (for product-free sequents, it has been shown in [7]).

Since AL is sound and complete with respect to language models, grammars based on AL cannot yield incorrect types of expressions provided that the initial type assignment is correct. Lambek [50] discusses the following example. Let us assign type PN←PN to adjectives. Then, AL yields type S to pseudo-sentence 'Joan likes poor him', if 'Joan', 'likes' and 'him' are typed as above. Lambek concludes that AL is too strong, and proposes to replace AL by its nonassociative variant without (ASS). Nonassociative Lambek Calculus (NL) is interesting for many reasons, but Lambek's conclusion seems to be misleading. Type PN←PN of 'poor' and type (PN→S)←PN of 'likes' are correct, and the sequent:

$$(\mathrm{PN}\rightarrow\mathrm{S})\leftarrow\mathrm{PN}\ \ \mathrm{PN}\leftarrow\mathrm{PN}\ \ (\mathrm{S}\leftarrow\mathrm{PN})\rightarrow\mathrm{S}\vdash\mathrm{PN}\rightarrow\mathrm{S}$$

is provable in AL (hence true in language models), but 'likes poor him' is not a syntactically correct expression of type PN→S (intransitive verb phrase). The conslusion must be: NP' is not a correct type of 'him'. On the other hand, it is difficult to find totally correct types of lexical atoms in natural language. Dropping associativity makes this task easier: now, 'likes poor' is not a syntactically correct expression of type (PN→S)←PN, hence NP' can correctly be assigned to 'him'. So, Lambek is right in his final solution. Language models of NL interpret types as phrase languages on V.

Type tranformations in AL correspond to semantic transformations definable in the lambda calculus. Primitive semantic types are t (truth value), e (entity), and possibly others. $a\rightarrow b$ is the type of functions which assign objects of type b to objects of type a. Syntactic types of AL correspond to semantic types: PN corresponds to e, S to t, and $A\rightarrow B$, $B\leftarrow A$ correspond to $a\rightarrow b$, where a corresponds to A and b to B. For instance, the semantic type of intransitive verb phrases is $e\rightarrow t$, of functions which assign truth values to entities; they can be identified with sets of entities. The semantic type corresponding to NP is $(e\rightarrow t)\rightarrow t$, of functions which assign truth values to objects of type $e\rightarrow t$; they can be identified with families of sets of entities. (The noun phrase 'every student' denotes the family of all sets W such that every student belongs to W.)

We say that a type transformation $a_1\ldots a_n\Rightarrow a$ is lambda definable, if there exists a lambda term M of type a with free variables x_i of type a_i, for $i=1,\ldots,n$. Let us consider composition (1) for $A=$PN, $B=$S. The corresponding type transformation $e\Rightarrow(e\rightarrow t)\rightarrow t$ is definable by the term:

$$\lambda x^{e\rightarrow t}.x^{e\rightarrow t}y^{e},$$

where the superscripts indicate types of variables. For any given value u of y^e, this term denotes a mapping F_u such that $F_u(f) = f(u)$, for any function f of type $e \to t$. Equivalently, F_u is the family of all sets W such that $u \in W$. Less formally, expansion corresponds to a semantic transformation which sends each entity u to the family of all (monadic) predicates true on u (or: properties of u). In semantical terms, the phrase 'John and every student' can be interpreted as follows. First, we expand type e of 'John' to type $(e \to t) \to t$ of noun phrases. Now, 'John' and 'every student' are of the same type. The entity denoted by 'John' is transformed by the lambda term to the family of all properties of this entity. The conjunction 'and' is interpreted as the Boolean meet on families of sets. Consequently, 'John and every student' denotes the family of all sets W such that John and all students belong to W, as expected.

Interconnections between type logics and semantic transformations have been studied by van Benthem [76, 78], Moortgat [58, 59] and their collaborators. van Benthem proves a kind of lambda completeness of the commutative AL: sequents provable in this system are precisely those which are definable by lambda terms, fulfilling three constraints: (i) every subterm has a free variable, (ii) every variable occurs at most once in each subterm, (iii) each lambda binds a variable in its scope. This refines the well-known Curry-Howard isomorphism between intuitionistic proofs and lambda calculus. Along these lines, derivations in type logics have a sense from the standpoint of Montague semantics and may serve as a basis for 'semantic parsing'.

Type logics as substructural logics

Lambek [49] presents AL in the form of a Gentzen-style sequent system. Sequents are of the form $\Gamma \vdash A$, where $\Gamma \in \mathrm{Tp}^+$ and $A \in \mathrm{Tp}$. The axioms are (Id) (it suffices to restrict them to primitive types A), and the inference rules are:

$$(\otimes\mathrm{L})\ \frac{\Gamma A B \Gamma' \vdash C}{\Gamma(A \otimes B)\Gamma' \vdash C},\ (\otimes\mathrm{R})\ \frac{\Gamma \vdash A;\ \Delta \vdash B}{\Gamma\Delta \vdash A \otimes B},$$

$$(\to\mathrm{L})\ \frac{\Gamma B \Gamma' \vdash C;\ \Delta \vdash A}{\Gamma\Delta(A \to B)\Gamma' \vdash C},\ (\to\mathrm{R})\ \frac{A\Gamma \vdash B}{\Gamma \vdash A \to B},$$

$$(\leftarrow\mathrm{L})\ \frac{\Gamma B \Gamma' \vdash C;\ \Delta \vdash A}{\Gamma(B \leftarrow A)\Delta\Gamma' \vdash C},\ (\leftarrow\mathrm{R})\ \frac{\Gamma A \vdash B}{\Gamma \vdash B \leftarrow A},$$

where $\Gamma \neq \epsilon$ in $(\to\mathrm{R})$ and $(\leftarrow\mathrm{R})$. Dropping the latter constraint yields a stronger system AL0 in which provable sequents may have empty antecedents. AL0 is not conservative over AL: $\vdash p \to p$ is provable in AL0,

by (Id) and (→R), hence $(p \to p) \to p \vdash p$ is provable in AL0, by (Id) and (→L), but the latter is not provable in AL. AL0 is complete with respect to language models in which languages are subsets of $V^\star$ (the set of all strings over V).

Provability in AL and AL0 is decidable, since the above axiomatizations yield cut-free sequent systems in which the complexity of the conclusion of each rule is greater than the complexity of any premise of this rule. As shown by Lambek [49], (CUT) is admissible in AL, and the same holds for AL0. Using (CUT), it is easy to prove that the sequent system of AL is equivalent to the 'algebraic' system given at the beginning of this section. As concerns computational complexity, AL and AL0 are NP-TIME decidable; recently, Pentus [67] shows NP-completeness.

Nonassociative Lambek Calculus (NL) deals with sequents $X \vdash A$ such that X is a type structure and A is a type. *Type structures* are defined as follows: (i) all types are (atomic) type structures, (ii) if X, Y are type structures, then $(X \circ Y)$ is a type structure. So, up to notation, type structures amount to phrase structures on Tp in the sense of section 2. The axioms are (Id), and the inference rules are:

$$(\otimes\text{L})\ \frac{X[A \circ B] \vdash C}{X[A \otimes B] \vdash C},\ (\otimes\text{R})\ \frac{X \vdash A;\ Y \vdash B}{X \circ Y \vdash A \otimes B},$$

$$(\to\text{L})\ \frac{X[B] \vdash C;\ Y \vdash A}{X[Y \circ (A \to B)] \vdash C},\ (\to\text{R})\ \frac{A \circ X \vdash B}{X \vdash A \to B},$$

$$(\leftarrow\text{L})\ \frac{X[B] \vdash C;\ Y \vdash A}{X[(B \leftarrow A) \circ Y] \vdash C},\ (\leftarrow\text{R})\ \frac{X \circ A \vdash B}{X \vdash B \leftarrow A}.$$

As usual, $X[Y]$ denotes structure X with a distinguished substructure Y, and in this context $X[Z]$ denotes the result of replacing Y with Z in X. NL0 admits the empty type structure 0, satisfying $0 \circ X = X$ and $X \circ 0 = X$, for any type structure X. The cut rule:

$$(\text{NCUT})\ \frac{X[A] \vdash B;\ Y \vdash A}{X[Y] \vdash B}$$

is admissible in NL and NL0 [50]. As shown in [33], NL is P-TIME decidable. A different proof of this fact will be sketched below.

NL and NL0 are genuine substructural logics. In the Gentzen format, they employ logical rules, that means, rules for logical constants only. They employ no structural rules, manipulating type structures. AL (AL0) can be obtained from NL (NL0) by affixing the structural rule of associativity:

$$(\text{RASS})\ X[(Y_1 \circ Y_2) \circ Y_3] \vdash A \text{ iff } X[Y_1 \circ (Y_2 \circ Y_3)] \vdash A;$$

as a result of (RASS), type structures can be identified with strings of types. Stronger systems arise by adding other structural rules, as e.g.

$$\text{(EXC)}\ \frac{\Gamma AB\Gamma' \vdash C}{\Gamma BA\Gamma' \vdash C},$$

$$\text{(WEA)}\ \frac{\Gamma\Gamma' \vdash B}{\Gamma A\Gamma' \vdash B},$$

$$\text{(CON)}\ \frac{\Gamma \vdash B}{\Delta \vdash B};$$

in (CON), Γ contains at least two occurrences of A, and Δ drops one of them. As a result of (EXC) (the rule of exchange), antecedents of sequents can be identified with multisets of types. For systems with (EXC), rules (WEA) (weakening) and (CON) (contraction) can be formulated in a more standard way:

$$\text{(WEA)}\ \frac{\Gamma \vdash B}{\Gamma A \vdash B},\ \text{(CON)}\ \frac{\Gamma AA \vdash B}{\Gamma A \vdash B}.$$

In the presence of (EXC), $\otimes$ is commutative, and $A \to B$ is equivalent to $B \leftarrow A$ (so, $\leftarrow$ can be abandoned). The product-free fragment of AL0 plus (EXC) amounts to the logic BCI (the system of types of combinators formed out of combinators B, C and I). A Hilbert-style system of BCI employs the axioms:

(B) $(B \to C) \to ((A \to B) \to (A \to C))$,

(C) $(A \to (B \to C)) \to (B \to (A \to C))$,

(I) $A \to A$,

and the only rule MP: from $A \to B$ and A infer B. In the tradition of substructural logics, $\otimes$ is called *fusion*. AL0 plus (EXC) amounts to BCI with fusion. Analogously, the product-free AL0 plus (EXC), (WEA) (resp. (CON)) amounts to BCK (resp. BCIW), and the system enriched with (EXC), (WEA), (CON) amounts to the logic of positive implication. The rule (CUT) is admissible for all systems, mentioned above.

Zielonka [84] proves that the product-free fragment of AL0 cannot be axiomatized by any finite number of axiom schemes with MP and its dual (from $B \leftarrow A$ and A infer B) as the only rules; much earlier [83], this author obtained similar results for AL. It holds for AL plus (EXC), but it remains open for NL0. Notice that, for systems like AL, (AL) plus (EXC), one takes axioms of the form $\Gamma \vdash A$ (closed under substitution), $\Gamma \neq \epsilon$, and (CUT) as the only rule.

Systems with (EXC) may be regarded as logics of semantic types. For instance, the product-free AL plus (EXC) has been studied by van Benthem [76] as a basic logic of semantics.

It is reasonable to consider further extensions of these systems. One direction is to admit lattice operations $\wedge$ and $\vee$. For associative systems, the rules for $\wedge$ are:

$$(\wedge\mathrm{L1})\ \frac{\Gamma A\Gamma' \vdash C}{\Gamma(A \wedge B)\Gamma' \vdash C},\ (\wedge\mathrm{L2})\ \frac{\Gamma B\Gamma' \vdash C}{\Gamma(A \wedge B)\Gamma' \vdash C},$$

$$(\wedge\mathrm{R})\ \frac{\Gamma \vdash A;\ \Gamma \vdash B}{\Gamma \vdash A \wedge B},$$

and the rules for $\vee$ are:

$$(\vee\mathrm{L})\ \frac{\Gamma A\Gamma' \vdash C;\ \Gamma B\Gamma' \vdash C}{\Gamma(A \vee B)\Gamma' \vdash C},$$

$$(\vee\mathrm{R1})\ \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B},\ (\vee\mathrm{R2})\ \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}.$$

Again, (CUT) is admissible, and the resulting systems are decidable. Since AL with $\wedge, \vee$ is a conservative extension of AL, then the decision problem for the former is NP-complete, by the Pentus result for AL. The precise computational complexity of NL with $\wedge, \vee$ is not known.

Systems with $\wedge, \vee$ lack some important property, characteristic of systems like AL, AL0, NL, NL0 (also with (EXC)). Proof trees in the latter systems show no 'material' dependence of different branches. Primitive types appearing in the conclusion of ($\otimes$R) come from premises independently of each other, and similarly for other rules. As a consequence, every proof tree can be transformed into another proof tree in which each axiom $p \vdash p$ appears precisely once (we use different copies of p for different occurrences of $p \vdash p$). In sequents appearing in the modified proof tree, each primitive type has 0 or 2 occurrences, and the initial proof tree can be obtained from the latter by substitution (one substitutes p for every copy of p). Accordingly, every provable sequent is a substitution of a provable sequent in which each primitive type occurs twice. This property is called Multiplicativity. In the tradition of linear logic [32], constants $\otimes, \rightarrow, \leftarrow$ are said to be *multiplicative*, while $\wedge, \vee$ are *additive*. Actually, in rules ($\wedge$R) and ($\vee$L) the antecedent string Γ must be the same in both premises, hence the two branches of the proof tree are 'materially' dependent.

Multiplicative systems admit a strong form of Interpolation. For $p \in$Pr, let $|A|_p$ denote the number of occurrences of p in A. $|\Gamma|_p$ is defined in a

similar way. $|\Gamma|$ denotes the total number of occurrences of primitive types in Γ, and similarly for $|A|$. We state an interpolation lemma, proven by Roorda [70].

(INT) Let $\Gamma\Delta\Gamma' \vdash B$ be provable in AL with $\Delta \neq \epsilon$. Then, there exists type A (an interpolant of Δ), satisfying: (i) $\Delta \vdash A$ is provable in AL, (ii) $\Gamma A\Gamma' \vdash B$ is provable in AL, (iii) for any $p \in$Pr, $|A|_p$ is not greater than the minimum of $|\Delta|_p$, $|\Gamma\Gamma' B|_p$.

The proof proceeds by induction on proofs in AL. Let us demonstrate one case. Assume $\Gamma\Gamma' \vdash A \otimes B$ is derivable from $\Gamma \vdash A$ and $\Gamma' \vdash B$, by ($\otimes$R). Let $\Gamma = \Gamma_1\Gamma_2$, $\Gamma' = \Gamma'_1\Gamma'_2$, $\Delta = \Gamma_2\Gamma'_1$ with Γ_2 and Γ'_1 being nonempty. We find an interpolant C of Δ. By the induction hypothesis, there exists an interpolant C_1 of Γ_2 in $\Gamma \vdash A$, and there exists an interpolant C_2 of Γ'_1 in $\Gamma' \vdash B$. Set $C = C_1 \otimes C_2$. $\Delta \vdash C$ is provable, by ($\otimes$R). Since $\Gamma_1 C_1 \vdash A$ and $C_2\Gamma'_2 \vdash B$ are provable, then $\Gamma_1 C\Gamma'_2 \vdash A \otimes B$ is provable, by ($\otimes$R) and ($\otimes$L). We also have:

$$|C_1|_p \leq |\Gamma_2|_p,\ |C_2|_p \leq |\Gamma'_1|_p,$$

which yields $|C|_p \leq |\Delta|_p$ (add both inequalities). In a similar way, we show $|C|_p \leq |\Gamma_1\Gamma'_2(A \otimes B)|_p$.

Using multiplicativity and interpolation, Pentus [65] proves that type grammars based on AL generate context-free languages (the converse statement that all context-free languages are generated by these grammars follows from the Gaifman theorem and the fact that AL is equivalent to AB for sequents $A_1 \ldots A_n \vdash p$ such that $A_1, \ldots, A_n$ are types of order at most 1 and p is primitive). By multiplicativity, for every sequent $\Gamma \vdash A$, provable in AL, there exists a provable sequent $\Gamma' \vdash A'$ in which each primitive type occurs exactly twice, and $\Gamma \vdash A = (\Gamma' \vdash A')\sigma$, for some substitution σ. Let $\Gamma = A_1 \ldots A_n$, $\Gamma' = A'_1 \ldots A'_n$. For each $i = 1, \ldots, n$, one finds an interpolant B_i of A'_i in $\Gamma' \vdash A'$ and an interpolant B of Γ'. Then, $B_1 \ldots B_n \vdash B$ is provable in AL, each primitive type occurs twice in this sequent and at most once in every type of this sequent; such sequents are said to be *thin*. By a combinatorial argument, Pentus proves the following binary reduction lemma for thin sequents: if $B_1 \ldots B_n \vdash B$ is thin, $|B_i| \leq m$, $|B| \leq m$, then there exist $1 \leq j < n$ and type C such that $|C| \leq m$ and C is an interpolant of $B_j B_{j+1}$ in $B_1 \ldots B_n \vdash B$. Applying σ, we prove the same binary reduction lemma for the initial sequent $\Gamma \vdash A$. Let G be a grammar based on AL, and let m be the maximal $|A|$, for types A appearing in G. All types in G employ only a finite collection of primitive types. By the binary reduction lemma,

every sequent $A_1 \dots A_n \vdash A$ provable in AL and such that all types appear in G can be derived by (CUT) from provable sequents $C_1C_2 \vdash C_3$, $C_1 \vdash C_2$ such that $|C_i| \leq m$. By the decidability of AL, one can effectively determine all sequents of this form, and they yield production rules of a CF-grammar equivalent to the initial type grammar.

This proof cannot be adjusted to systems with $\wedge, \vee$. Grammars based on AL, NL with $\wedge$ generate languages which are not context-free [37], which follows from the fact that the intersection of two context-free languages need not be context-free.

For NL, NL0, the formulation of (INT) is the following. Let $X[Y] \vdash B$ be provable, $Y \neq 0$. Then, there exists type A, satisfying: (i) $Y \vdash A$ is provable, (ii) $X[A] \vdash B$ is provable, (iii) for any $p \in$Pr, $|A|_p$ is not greater than the minimum of $|Y|_p$, $|X[q]B|_p$, where $q \neq p$.

Interestingly, nonassociative systems admit another form of interpolation, not admissible in associative ones. Let T be a finite set of types, closed under subtypes. $T-$sequents are sequents using types from T only.

(NINT) Let $X[Y] \vdash B$ be a provable $T-$sequent. Then, there exists $A \in T$ such that (i) and (ii) hold true.

(NINT) can easily be proven by induction on proofs in NL, NL0 (also with (EXC)); a similar result has been announced by Jäger (unpublished) As a consequence, grammars based on these systems yield context-free languages. Other proofs, based on a normalization procedure for nonassociative systems, has been given in [9, 40, 41].

Other type logics, e.g. multimodal systems, pregroup systems, will be discussed later on.

Models and grammars

Abstract algebraic frames of NL are *residuated groupoids*, i.e. structures of the form $(M, \leq, \cdot, \backslash, /)$ such that $(M, \leq)$ is a nonempty poset, and $\cdot, \backslash, /$ are binary operations on M, satisfying the condition of residuation:

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

for all $a, b, c \in M$. *A model* is a pair (M, μ) such that M is a residuated groupoid, and μ is a mapping from Pr into M, which is extended to all types as for the case of language models. It is also extended to type structures: $\mu(X \circ Y) = \mu(X)\mu(Y)$. A sequent $X \vdash A$ is said to be *true* in a model (M, μ), if $\mu(X) \leq \mu(A)$.

A standard argument, using a canonical model, yields the completeness of NL with respect to residuated groupoids. We can consider theories in NL, i.e. systems arising from NL by affixing a set Φ of new axioms; then, the rule (NCUT) is treated as a legal rule of the system. A system S enriched by a set Φ of new axioms will be denoted S(Φ). Caution: we do not assume new axioms are closed under substitution. The strong completeness also holds: the sequents provable in NL(Φ) are precisely those which are true in all models (M, μ) in which all sequents from Φ are true.

Frames for NL0 are residuated groupoids with unit 1, satisfying: $1a = a1 = a$, for all elements a. We set $\mu(0) = 1$. To construct a canonical model (from equivalence classes of types) it is expedient to add a propositional constant 1 to the language, with the following rules:

$$(1\mathrm{L})\ \frac{X[0] \vdash A}{X[1] \vdash A},\ (1\mathrm{R})\ 0 \vdash 1;$$

recall that $0 \circ Y = Y \circ 0 = Y$ in the metalanguage, hence $X[1 \circ Y] \vdash A$ can be infereed from $X[Y] \vdash A$. This system is denoted NL1. Cut elimination holds for NL1, and NL1 is conservative over NL0. The completeness of NL1 can be shown by the construction of a canonical model, and consequently, NL0 is also complete with respect to residuated groupoids with unit.

Residuated groupoids in which $\cdot$ is associative are called *residuated semigroups*. Residuated semigroups with unit are called *residuated monoids*. AL is strongly complete with respect to residuated semigroups, and AL1 is strongly complete with respect to residuated monoids (AL1 is NL1 plus (RASS)). Since AL1 is conservative over AL0, the latter system is complete with respect to residuated monoids. For systems with $\wedge, \vee$, the frames are residuated groupoids (semigroups) $(M, \leq, \cdot, \backslash, /)$ such that $(M, \leq)$ is a lattice; we call them lattice ordered (l.o.) residuated semigroups (monoids), but the term 'residuated lattice' is also used in literature. Completeness can be obtained, as above. Systems with (EXC) are complete with respect to commutative residuated semigroups (monoids, groupoids) in which $ab = ba$, for all elements a, b.

More special frames are powerset frames. Let $(M, \cdot)$ be a groupoid. For $L_1, L_2 \subseteq M$, one defines operations $\cdot, \backslash, /$, by (5), (6), (7) (replace V^+ by M). The structure $(P(M), \subseteq, \cdot, \backslash, /)$ is a residuated groupoid (*the powerset residuated groupoid* over $(M, \cdot)$). If $(M, \cdot)$ is a semigroup, then the powerset structure is a residuated semigroup (*the powerset residuated semigroup* over $(M, \cdot)$). If $(M, \cdot, 1)$ is a monoid (i.e. a semigroup with unit), then $\{1\}$ is the unit in the powerset structure, and similarly for groupoids. AL (resp. AL1)

is strongly complete with respect to powerset residuated semigroups (resp. monoids) [8].

The proof given in [8] seems to be one of the first applications of labelled deductive systems, extensively studied by Gabbay [29]. Systems of that kind admit expressions of the form $t : A$ such that A is a logical formula, and t is a label. Rules of the system perform some logical operations on formulas and algebraic operations on labels. They are intended to formalize a part of metalanguage for the object language; for instance, $t : A$ may express the relation: A is true in state (world) t. In [8], $t : A$ means: the element t of M belongs to $\mu(A)$. Labels are defined by recursion: (i) all types are atomic labels, (ii) if $t_1, \ldots, t_n$, $n \geq 1$, are atomic labels, then $t_1 \ldots t_n$ is a label, (iii) if t is a label, and A, B are types, then $(t, A, B, 1)$ and $(t, A, B, 2)$ are atomic labels. One says that label t *reduces* to label t', if t' arises from t by a finite number of replacements of $(s, A, B, 1)(s, A, B, 2)$ by s. The axioms are (LID) $A : A$, and the rules are ND rules of elimination and introduction of $\otimes, \rightarrow, \leftarrow$.

(L$\otimes$E) from $t : A \otimes B$ infer both $(t, A, B, 1) : A$ and $(t, A, B, 2) : B$,

(L$\otimes$I) from $s : A$ and $t : B$ infer $st : A \otimes B$,

(L$\rightarrow$E) from $s : A$ and $t : A \rightarrow B$ infer $st : B$,

(L$\rightarrow$I) from $As : B$ infer $s : A \rightarrow B$,

and the dual rules (L$\leftarrow$E), (L$\leftarrow$I). We also need a reduction rule: from $t : A$ infer $t' : A$, whenever t reduces to t'. For any sequent $A_1 \ldots A_n \vdash B$, $n \geq 1$, we introduce a rule: from $t_i : A_i$, for $i = 1, \ldots, n$, infer $t_1 \ldots t_n : B$. The system LAL(Φ) admits the latter rule, for every sequent from Φ. To any label t one assigns a string of types $T(t)$ by the following recursion: (i) $T(A) = A$, (ii) $T(t_1 \ldots t_n) = T(t_1) \ldots T(t_n)$, (iii) $T((t, A, B, 1)) = A$, $T((t, A, B, 2)) = B$. One shows: $t : A$ is provable in LAL(Φ) iff $T(t) \vdash A$ is provable in AL(Φ). A label is said to be *irreducible* if it starts no proper reduction. We construct a canonical model. Let $(M, \cdot)$ be a semigroup in which M is the set of irreducible labels, and, for $s, t \in M$, $s \cdot t$ is the only irreducible label t' such that st reduces to t' (the uniqueness follows from the Church-Rosser property of reduction). We consider the powerset structure $P(M)$ and the canonical mapping μ, defined by: $\mu(p)$ equals the set of all $t \in M$ such that $t : p$ is provable in LAL(Φ). We prove: $\mu(A)$ equals the set of all $t \in M$ such that $t : A$ is provable in LAL(Φ). Now, if $A \vdash B$ is not provable in AL(Φ), then $A : B$ is not provable in LAL(Φ) (by $T(A) = A$), hence $A \in \mu(A)$ and $A \notin \mu(B)$. Consequently, $A \vdash B$ is not true in $(P(M), \mu)$. This yields the

strong completeness of AL with respect to powerset frames. In a similar way, one can prove a representation theorem: every residuated semigroup is embeddable into a powerset residuated semigroup.

In [46], a similar argument shows the strong completeness of NL with respect to powerset residuated groupoids and the corresponding representation theorem: every residuated groupoid is embeddable into a powerset residuated groupoid. If K, K' are classes of frames such that $K \subseteq K'$, a propositional system is (strongly) complete with respect to K, and there holds the representation theorem: every frame from K is embeddable into a frame from K', then this system is (strongly) complete with respect to K'. So, the completeness of a system with respect to concrete frames follows from its completeness with respect to abstract frames and the embeddability of abstract frames into concrete frames.

Language models discussed at the beginning of this section are special powerset structures in which the underlying semigroup is the free semigroup generated by V, i.e. the set V^+ with concatenation. Pentus [66] shows the completeness of AL with respect to language models (the proof is very sophisticated). Strong completeness fails. Take $p \vdash p \otimes p$ as a new axiom. If $\mu(p) \subseteq \mu(p)\mu(p)$, then $\mu(p) = \emptyset$, hence $\mu(p) \subseteq \mu(q)$ in all language models, satisfying this axiom, but $p \vdash q$ is not provable in AL($p \vdash p \otimes p$) (it is easy to find a powerset model over a semigroup in which the axiom is true, but $p \vdash q$ is false). Consequently, the representation theorem also fails: not every residuated semigroup is embeddable into the powerset residuated semigroup over a free semigroup.

These completeness results are easier for product-free fragments of type logics. Consider the free semigroup generated by Tp. In the powerset structure we define $\mu(p)$ as the set of all Γ such that $\Gamma \vdash p$ is provable in AL. One proves by induction: for any product-free type A, $\mu(A)$ is the set of all Γ such that $\Gamma \vdash A$ is provable in AL. Now, if $A \vdash B$ is not provable in AL, then $A \in \mu(A)$ and $A \notin \mu(B)$, hence $A \vdash B$ is not true in the model. Similar arguments can be provided for product-free fragments of AL0, NL, NL0 and their variants with 1, $\wedge$ [8].

For systems with product, this simpler method can be applied, if powerset models are replaced with more general cone models. In a poset $(M, \leq)$, a set $L \subseteq M$ is called *a (lower) cone* if $a \leq b$ and $b \in L$ entail $a \in M$. By $C(M)$ we denote the set of cones in M. Let $(M, \leq, \cdot)$ be a partially ordered semigroup (that means: we assume: $a \leq b$ entails $ca \leq cb$ and $ac \leq bc$). One defines operations $\backslash, /$ on $C(M)$ as above and the product as follows:

(5') $L_1 \cdot L_2 = \{x : (\exists y \in L_1, z \in L_2) x \leq yz\}$.

Then, $(C(M), \subseteq, \cdot, \backslash, /)$ is a residuated semigroup. There holds the representation theorem: every residuated semigroup M is embeddable into the residuated semigroup $C(M)$. This yields the strong completeness of AL with respect to cone models. Došen [23] gives a direct proof of this fact, using a canonical model construction with cone models instead of powerset models; the underlying structure is Tp with $\otimes$ as the operation and the provable $\vdash$ as $\leq$ (it is a preorder, but see below).

A refinement of this construction can be used to prove the subformula property for theories (since cut elimination fails, this property cannot be shown in a standard way). In [8], it was proven with the aid of the labelled deductive system, mentioned above. Here, we sketch a proof using canonical cone models [17]. First, observe that, in the construction of $C(M)$, the underlying structure $(M, \leq, \cdot)$ may be a preordered semigroup, that means, $\leq$ is reflexive and transitive, $\cdot$ is monotone in both arguments, and associativity holds modulo the equivalence generated by $\leq$: $a \sim b$ iff both $a \leq b$ and $b \leq a$. Let T be a set of types, closed under subtypes, which contains all types appearing in Φ. Take $M = T^+$ with concatenation. We write $\Gamma \vdash_T A$ if there exists a proof of $\Gamma \vdash A$ in AL(Φ) which consists of $T-$sequents only (see the end of subsection 3.1). We define a preorder $\leq$ on M as the reflexive and transitive closure of the following relation: $\Gamma\Delta\Gamma' \leq \Gamma A\Gamma'$ if $\Delta \vdash_T A$, $\Gamma AB\Gamma' \leq \Gamma(A \otimes B)\Gamma'$ if $(A \otimes B) \in T$. Then, $(M, \leq)$ with concatenation is a preordered semigroup. We consider the residuated semigroup $C(M)$. A mapping μ is defined as follows: $\mu(p) = \{\Gamma \in M : \Gamma \vdash_T p\}$. One proves $\mu(A) = \{\Gamma \in M : \Gamma \vdash_T A\}$, for all $A \in T$. Now, let $\Gamma \vdash A$ be provable in AL(Φ). Let T be the set of all subtypes of the types appearing in Φ and $\Gamma \vdash A$. Axioms from Φ are true in $(C(M), \mu)$, hence $\mu(\Gamma) \subseteq \mu(A)$, by soundness. Since $\Gamma \in \mu(\Gamma)$, then $\Gamma \in \mu(A)$, and consequently, $\Gamma \vdash_T A$. Thus, $\Gamma \vdash A$ has a proof in AL(Φ) in which all types are in T. A similar argument can be given for NL. For product-free fragments, powerset frames can be used instead of cone frames.

Now, we can prove that the consequence relation '$X \vdash A$ follows from Φ in NL' (that means, $X \vdash A$ is provable in NL(Φ)) is P-TIME decidable (Φ is a finite set of sequents $C \vdash D$). Let T be the smallest set of types, being closed under subtypes and containing all types appearing in $X \vdash A$ and Φ. We can construct T from the data in polynomial time, and the size of T is not greater than the number of primitive types and logical constants in the data. Recall that a $T-$sequent is a sequent whose all types are in T. A sequent is said to be *basic* if it is a $T-$sequent of the form $C_1 \circ C_2 \vdash C_3$ or $C_1 \vdash C_2$. The number of basic sequents equals $n^2 + n^3$, where n is the size of T. We can effectively find all basic sequents provable in NL(Φ).

First, we construct a set S^T of provable basic sequents. S_0 consists of all sequents from Φ, $T-$sequents (Id), and $T-$sequents of the form:

$$A \circ B \vdash A \otimes B,\ A \circ (A \to B) \vdash B,\ (B \leftarrow A) \circ A \vdash B.$$

Assume S_n has already been defined. S_{n+1} is S_n enriched with sequents resulting from the following rules:

(S1) if $A \circ B \vdash C$ is in S_n and $A \otimes B$ is in T, then $A \otimes B \vdash C$ is in S_{n+1},

(S2) if $A \circ B \vdash C$ is in S_n and $A \to C$ is in T, then $B \vdash A \to C$ is in S_{n+1},

(S3) if $A \circ B \vdash C$ is in S_n and $C \leftarrow B$ is in T, then $A \vdash C \leftarrow B$ is in S_{n+1},

(S4) if $A \circ B \vdash C$ and $A' \vdash A$ are in S_n, then $A' \circ B \vdash C$ is in S_{n+1},

(S5) if $A \circ B \vdash C$ and $B' \vdash B$ are in S_n, then $A \circ B' \vdash C$ is in S_{n+1},

(S6) if $X \vdash A$ and $A \vdash B$ are in S_n, then $X \vdash B$ is in S_{n+1}.

We define S^T as the join of this chain. Clearly, $S^T = S_k$, for the least k such that $S_k = S_{k+1}$, and this k is not greater than the number of basic sequents. The construction of S_{n+1} from S_n can be done in a time polynomial in the number of basic sequents. Consequently, S^T can be constructed in a time polynomial in the size of the data. Clearly, S^T is closed under (NCUT) restricted to basic sequents. Let $S(T)$ denote the system whose axioms are all sequents from S^T and the only rule is (NCUT). It is easy to show that all basic sequents derivable in $S(T)$ belong to S^T (use induction on derivations). All derivations in $S(T)$ are actually derivations in a CF-grammar whose production rules are (reversed) sequents from S^T. By the known CYK algorithm, checking derivability in CF-grammars, the derivability in $S(T)$ is P-TIME decidable. Now, it suffices to prove that NL(Φ) is equivalent to $S(T)$ in the scope of $T-$sequents. It is easy to show that all sequents derivable in $S(T)$ are derivable in NL(Φ). To show the converse, we first observe that the interpolation lemma (NINT) evidently holds for $S(T)$. By the subformula property, $T-$sequents provable in NL(Φ) have proofs which consist of $T-$sequents only. All axioms of NL(Φ), being $T-$sequents, are axioms of $S(T)$. It suffices to show that $S(T)$ is closed under rules of NL(Φ), restricted to $T-$sequents. It is obvious for (NCUT). Let us consider ($\otimes$L). Assume $X[A \circ B] \vdash C$ is provable in $S(T)$. By (NINT) for $S(T)$, there is $D \in T$ such that both $A \circ B \vdash D$ and $X[D] \vdash C$ are provable in $S(T)$. $A \circ B \vdash D$ is a basic sequent, so it belongs to S^T. Since $A \otimes B$ is in T, then $A \otimes B \vdash D$ belongs to S^T, by (S1). Then, $X[A \otimes B] \vdash C$ is provable

in $S(T)$, by (NCUT). Let us consider ($\otimes$R). Assume $X \vdash A$ and $Y \vdash B$ are provable in $S(T)$, and $A \otimes B$ is in T. Since $A \circ B \vdash A \otimes B$ is in S_0, then $X \circ Y \vdash A \otimes B$ is provable in $S(T)$, by (NCUT). The remaining rules are handled in a similar way.

The above construction yields a CF-grammar equivalent to the given type grammar based on NL(Φ). Accordingly, type grammars based on finite theories over NL generate context-free languages. These results strentgthen earlier results on P-TIME decidability of NL [33] and the equivalence of grammars based on NL with CF-grammars. Analogous facts can be proven for other nonassociative systems, e.g. NL1, NL with (EXC), multi-modal logics and GLC (see below) [17].

Nonlogical axioms can be of interest for linguistics for several reasons. For the case of AL, grammars based on finite theories generate all recursively enumerable languages. This fact has been proven in [7] for the product free fragment. With product, the proof is much easier: every rewriting system with rules of the form $pq \mapsto r$ and $p \mapsto qr$ can be simulated in AL enriched with axioms $p \otimes q \vdash r$ and $p \vdash q \otimes r$ (use the subformula property to show conservativity [17]). Thus, associative systems with nonlogical axioms surpass the context-free world. Nonlogical axioms can express subtyping; for instance, restrictive adjectives are a subcategory of adjectives. Further, it is known that not every linguistically sound transformation can be deduced in AL. For instance, 'and' of type (S$\rightarrow$S)$\leftarrow$S can also act as a verb phrase conjunction of type (VP$\rightarrow$VP)$\leftarrow$VP, e.g. in 'Joan sings and dances'. One cannot transform the former type to the latter in AL (it is possible in AL with (EXC) and (CON)). So, the corresponding sequent can be added to AL or NL as a nonlogical axiom. By enriching NL with finitely many new axioms, derivable in AL or not, we can improve its expressibility without lacking the nice computational simplicity.

In powerset frames and cone frames, $\wedge$ can be interpreted as the intersection of sets: $\mu(A \wedge B) = \mu(A) \cap \mu(B)$. In general, $\vee$ should not be modelled as the union of sets; this would entail the distributivity of $\wedge$ under $\vee$, which could not be derived in the systems. Adequate frames consist of ideals, i.e. cones closed under $\vee$. The join of two ideals is the smallest ideal containing them [69].

An interesting problem is to extend type logics to be able to express negative facts. We briefly discuss some attempts from [12].

A De Morgan negation $\neg$ satisfies Double Negation $\neg\neg a = a$ and Transposition: if $a \leq b$ then $\neg b \leq \neg a$ [24]. AL enriched with $\neg$ with additional

axioms: $\neg\neg A \vdash A$, $A \vdash \neg\neg A$ and the rule:

$$\text{(TRA)}\ \frac{A \vdash B}{\neg B \vdash \neg A}$$

does not admit cut elimination. With (CUT), it is complete with respect to residuated semigroups with De Morgan negation. Its product free fragment is strongly complete with respect to powerset residuated semigroups with a quasi-Boolean complement in the sense of [6]: for a fixed involutive mapping $f : M \mapsto M$ and $P \subseteq M$, one defines $-P = M - f[P]$. The full system is strongly complete with respect to cone models over semigroups (the complement is quasi-Boolean). These facts follow from representation theorems: (i) for every residuated semigroup with De Morgan negation, its product free reduct is embeddable into some powerset residuated semigroup with quasi-Boolean complement, (ii) every residuated semigroup with De Morgan negation is embeddable into some residuated semigroup of cones (over some semigroup) with quasi-Boolen complement [12].

Unfortunately, AL with De Morgan negation is not complete with respect to linguistically intended models: powerset residuated semigroups with Boolean complement [12]. Set $C = B \to B$ and consider the sequent:

$$(C \to C) \to \neg A \vdash \neg((C \to C) \to A).$$

In powerset models, $\mu(C \to C)$ must be nonempty: if $\mu(C)$ is empty then $\mu(C \to C) = M$; if $\mu(C)$ is nonempty, then $\mu(C \to C)$ is nonempty, since $\mu(C) \subseteq \mu(C \to C)$. Consequently, the sequent is true in all powerset models in which $\neg$ is interpreted as the Boolean complement of sets. It is not provable in the system, since it is not true in the following model. Set $V = \{a, b\}$ and consider the free semigroup V^+. For any string $x \in V^+$, let $f(x)$ be the mirror image of x, e.g. $f(ab) = ba$. Then, f is involutive, i.e. $f \circ f =$Id (the identity mapping). We consider the powerset residuated semigroup over V^+ with the quasi-Boolean complement, given by f. Let $\mu(B)$ be the set of all $b-$strings, and let $\mu(A)$ be the set of all strings in V^+ whose last symbol is a. Then, $\mu(C) = \mu(B)$, and $\mu(\neg A)$ equals $M - f[\mu(A)]$, i.e. the set of all strings in V^+ whose first symbol is b. Consequently, μ sends the left side of the sequent to the total set V^+. $\mu((C \to C) \to A)$ equals $\mu(A)$, hence μ sends the right side of the sequent to $M - f[\mu(A)]$. Since the latter set is not total, then the sequent is not true.

The decidability of AL with De Morgan negation seems to be an open problem, and similarly for other type logics, mentioned above.

We have discussed frames in which logical constants of type logics are interpreted by means of some operations on sets which are naturally deter-

mined by the underlying semigroup (groupoid) structure. A more general approach uses Kripke-style frames for modal logics [78, 80]. A frame is a pair (M, R) such that M is a set of states, and R is a ternary relation on M. For sets $P_1, P_2 \subseteq M$, one defines:

(K1) $P_1 \cdot P_2 = \{z \in M : (\exists x \in P_1, y \in P_2)\, R(x, y, z)\}$,

(K2) $P_1 \backslash P_2 = \{y \in M : (\forall x \in P_1, z \in M)(R(x, y, z) \Rightarrow z \in P_2)\}$,

(K3) $P_1 / P_2 = \{x \in M : (\forall y \in P_2, z \in M)(R(x, y, z) \Rightarrow z \in P_1)\}$.

The structure $(P(M), \subseteq, \cdot, \backslash, /)$ is a residuated groupoid. Every power-set residuated groupoid is a structure of this form, since one can define: $R(x, y, z)$ iff $z = xy$. Consequently, NL is strongly complete with respect to Kripke models. A similar fact holds for AL and Kripke frames, satisfying associativity: (i) if $R(x, y, z)$ and $R(z, u, v)$ then there exists w such that $R(y, u, w)$ and $R(x, w, v)$, (ii) if $R(y, u, w)$ and $R(x, w, v)$ then there exists z such that $R(x, y, z)$ and $R(z, u, v)$.

Kripke frames suggest a generalization of type logics toward multi-modal systems [80, 59]. Besides a ternary relation R, one also regards a binary relation S, and defines unary modalities $\diamondsuit, \Box$ as follows:

(K4) $\diamondsuit(P) = \{x \in M : (\exists y \in P) S(x, y)\}$,

(K5) $\Box(P) = \{x \in M : (\forall y \in M)(S(x, y) \Rightarrow y \in P)\}$.

The above are standard modalities of Classical Modal Logic. Moortgat [59] also considers minimal modalities of substructural logics [69]:

(K6) $\diamondsuit(P) = \{y \in M : (\exists x \in P) S(x, y)\}$,

and $\Box$ is defined as above. Clearly, $\diamondsuit$ yields the image under S, and $\Box$ yields the coimage under S. The linguistic role of modalities is to control the usage of structural rules in type logics. The basic logic BL is NL with the following rules for minimal modalities:

$$(\diamondsuit\mathrm{L})\ \frac{X[\bullet A] \vdash B}{X[\diamondsuit A] \vdash B},\ (\diamondsuit\mathrm{R})\ \frac{X \vdash A}{\bullet X \vdash \diamondsuit A},$$

$$(\Box\mathrm{L})\ \frac{X[A] \vdash B}{X[\bullet \Box A] \vdash B},\ (\Box\mathrm{R})\ \frac{\bullet X \vdash A}{X \vdash \Box A},$$

where $\bullet$ is a new, unary operation on type structures. Moortgat admits several modality pairs $\diamondsuit_i, \Box_i$ with the corresponding operations $\bullet_i$, for $i = 1, \ldots, n$. A restricted commutativity can be expressed, say, by the axiom:

$$\diamondsuit_1 A \otimes B \vdash B \otimes \diamondsuit_1 A,$$

and similarly for restricted associativity, contraction, etc. One can also employ several residuation triples $\otimes_j, \rightarrow_j, \leftarrow_j$ with the corresponding structure operations $\circ_j$, for $j = 1, \ldots, m$. Associativity (commutativity, contraction) may hold for some of them. Such hybrid systems can easily be modelled by Kripke frames. Some of them admit cut elimination.

An analogous formalism, under the name Generalized Lambek Calculus (GLC), has been proposed in [11]; Dunn [25] studies a closely related system with more general models. One starts from abstract algebras (M, F) such that M is a nonempty set and $F = (f_i)_{i \in I}$ is an indexed family of operations on M. On the powerset $P(M)$, one defines powerset operations:

$$f_i(P_1, \ldots, P_n) = \{f_i(x_1, \ldots, x_n) : x_1 \in P_1, \ldots, x_n \in P_n\},$$

and for each $1 \leq j \leq n$ (n is the arity of f_i), the residuation operation f_i^j is defined as follows: $f_i^j(P_1, \ldots, P_n)$ equals the set of all $x_j \in M$ such that $f_i(x_1, \ldots, x_n) \in P_j$, for all $x_k \in P_k$, $k \neq j$. The resulting structures are obvious generalizations of powerset residuated groupoids. Types of GLC are formed out of primitive types by means of logical constants $\otimes_i, \rightarrow_i^j$ and the structure operation $\circ_i$, for $i \in I$ (their arity is that of f_i). The axioms are (Id), and the inference rules are:

$$(\otimes_i \mathrm{L})\ \frac{X[\circ_i(A_1, \ldots, A_n)] \vdash B}{X[\otimes_i(A_1, \ldots, A_n] \vdash B},$$

$$(\otimes_i \mathrm{R})\ \frac{X_1 \vdash A_1; \ldots; X_n \vdash A_n}{\circ_i(X_1, \ldots, X_n) \vdash \otimes_i(A_1, \ldots, A_n)},$$

$$(\rightarrow_i^j \mathrm{L})\ \frac{X[A_j] \vdash B;\ Y_1 \vdash A_1; \ldots; Y_n \vdash A_n}{x[\circ_i(Y_1, \ldots, \rightarrow_i^j (A_1, \ldots, A_n), \ldots, Y_n)] \vdash B},$$

$$(\rightarrow_i^j \mathrm{R})\ \frac{\circ_i(A_1, \ldots, X, \ldots, A_n) \vdash A_j}{X \vdash \rightarrow_i^j (A_1, \ldots, A_n)},$$

and (NCUT). In rule $(\rightarrow_i^j \mathrm{L})$, the premise $Y_j \vdash A_j$ is dropped and the type $\rightarrow_i^j (\ldots)$ is the $j-$th argument of $\circ_i$ in the conclusion. In rule $(\rightarrow_i^j \mathrm{R})$, X is the $j-$th argument of $\circ_i$ in the premise. GLC admits cut elimination. For $n = 1$, $\otimes_i$, $\rightarrow_i^1$ and $\circ_i$ behave like $\Diamond$, $\Box$ and $\bullet$, respectively. As shown in [46], GLC is strongly complete with respect to powerset frames over abstract algebras. Clearly, $\otimes_i$ is interpreted by f_i and $\rightarrow_i^j$ by f_i^j. By cut elimination, GLC is decidable; actually, all finitely axiomatizable theories over GLC are P-Time decidable [17]. This also holds, obviously, for multi-modal systems

without additional structural postulates (rules). Grammars based on GLC generate context-free languages [42, 17].

Systems with many products can be used in linguistics to formalize different modes of composition of expressions. Instead of concatenation, one may use other operations which compose a complex expression from simpler ones. A good candidate is substitution. We enrich the lexicon V with variables $x_1, x_2, \ldots$ and define $f_i(a, b)$ to be equal to the result of substitution of b for x_i in a. In this way, we can analyse discontinuous phrases. For instance, 'if ... then ...' can be identified with 'if x_1 then x_2' and be assigned type $\rightarrow_1^1 (\rightarrow_2^1 \text{(S,S)},\text{S})$. A similar approach to discontinuity has been proposed by Morrill [61, 62].

Type logics can also be modelled by algebras of relations; see van Benthem [78, 79] for a thorough discussion of information logics formalized in this way. For relations $R, S \subseteq M^2$, one defines:

(RM1) $R \cdot S = \{(x, y) \in M^2 : \exists z((x, z) \in R \& (z, y) \in S)\}$,

(RM2) $R\backslash S = \{(x, y) \in M^2 : (\forall z)((z, x) \in R \Rightarrow (z, y) \in S)\}$,

(RM3) $S/R = \{(x, y) \in M^2 : (\forall z)((y, z) \in R \Rightarrow (x, z) \in S)\}$.

The structure $(P(M^2), \subseteq, \cdot, \backslash, /, I)$, where I is the identity relation, is a residuated monoid. It has been shown in [4] that every residuated monoid is embeddable into a structure of that kind, and consequently, AL0 and AL1 are strongly complete with respect to such models. For AL, one uses restricted frames $P(T)$ such that T is a transitive relation on M (replace M^2 with T in (RM1)-(RM3)). Other completeness and representation theorems for type logics with respect to relational models can be found in [48, 47, 19, 73].

Relational models show a link between type logics and logics of programs (dynamic logics). $R \cdot S$ corresponds to the composition of programs R and S. $R\backslash S$ (resp. S/R) is the weakest post-specification (resp. pre-specification) of program R with respect to program S in the sense of Hoare and Jifeng [35]. If one adds $+$ and $\star$, then the resulting algebras are action algebras in the sense of Pratt [68]. In opposition to Kleene algebras, based on $\cdot, +, \star$, action algebras form a variety with finitely many equational axioms [68]. Many basic properties of action algebras and logics seem unknown, as yet. Since AL and its variants are complete with respect to relational models, proof systems for these logics can be designed in the style of relational proof systems of Orłowska [63, 55]. Labelled deductive systems corresponding to relational models are studied in [48, 47].

Type logics are directly related to linear logics, introduced by Girard [32]. Linear logics are commutative and, besides logical constants $\otimes, \rightarrow, \wedge, \vee$, they

admit $\oplus$, $\neg$, 0, 1, $\top$ and $\bot$. $\oplus$ is a dual of $\otimes$; it corresponds to a composition of structures on the right hand of sequents. $\neg$ is a linear negation. 0 is the neutral element for $\oplus$ and 1 for $\otimes$, while $\top$ and $\bot$ are the lattice top and bottom.

The most easy way to explain the meaning of linear constants is to consider algebraic models of linear logic. A bilinear algebra is a residuated monoid in which there exists an element 0, satisfying:

$$0/(a\backslash 0) = a = (0/a)\backslash a,$$

for all elements a. In a bilinear algebra, one defines $a^l = 0/a$, $a^r = a\backslash 0$. One proves $(b^r a^r)^l = (b^l a^l)^r$, for all a, b; this element is denoted by $a \oplus b$.

Bilinear algebras which are bounded lattices are models of Noncommutative Linear Logic of Abrusci [1]. If they additionally satisfy: $0/a = a\backslash 0$, for all a, then they determine Cyclic Linear Logic of Yetter [82]. Commutative bilinear algebras determine Classical Linear Logic of Girard. In commutative bilinear algebras and cyclic bilinear algebras $a^l = a^r$, for all a; this operation corresponds to linear negation. The term 'bilinear algebra' is due to Lambek [51]. Bilinear algebras determine the multiplicative fragment of Noncommutative Linear Logic. As shown in [2], AL1 is a conservative, multiplicative fragment of both Noncommutative Linear Logic and Cyclic Linear Logic. Girard interprets Classical Linear Logic by means of phase spaces. Phase spaces are powerset frames over commutative monoids with a distinguished set $P \subseteq M$. Sets of the form P/Q, for $Q \subseteq M$, are called facts. Linear constants are interpreted as operations on facts. For instance, linear negation correspond to the operation $Q^\bot = P/Q$ and $\otimes$ to the operation $Q_1 \cdot Q_2 = (Q_1 Q_2)^{\bot\bot}$. Linear logics show many profound interactions with logical proof theory and theoretical computer science. Their role for linguistics remains not quite clear, since linear constants lack a natural linguistic sense. However, a graph-theoretic representation of proofs in the form of proof nets in multiplicative fragments of linear logics has attracted some linguists; see e.g. Casadio [21].

At the end of this subsection, we mention a recent innovation in type logics, due to Lambek [52]. Instead of binary operations $\backslash, /$, Lambek uses unary operations l and r, of left and right adjoint, respectively. *A pregroup* is a structure $(M, \leq, \cdot, l, r, 1)$ such that $(M, \leq, \cdot, 1)$ is a p.o. monoid, and the following inequalities:

$$\text{(PRE)}\ a^l a \leq 1 \leq a a^l,\ a a^r \leq 1 \leq a^r a,$$

hold, for every $a \in M$. Equivalently, pregroups are bilinear algebras in which $\cdot = \oplus$ and $0 = 1$. Commutative pregroups coincide with commutative p.o.

groups. A pregroup M is a p.o. group iff $a^l = a^r$, for all $a \in M$. Pregroups which are not p.o. groups are said to be *proper*. As shown in [15], proper pregroups can be neither finite, nor totally ordered, nor bounded. Every pregroup is a residuated monoid with operations $\backslash, /$ defined by: $a\backslash b = a^r b$ and $a/b = ab^l$. A residuated monoid is a pregroup iff $a\backslash(bc) \leq (a\backslash b)c$ and $(ab)/c \leq a(b/c)$, for all elements a, b, c (then, $a^l = 1/a$, $a^r = a\backslash 1$).

Concrete pregroups can be formed out of order preserving functions on a poset $(W, \leq)$ which are downward and upward unbounded:

$$\forall x \exists y(f(y) \leq x),\ \forall x \exists y(f(y) \geq x).$$

One defines $f \leq g$ iff $f(x) \leq g(x)$, for all x, and $(f \cdot g)(x) = f(g(x))$. The so-defined structure is a p.o. monoid with the identity function as the unit. In every p.o. monoid, l and r can be defined as partial operations: a^l is the unique b (if exists) such that $ba \leq 1 \leq ab$, and a^r is the unique b (if exists) such that $ab \leq 1 \leq ba$. Every p.o. monoid contains a greatest pregroup: it consists of all elements a such that $a^{l\cdots l}$ and $a^{r\cdots r}$ exist, for all finite iterations of l and r (in pregroups $a^{lr} = a^{rl} = a$, so mixed iterations need not be considered). In some cases, this greatest pregroup is proper. Lambek's example is the pregroup of all unbounded, order preserving functions on the poset of integers. Other models have been studied in [15].

For linguistics, Lambek offers free pregroups, i.e. pregroups induced by the formal theory (with special axioms of the form $p \vdash q$). For instance, if p_i is the type of personal pronouns in $i-$th Person, and p is the common type of personal pronouns, then $p_i \vdash p$ is a special axiom. If s is the type of sentences, then 'works' is assigned type $p_3^r s$. Accordingly, 'he works' is parsed as $p_3 p_3^r s$, hence it reduces to s, by (PRE). This method has successfully been used by Lambek and his followers to analyse quite delicate syntactic structures of English, German, Italian and other ethnic languages; see e.g. [22, 53, 45].

Free pregroups can be presented in the form of a formal system; Lambek calls it Compact Bilinear Logic. Primitive types are as above. Atoms are of the form $p^{(n)}$, for primitive p and arbitrary integers n. We interprete atoms as follows: $p^{(0)} = p$, $p^{(n+1)} = (p^{(n)})^r$, $p^{(n-1)} = (p^{(n)})^l$. Types are finite strings of atoms. In pregroups, $(ab)^l = b^l a^l$ and $(ab)^r = b^r a^r$. Thus, for type $a_1 \ldots a_k$, the left adjoint is $a_k^l \ldots a_1^l$, and the right adjoint is $a_k^r \ldots a_1^r$. Special axioms form a finite set of rules $p \vdash q$, for primitive p, q. The system is based on the following rewriting rules:

(P-CON) Contraction: $\Gamma a^{(n)} a^{(n+1)} \Delta \vdash \Gamma\Delta$,

(P-EXP) Expansion: $\Gamma\Delta \vdash \Gamma a^{(n+1)} a^{(n)} \Delta$,

(P-IND) Induced Step: $\Gamma a^{(n)}\Delta \vdash \Gamma b^{(n)}\Delta$ if either n is even and $a \vdash b$ is a special axiom, or n is odd and $b \vdash a$ is a special axiom.

(P-IND) expresses tha fact that in pregroups $a \leq b$ entails $b^l \leq a^l$ and $b^r \leq a^r$. A derivation is a standard rewriting procedure. Lambek [52] shows an important property: if Γ reduces to Γ', then there is a type Δ such that Γ reduces to Δ by (P-CON) and (P-IND) only, and Δ reduces to Γ' by (P-EXP) and (P-IND) only (the switching lemma). As a consequence, if Γ reduces to an atom or ϵ, then the reduction can use (P-CON) and (P-IND) only. Such reductions are derivations in CF-grammars based on (a finite set) of the rewriting rules. Accordingly, grammars based on free pregroups generate context-free languages. The provability is P-TIME decidable, since $\Gamma \vdash \Delta$ is provable iff $\Delta^l\Gamma$ reduces to ϵ [15].

The logic of pregroups can be formalized as a sequent system which admits cut elimination; two versions of such a system are formulated in [16]. The cut elimination theorem for them is closely related to the switching lemma (they are deducible from each other). Let us recall the one-sided version. Atomic formulas are atoms (in the above sense) and the constant 1. Formulas are formed out of atomic formulas by $\otimes$. Adjoints are defined in the metalanguage. For terms, the definition is given above. We set $1^l = 1^r = 1$ and $(A \otimes B)^d = A^d \otimes B^d$, for $d = l, r$. Sequents are finite strings of formulas. The axiom is ϵ, and the inference rules are:

$$\frac{\Gamma AB\Gamma'}{\Gamma(A \otimes B)\Gamma'}, \quad \frac{\Gamma\Gamma'}{\Gamma 1\Gamma'},$$

$$\frac{\Gamma\Gamma'}{\Gamma AA^l\Gamma'}, \quad \frac{\Gamma\Gamma'}{\Gamma A^r A\Gamma'}.$$

The cut elimination theorem is the following: (i) if ΓA^l and $A\Delta$ are provable then $\Gamma\Delta$ is provable, (ii) if ΓA and $A^r\Delta$ are provable, then $\Gamma\Delta$ is provable. Γ reduces to Δ in the rewriting system iff $Delta\Gamma^l$ is provable in the sequent system.

We had began with a simple logic AB, went through a variety of type logics, and have finished at a strong system of Compact Bilinear Logic. Relatively weak systems like AB, NL and the strong logic of pregroups are P-TIME decidable, while the intermediate ones are NP-complete. All systems except the last one can easily be interpreted as fragments of Intuitionistic Logic or type logics of lambda calculus. This seems to be impossible for Compact Bilinear Logic, hence its semantical validity remains an open problem.

References

[1] V.M. Abrusci, Phase Semantics and Sequent Calculus for Pure Noncommutative Classical Linear Logic, *Journal of Symbolic Logic* 56 (1991), 1403-1451.

[2] V.M. Abrusci, Classical Conservative Extensions of Lambek Calculus, *Studia Logica* 71.3 (2002), 277-314.

[3] K. Ajdukiewicz, Die syntaktische Konnexität, *Studia Philosophica* 1 (1935), 1-27.

[4] H. Andréka and S. Mikulás, Lambek Calculus and Its Relational Semantics. Completeness and Incompleteness, *Journal of Logic, Language and Information* 3 (1994), 1-37.

[5] Y. Bar-Hillel, C. Gaifman and E. Shamir, On categorial and phrase structure grammars, *Bull. Res Council Israel* F 9 (1960), 155-166.

[6] A. Białynicki-Birula and H. Rasiowa, On the Representation of Quasi-Boolean Algebras, *Bull. Acad. Polonaise Scie.* 5 (1957), 259-261.

[7] W. Buszkowski, Some Decision Problems in the Theory of Syntactic Categories, *Zeitschrift f. math. Logik u. Grundlagen der Mathematik* 28 (1982), 539-548.

[8] W. Buszkowski, Completeness Results for Lambek Syntactic Calculus, *Zeitschrift f. math. Logik u. Grundlagen der Mathematik* 32 (1986), 13-28.

[9] W. Buszkowski, Generative Capacity of Nonassociative Lambek Calculus, *Bull. Polish Academy Scie. Math.* 34 (1986), 507-516.

[10] W. Buszkowski, Gaifman's Theorem on Categorial Grammars Revisited, *Studia Logica* 47 (1988), 23-33.

[11] W. Buszkowski, *Logical Foundations of Ajdukiewicz-Lambek Categorial Grammars* (in Polish), Polish Scientific Publishers, Warsaw, 1989.

[12] W. Buszkowski, Categorial Grammars with Negative Information, in: H. Wansing (ed.), *Negation. A notion in focus*, de Gruyter, Berlin, 1996, 107-126.

[13] W. Buszkowski, The Ajdukiewicz Calculus, Polish Notation and Hilbert Style Proofs, in: K. Kijania-Placek and J. Woleński (eds.), *The Lvov-Warsaw School and Contemporary Philosophy*, Kluwer, Dordrecht, 1998, 241-252.

[14] W. Buszkowski, Mathematical Linguistics and Proof Theory, in: J. van Benthem and A. ter Meulen (eds.), *Handbook of Logic and Language*, Elsevier, Amsterdam, 1997, 683-736.

[15] W. Buszkowski, Lambek Grammars Based on Pregroups, in: P. de Groote, G. Morrill and C. Retoré (eds.), *Logical Aspects of Computational Linguistics*, LNAI 2099, Springer, Berlin, 2001, 95-109.

[16] W. Buszkowski, Sequent Systems for Compact Bilinear Logic, *Mathematical Logic Quarterly* (2003), to appear.

[17] W. Buszkowski, Lambek Calculus with Nonlogical Axioms, in: C. Casadio et al. (eds.), *A Festschrift for Jim Lambek*, CSLI Publications, Stanford, to appear.

[18] W. Buszkowski and B. Dziemidowicz, Restricted Optimal Unification, in: Y. Hamamatsu et al. (eds.), *Formal Methods and Intelligent Techniques in Control, Decision Making, Multimedia and Robotics*, Warsaw, 2000, 1-8.

[19] W. Buszkowski and M. Kołowska-Gawiejnowicz, Representation of Residuated Semigroups in Some Algebras of Relations. The Method of Canonical Models, *Fundamenta Informaticae* 31 (1997), 1-12.

[20] W. Buszkowski and G. Penn, Categorial Grammars Determined from Linguistic Data by Unification, *Studia Logica* 49 (1990), 431-454.

[21] C. Casadio, Non-Commutative Linear Logic in Linguistics, *Grammars* 3/4 (2001), 1-19.

[22] C. Casadio and J. Lambek, An Algebraic Analysis of Clitic Pronouns in Italian, in: P. de Groote, G. Morrill and C. Retoré (eds.), *Logical Aspects of Computational Linguistics*, LNAI 2099, Springer, Berlin, 2001, 110-124.

[23] K. Došen, A Completeness Theorem for the Lambek Calculus of Syntactic Categories, *Zeitschrift f. math. Logik u. Grundlagen der Mathematik* 31 (1985), 235-241.

[24] J.M. Dunn, Perp and Star: Two Treatments of Negation, in: J. Tomberlin (ed.), *Philosophical Perspectives (Philosophy of Language and Logic)* 7 (1993), 331-357.

[25] J.M. Dunn, Partial Gaggles Applied to Logics with Restricted Structural Rules, in: P. Schroeder-Heister and K. Došen (eds.), *Substructural Logics*, Clarendon Press, Oxford, 1993, 63-108.

[26] B. Dziemidowicz, Optimal Unification and Learning Algorithms for Categorial Grammars, *Fundamenta Informaticae* 49 (2002), 297-308.

[27] B. Dziemidowicz, On Learnability of Restricted Classes of Categorial Grammars, *Fundamenta Informaticae*, to appear.

[28] D.M. Gabbay, A General Theory of Structured Consequence Relations, in: P. Schroeder-Heister and K. Došen (eds.), *Substructural Logics*, Clarendon Press, Oxford, 1993, 109-151.

[29] D¿M. Gabbay, *Labelled Deductive Systems*, Oxford University Press, Oxford, 1996.

[30] M. Gécség and M. Steinby, *Tree Automata*, Akademiai Kiadó, Budapest, 1984.

[31] E.M. Gold, Language Identification in the Limit, *Information and Control* 10 (1867), 447-474.

[32] J.Y. Girard, Linear logic, *Theoretical Computer Science* 50 (1987), 1-102.

[33] P. de Groote and F. Lamarche, Classical Non-Associative Lambek Calculus, *Studia Logica* 71.3 (2002), 355-388.

[34] H. Hiż, Grammar logicism, *The Monist* 51 (1967), 110-127.

[35] C.A.R. Hoare and H. Jifeng, The weakest prespecification, *Fundamenta Informaticae* 9 (1986), 51-84, 217-252.

[36] E. Husserl, *Logische Untersuchungen*, Halle, 1900-1901.

[37] M. Kanazawa, The Lambek Calculus Enriched with Additional Connectives, *Journal of Logic, Language and Information* 1.2 (1992), 141-171.

[38] M. Kanazawa, Identification in the Limit of Categorial Grammars, *Journal of Logic, Language and Information* 5 (1996), 115-155.

[39] M. Kanazawa, *Learnable Classes of Categorial Grammars*, CSLI Publications, Stanford, 1988.

[40] M. Kandulski, The Equivalence of Nonassociative Lambek Categorial Grammars and Context-Free Grammars, *Zeitschrift f. math. Logik u. Grundlagen der Mathematik* 34 (1988), 41-52.

[41] M. Kandulski, Normal Form of Derivations in the Nonassociative and Commutative Lambek Calculus with Product, *Mathematical Logic Quarterly* 39 (1993), 103-114.

[42] M. Kandulski, On Generalized Ajdukiewicz and Lambek Calculi and Grammars, *Fundamenta Informaticae* 30.2 (1997), 169-181.

[43] M. Kandulski, Derived Tree Languages of Nonassociative Lambek Categorial Grammars with Product, *Fundamenta Informaticae* (2003), to appear.

[44] S. Kapur, *Computational Learning of Languages*, Ph.D. Thesis, Cornell University, 1991.

[45] A. Kiślak, Pregroups versus English and Polish grammar, in: V.M. Abrusci and C. Casadio (eds.), *New Perspectives in Logic and Formal Linguistics*, Papers in Formal Linguistics and Logic, Bulzoni Editore, Roma, 2002, 129-154.

[46] M. Kołowska-Gawiejnowicz, Powerset Residuated Algebras and Generalized Lambek Calculus, *Mathematical Logic Quarterly* 43 (1997), 60-72.

[47] M. Kołowska-Gawiejnowicz, A Labelled Deductive System for Relational Semantics of the Lambek Calculus, *Mathematical Logic Quarterly* 45 (1999), 51-58.

[48] N. Kurtonina, *Frames and Labels. A modal analysis of categorial inference*, Ph.D. Thesis, University of Utrecht, 1995.

[49] J. Lambek, The mathematics of sentence structure, *American Mathematical Monthly* 65 (1958), 154-170.

[50] J. Lambek, On the Calculus of Syntactic Types, in: R. Jakobson (ed.), *Structure of Language and Its Mathematical Aspects*, Proc. Symp. Appl. Math., AMS, Providence, 1961, 166-178.

[51] J. Lambek, Bilinear Logic in Algebra and Linguistics, in: J.Y. Girard, Y. Lafont and L. Regnier (eds.), *Advances in Linear Logic*, Cambridge University Press, Cambridge, 1995, 43-59.

[52] J. Lambek, Type Grammars Revisited, in: A. Lecomte, F. Lamarche and G. Perrier (eds.), *Logical Aspects of Computational Linguistics*, LNAI 1582, Springer, Berlin, 1999, 1-27.

[53] J. Lambek, Type Grammars as Pregroups, *Grammars* 4 (2001), 21-39.

[54] S. Leśniewski, Grundzüge einer neuen System der Grundlagen der Mathematik, *Fundamenta Mathematicae* 14 (1929), 1-81.

[55] W. MacCaull and E. Orłowska, Correspondence Results for Relational Proof Systems with Application to the Lambek Calculus, *Studia Logica* 71.3 (2002), 389-414.

[56] J. Marciniec, Learning Categorial Grammars by Unification with Negative Constraints, *Journal of Applied Non-Classical Logics* 4 (1994), 181-200.

[57] R. Montague, *Formal Philosophy*, Selected papers of R. Montague edited by R. Thomason, Yale University Press, New Haven, 1974.

[58] M. Moortgat, *Categorial Investigations. Logical and Linguistic Aspects of the Lambek Calculus*, Foris, Dordrecht, 1988.

[59] M. Moortgat, Categorial Type Logics, in: J. van Benthem and A. ter Meulen (eds.), *Handbook of Logic and Language*, Elsevier, Amsterdam, 1997, 93-177.

[60] M. Moortgat, Structural Equations in Language Learning, in: P. de Groote, G. Morrill and C. Retoré (eds.), *Logical Aspects of Computational Linguistics*, LNAI 2099, Springer, Berlin, 2001, 1-16.

[61] G. Morrill, *Type Logical Grammar. Categorial Logic of Signs*, Kluwer, Dordrecht, 1994.

[62] G. Morrill, A Generalised Discontinuity, manuscript, LICS, Ottawa, 2003.

[63] E. Orłowska, Relational Proof System for Relevant Logics, *Journal of Symbolic Logic* 57 (1992), 1425-1440.

[64] D. Osherson, D. de Jongh, E. Martin and S. Weinstein, Formal learning theory, in: J. van Benthem and A. ter Meulen (eds.), *Handbook of Logic and Language*, Elsevier, Amsterdam, 1997, 737-775.

[65] M. Pentus, Lambek Grammars are Context-Free, *Proc. of 8th IEEE Symposium on Logic in Computer Science*, 1993, 429-433.

[66] M. Pentus, Models for the Lambek Calculus, *Annals of Pure and Applied Logic* 75 (1995), 179-213.

[67] M. Pentus, Lambek calculus is NP-complete, manuscript, Moscow State University, 2003.

[68] V. Pratt, Action Logic and Pure Induction, in: J. van Eijck (ed.), *Logics in AI*, LNAI 478, Springer, Berlin, 1991, 97-120.

[69] G. Restall, *An Introduction to Substructural Logics*, Routledge, London, 2001.

[70] D. Roorda, *Resource Logics. Proof-Theoretical Investigations*, Ph.D. Thesis, University of Amsterdam, 1991.

[71] T. Shinohara, Inductive Inference of Monotonic Formal Systems from Positive Data, in: S. Arikawa et al. (eds.), *Algorithmic Learning Theory*, Springer, Tokyo, 1990, 339-351.

[72] T.A. Sudkamp, *Languages and Machines. An Introduction to the Theory of Computer Science*, Addison-Wesley, Reading, 1991.

[73] M. Szczerba, Representation Theorems for Residuated Groupoids, in: C. Retoré (ed.), *Logical Aspects of Computational Linguistics*, LNAI 1328, Springer, Berlin, 1997, 426-434.

[74] H-J. Tiede, *Deductive Systems and Grammars*, Ph.D. Thesis, Indiana University, 1999.

[75] H. Uszkoreit, Categorial Unification Grammars, *Proc. 11th International Conference on Computational Linguistics*, Bonn, 1986, 187-194.

[76] J. van Benthem, *Essays in Logical Semantics*, D. Reidel, Dordrecht, 1986.

[77] J. van Benthem, Categorial Equations, in: E. Klein and J. van Benthem (eds.), *Categories, Polymorphism and Unification*, University of Amsterdam, 1987, 1-17.

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

[79] J. van Benthem, *Exploring Logical Dynamics*, CSLI, Stanford, 1996.

[80] J. van Benthem, Categorial Grammar at a Cross-Roads, in: C. Casadio et al. (eds.), *A Festschrift for Jim Lambek*, CSLI Publications, to appear.

[81] K. Wright, Identification of unions of languages drawn from an identifiable class, in: *The 1989 Workshop on Computational Learning Theory*, Morgan Kaufmann, San Mateo, 1989, 328-333.

[82] D.N. Yetter, Quantales and (Non-Commutative) Linear Logic, *Journal of Symbolic Logic* 55 (1996), 41-64.

[83] W. Zielonka, Axiomatizability of Ajdukiewicz-Lambek Calculus by means of Cancellation Schemes, *Zeitschrift f. math. Logik u. Grundlagen der Mathematik* 27 (1981), 215-224.

[84] W. Zielonka, On Reduction Systems Equivalent to the Lambek Calculus with the Empty String, *Studia Logica* 71.1 (2002), 31-46.