Fundamenta Informaticae XX (2006) 1–15
IOS Press

Rough Sets and Learning by Unification

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

Abstract. We apply basic notions of the theory of rough sets, due to Pawlak [30, 31], to explicate certain properties of unification-based learning algorithms for categorial grammars, introduced in [6, 11] and further developed in e.g. [18, 19, 24, 25, 26, 9, 28, 14, 15, 3]. The outcomes of these algorithms can be used to compute both the lower and the upper approximation of the searched language with respect to the given, finite sample. We show that these methods are also appropriate for other kinds of formal grammars, e.g. context-free grammars and context-sensitive grammars.

1. Introduction

Formal learning theory is concerned with algorithms which compute formal grammars on the basis of finite data: finite sets of expressions from the searched language [29]. The language L is presented as an enumeration $(e_n)_{n\in\omega}$ of all its expressions. For each n, the algorithm is executed on input $(e_0, \ldots, e_n)$ and returns a grammar G_n from a fixed class of grammars $\mathcal{G}$. The algorithm succeeds, if there exists $G \in \mathcal{G}$ such that G is a correct grammar for L and $G_n = G$, for all but finitely many n. The class $\mathcal{G}$ is said to be (effectively) learnable, if there exists an algorithm which succeeds for any language generated by a grammar from $\mathcal{G}$ and any possible enumeration of this language.

These are basic notions of formal learning theory, more strictly, the theory of Gold [17] of learning from positive data and success as identification of language in the limit. We precisely define them in section 3. Gold observed that no class of Chomsky hierarchy is learnable from positive data. Learnability requires some additional constraints. For instance, for any integer $k > 0$, the class of context-sensitive grammars with at most k production rules and the class of $k-$valued classical categorial grammars are learnable [34, 18, 19]. If also unrestricted negative data are admitted, then any recursively enumerable class of grammars is learnable, whenever its universal membership problem is decidable.

Address for correspondence: Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Umultowska 87, 61-614 Poznań, Poland

Positive learnability results often use a special property of classes of languages (grammars), called *finite elasticity*. Every class of grammars, satisfying the general conditions, formulated at the end of the last paragraph, and having finite elasticity, is learnable [35, 20]. Algorithms, stemming from finite elasticity, run through a (computable) sequence $(G_n)_{n\in\omega}$ of all grammars from $\mathcal{G}$ and return the first grammar, compatible with the input and, possibly, fulfilling certain additional constraints.

A disadvantage of such algorithms is that the grammar G_n, returned on step n, brings, usually, no new information about the searched language, which is not transparent in the input data. Only the limit grammar $G = G_n$, for all but finitely many n, correctly describes the language. Unfortunately, except for some very special cases, the learner does not know that the limit has already been attained. Furthermore, such algorithms are extremely inefficient.

In this paper we show that some other algorithms can yield much more. It is expedient to use the basic idea of rough set theory of Pawlak [30, 31]. One disposes with an incomplete knowledge about properties of objects from some universe. A set of objects X is represented as a pair of two sets: the lower approximation of X and the upper approximation of X, here denoted by $\mathrm{LA}(X)$ and $\mathrm{UA}(X)$, respectively. $\mathrm{LA}(X)$ consists of those objects which necessarily belong to X, and $\mathrm{UA}(X)$ of those objects which possibly belong to X, on the basis of one's knowledge. Rough sets were extensively studied as a framework of knowledge representation and reasoning about data (see e.g. [12, 33, 13, 32]). As far as the author knows, they have not yet been applied in formal learning theory in the sense of this paper.

The learning algorithms, based on unification, worked out for classical categorial grammars in [6, 11] and further elaborated in [18, 19], can be arranged in such a way that, on each step n, they return a finite description of both the lower approximation and the upper approximation of the searched language on the basis of the input data $(e_0, \ldots, e_n)$. These notions are precisely defined in section 3.

In section 2 we recall some basic notions of the theory of formal grammars. Section 3 discusses learnability and related concepts. Algorithms based on unification are considered in section 4. Section 5 contains our main results for different kinds of (not only categorial) grammars.

2. Formal grammars

We start from classical categorial grammars, playing an essential role in our exposition.

Types are formed out of *atomic types* (variables) $p, q, r, \ldots$ by means of operation symbols $\backslash, /$. The type logic AB (after Ajdukiewicz [1] and Bar-Hillel [2]) is a rewriting system, based on the following rules:

$$\text{(AB1) } \Gamma, A, A\backslash B, \Delta \Rightarrow \Gamma, B, \Delta, \quad \text{(AB2) } \Gamma, B/A, A, \Delta \Rightarrow \Gamma, B, \Delta.$$

Γ, Δ stand for arbitrary finite sequences of types. $\Rightarrow^*$ denotes the reflexive and transitive closure of the relation $\Rightarrow$, defined by (AB1), (AB2).

A (classical) categorial grammar is defined as a triple $G = (\Sigma_G, I_G, S_G)$; Σ_G is a nonempty finite lexicon (alphabet), I_G is a finite relation between elements of Σ_G and types, and S_G is a designated variable. Σ_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 $(v, A) \in I_G$. G *assigns* a type A to a string $a = v_1 \ldots v_n$, $v_i \in \Sigma_G$, if there are types A_i such that $G : v_i \mapsto A_i$, for $i = 1, \ldots, n$, and $A_1, \ldots, A_n \Rightarrow^* A$; we write $a \mapsto_G A$. *The language* of G ($L(G)$) is the set of all strings $a \in \Sigma_G^+$ such that $a \mapsto_G S_G$.

Functor-argument structures (f-structures) are syntactic structures, provided by categorial grammars. The set $F(\Sigma)$, of f-structures on Σ, is recursively defined as follows: (i) $\Sigma \subseteq F(\Sigma)$, (ii) if $X, Y \in F(\Sigma)$ then $(XY)_1 \in F(\Sigma)$ and $(XY)_2 \in F(\Sigma)$. X is *the functor* and Y is *the argument* in $(XY)_1$, and the converse holds for $(XY)_2$. Types can be identified with f-structures on the set of variables: write $(AB)_2$ for $A\backslash B$, and $(AB)_1$ for A/B. *Phrase structures* (p-structures) are obtained from f-structures by dropping functor indices.

Let G be a categorial grammar. We define a relation $X \mapsto_G A$, for $X \in F(\Sigma_G)$ and types A: (i) $v \mapsto_G A$ whenever $A \in I_G(v)$, (ii) if $X \mapsto_G A/B$ and $Y \mapsto_G B$ then $(XY)_1 \mapsto_G A$, (iii) if $X \mapsto_G A$ and $Y \mapsto_G A\backslash B$ then $(XY)_2 \mapsto_G B$. *The f-language* of G ($L_f(G)$) is the set of all X such that $X \mapsto_G S_G$. *The p-language* of G ($L_p(G)$) consists of all p-structures, underlying the f-structures from $L_f(G)$.

For instance, let G admit the following assignment:

$$\text{Kazimierz} \mapsto A, \text{ teaches} \mapsto B, \text{ excellently} \mapsto C,$$

where A =PN (proper noun), $B = A\backslash$S, $C = B\backslash B$, and S_G =S (sentence). Then, $L_f(G)$ contains infinitely many structures, e.g.:

1. (Kazimierz teaches)$_2$,
2. (Kazimierz (teaches excellently)$_2$)$_2$,
3. (Kazimierz ((teaches excellently)$_2$ excellently)$_2$)$_2$,

and so on. $L_p(G)$ (resp. $L(G)$) contains the p-structures (resp. strings of words) resulting from the above structures.

The above grammar is 1-valued (rigid), this means: it assigns at most one type to any lexical atom. This property 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 in the sentence 'not every man works' with structure (not ((every man) works)), but also type C/C (C defined as above) in 'John works not hardly'. A categorial grammar G is said to be $k-$*valued*, if I_G assigns at most k types to each $v \in \Sigma_G$.

On the level of strings, categorial grammars generate the same languages as ($\epsilon-$free) context-free grammars [2].*A context-free grammar* (CF-grammar) is a quadruple $G = (\Sigma_G, N_G, R_G, S_G)$ such that Σ_G, N_G are disjoint finite alphabets, $S_G \in N_G$, and R_G is a finite subset of $N_G \times (\Sigma_G \cup N_G)^*$. Elements of Σ_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 a string b is *directly derivable* from a 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 a string b is *derivable* from a string a in G (write $a \Rightarrow_G^* b$) 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 $a \in \Sigma_G^*$ such that $S_G \Rightarrow_G^* a$. A CF-grammar G is said to be $\epsilon-$*free*, if a is nonempty, for any rule $(A \mapsto a) \in R_G$. Clearly, if G is $\epsilon-$free, then $L(G) \subseteq \Sigma_G^+$.

Grammars G, G' are said to be (weakly) *equivalent* if $L(G) = L(G')$. It is well-known 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 $A, B, C \in N_{G'}$, $v \in \Sigma_{G'}$.

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

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

$$A \mapsto \text{Kazimierz},\ B \mapsto \text{teaches},\ C \mapsto \text{excellently},$$

with $S_G = \mathrm{S}$, $N_G = \{S, A, B, C\}$, and Σ_G consisting of lexical atoms 'Kazimierz', 'teaches' and 'excellently'. A derivation of 'Kazimierz teaches excellently' is:

$$\mathrm{S} \Rightarrow AB \Rightarrow ABC \Rightarrow^* \text{Kazimierz teaches excellently}.$$

Every CF-derivation determines a unique phrase structure on the derived string; the above derivation leads to structure (Kazimierz (teaches excellently)). *The p-language* of CF-grammar G ($L_p(G)$) consists of all phrase structures of strings from $L(G)$ which are determined by possible derivations of these strings. One could also preserve nonterminals appearing in the derivation; the resulting labeled phrase structure (Kazimierz $_A$(teaches $_B$excellently$_C$)$_B$)$_S$ is a linear representation of a derivation tree.

A principal difference between categorial grammars and CF-grammars is that the former are *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 put the linguistic information in the production rules which underly the derivation procedure.

Lexicality is characteristic of all kinds of categorial grammar: the universal rules for derivation procedures are provable sequents of some logics, being independent of the particular language. Besides classical categorial grammars, defined above, there are studied categorial grammars, based on richer logics, e.g. different variants of the Lambek calculus and substructural logics [21, 22, 4, 8, 27, 10].

Classical categorial grammars are not equivalent to CF-grammars on the level of phrase structures. Let $P(\Sigma)$ denote the set of all phrase structures on alphabet Σ. Let $L \subseteq P(\Sigma)$. We say that $X \in P(\Sigma)$ is equivalent to $Y \in P(\Sigma)$ with respect to L (write $X \sim_L Y$) if, for all $Z \in P(\Sigma)$, $Z[X] \in L$ 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). It is well-known [16] that $L = L_p(G)$, for some CF-grammar G, iff the relation $\sim_L$ is of finite index. By *the external degree* of $X \in P(\Sigma)$ (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 is known [7] that $L = L_p(G)$, for some classical categorial grammar G, iff both $\sim_L$ is of finite index and L is of bounded degree. Consequently, p-languages of classical categorial grammarss form a proper subclass of the class of p-languages of CF-grammars.

A similar characterization can be given for f-languages generated by classical categorial grammars: for $L \subseteq F(\Sigma)$, there is a grammar 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 [7, 8]). Consequently, f-languages of CCGs are regular tree languages. Standard techniques of tree automata [16] yield the decidability of the emptiness problem, the inclusion problem and the equality problem for these languages; for a discussion, see [7]. In particular, the inclusion problem $L^f(G) \subseteq L^f(G')$ is decidable, and similarly for p-languages of classical categorial grammars and CF-grammars.

3. Learning grammars

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. We assume that there exists a recursive isomorphism of Ω onto the set of natural numbers ω.

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 $\varphi((s_i)_{i\leq n})$ 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 $\mathcal{G}$; an algorithm which computes φ is called *a learning algorithm* for $\mathcal{G}$.

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 $L(\mathcal{G})$ contains 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, classical categorial grammars, 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 [34]. Kanazawa [18, 19] shows that $k-$valued classical categorial grammars are learnable.

Wright [35] defines 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 [20] proves 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 $D_L \subseteq L'$.

In [10], it is shown that finite elasticity is equivalent to the following:

(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 $L \subseteq L'$.

This yields Kapur's theorem, since (D) follows from (D'). If $\mathcal{L}$ is a closure system, this 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 $i \in \omega$ [10].

Let $\mathcal{G} \subseteq \Omega$. We always assume that $\mathcal{G}$ satisfies the following conditions:

(G1) $\mathcal{G}$ is recursively enumerable,

(G2) the universal membership problem $e \in L(G)$, for $e \in E$, $G \in \mathcal{G}$, is decidable.

If $\mathcal{G}$ is recursively enumerable, then there exists a computable sequence $(G_n)_{n\in\omega}$, of all grammars from $\mathcal{G}$.

If $\mathcal{G}$ satisfies (G1), (G2), and $L(\mathcal{G})$ satisfies (D), then $\mathcal{G}$ is learnable [20]. We sketch the proof.

Proof:
We consider two cases.

Case 1. The inclusion problem $L(G_1) \subseteq L(G_2)$, for $G_1, G_2 \in \mathcal{G}$, is decidable (this holds for structure languages of categorial grammars and CF-grammars). Let $(G_n)_{n\in\omega}$ be a computable sequence of all grammars from $\mathcal{G}$. The learning function φ is defined as follows. Let $(s_i)_{i\leq n}$ be a finite sequence of expressions from E. One considers grammars $G_0, \ldots, G_n$. One marks those grammars G_j, $j = 1, \ldots, n$, which satisfy $s_0, \ldots, s_n \in L(G_j)$ (if no G_j fulfilling this condition exists, then φ is undefined). Then, one takes the first marked grammar G_j whose language is minimal (with respect to inclusion) among languages of marked grammars (this means, there exists no marked grammar G_k such that $L(G_k) \subset L(G_j)$), and this grammar is the value of φ for the given sequence. Now, let (s_n) be a text for some language $L \in L(\mathcal{G})$. By (D), there exists a finite set $D_L \subseteq L$ such that L is the smallest language from $L(\mathcal{G})$ which contains D_L. There is $n' \in \omega$ such that $D_L \subseteq \{s_0, \ldots, s_m\}$, for all $m \geq n'$. Let G_i be the first grammar in the sequence (G_n) such that $L(G_n) = L$. Clearly, for each $m \geq \max(i, n')$, G_i will be taken as the value of φ for $(s_i)_{i\leq m}$.

Case 2. The inclusion problem is undecidable (this holds for string languages of classical categorial grammars and CF-grammars). The construction is similar to the one for Case 1 except that we additionally employ a fixed sequence $(e_n)_{n\in\omega}$ of all expressions from E and define $E_n = \{e_0, \ldots, e_n\}$. The value of φ for $(s_i)_{i\leq n}$ is defined as the first marked grammar G_j such that $L(G_j) \cap E_n$ is minimal among languages $L(G_k) \cap E_n$, for marked grammars G_k. Now, G_i, defined in Case 1, need not be the value of φ on $(s_i)_{i\leq m}$, for all $m \geq \max(i, n')$; it may happen that $\varphi((s_i)_{i\leq m}) = G_l$, for some $l < i$ such that $L(G_l) \cap E_m = L(G_i \cap E_m$. This may happen for finitely many m only, since L is the smallest language from $L(\mathcal{G})$ containing D_L. Accordingly, φ converges to G_i. □

We have only regarded positive data. Many learning procedures take into account positive and negative data. *A pn-text* for a language L can be defined as a sequence $((e_n, d_n))_{n\in\omega}$ such that $(e_n)_{n\in\omega}$ is a sequence of all expressions from E, and $d_n = 1$ if $e_n \in L$, $d_n = 0$ if $e_n \notin L$.

It was observed by Gold [17] that any class of grammars satisfying (G1), (G2) is learnable from positive and negative data. Let $(G_n)_{n\in\omega}$ be a computable sequence of all grammars from $\mathcal{G}$. φ is defined on $(e_i, d_i)_{i\leq n}$ as the first grammar G_j in the sequence which is compatible with the data. One easily shows that, for any pn-text for $L \in L(\mathcal{G})$, φ converges to a grammar for L.

Many results in formal learning theory were obtained by analogous methods; the main technical issue was to prove finite elasticity of a class of languages. Algorithms based on the search for the first grammar in the sequence of all grammars which is compatible with the data have some drawbacks. Clearly, the procedure of running a computable sequence of all grammars from some infinite class is strongly inefficient. To exhibit other drawbacks, we use some ideas of rough set theory of Pawlak [30, 31], namely a representation of a set of objects by two sets: the lower approximation and the upper approximation of this set.

Let $\mathcal{L}$ be a class of languages. For a language $L \in \mathcal{L}$ and a set $D \subseteq L$, *the lower approximation* and *the upper approximation* of L *determined by* D (with respect to $\mathcal{L}$) are defined as follows:

$$\mathrm{LA}^{\mathcal{L}}(L, D) = \bigcap\{L' \in \mathcal{L} : D \subseteq L'\};\ \mathrm{UA}^{\mathcal{L}}(L, D) = \bigcup\{L' \in \mathcal{L} : D \subseteq L'\}.$$

These definitions seem very natural. If we merely know that expressions from D belong to L, and $L \in \mathcal{L}$, then precisely the expressions from $\mathrm{LA}^{\mathcal{L}}(L, D)$ necessarily belong to L on the basis of our knowledge, and precisely the expressions from $\mathrm{UA}^{\mathcal{L}}(L, D)$ possibly belong to L on the basis of our knowledge. We write $\mathrm{LA}^{\mathcal{G}}(L, D)$ for $\mathrm{LA}^{L(\mathcal{G})}(L, D)$, and similarly for UA.

The learning algorithms, described above, compute neither $\mathrm{LA}^{\mathcal{G}}(L, D)$, nor $\mathrm{UA}^{\mathcal{G}}(L, D)$, if $D \subseteq L$ is the set of input data. If G_i is the first grammar from (G_n), compatible with D, then $\mathrm{LA}^{\mathcal{G}}(L, D) \subseteq L(G_i) \subseteq \mathrm{UA}^{\mathcal{G}}(L, D)$, but these inclusions do not justify any new positive or negative inference about the extent of L which is not transparent in the input data. Of course, when the algorithm finds the proper grammar for L, the grammar produces the lower approximation of L determined by the input data. The problem is that the learner usually does not know that the proper grammar has just been found; she cannot exclude that larger data will enforce the change of the hypothesis. Only for very special cases, e.g. if one learns from positive data, and $L(G_i)$ has been proved to be maximal in $L(\mathcal{G})$, one can conclude that the learning procedure has been finished.

4. Learning by unification

We show below that algorithms based on unification, proposed in [6, 11] and further elaborated by Kanazawa [18, 19], Marciniec [24, 25] and others, do compute both the lower and the upper approximation of the searched language.

First, we demonstrate these algorithms for classical categorial grammars (learning from structures). Let us consider the grammar system (Ω, E, L) such that Ω is the class of classical categorial grammars (with a fixed lexicon Σ), $E = F(\Sigma)$, and $L(G) = L_f(G)$, for $G \in \Omega$. We assume that all grammars in Ω have the same principal type S. Since Σ and S are fixed, a grammar G can be identified with I_G.

First, we recall a unification-based learning procedure from [11]. Let $D \subseteq F(\Sigma)$ be a nonempty, finite set. We define a grammar $\mathrm{GF}(D)$, called *the general form of a grammar* for D. S is treated as a constant, whereas other atomic 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 $A\backslash B$,

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

$\mathrm{GF}(D)$ contains all assignments $v \mapsto A$ obtained in this way, for $v \in \Sigma$; the principal type of $\mathrm{GF}(D)$ is S.

Let us consider an example. Let D consist of structures (Joan works)$_2$ and (Joan (works hardly)$_2$)$_2$. One assigns type S to these structures and the types: x to 'Joan' (in the first structure), y to 'Joan' (in the second structure), z to 'works' (in the second structure). By (fl), one assigns: $x\backslash$S to 'works' (in the first structure), $y\backslash$S to (works hardly)$_2$, and $z\backslash(y\backslash$S) to 'hardly'. Accordingly, $\mathrm{GF}(D)$ yields the assignment:

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

It is easy to show that $L_f(\mathrm{GF}(D)) = D$ [11]. For our purposes, $\mathrm{GF}(D)$ is the input of a further procedure of unification.

A substitution is a mapping from variables to types; it is naturally extended to a mapping from types to types. If G is a grammar, and σ is a substitution, then $G\sigma$ is a (unique) grammar H such that $H : v \mapsto A\sigma$, 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 $I_G \subseteq I_{G'}$. If $G \subseteq G'$ then $L_f(G) \subseteq L_f(G')$. The following lemma has been proved in [11].

Lemma 4.1. Let $D \subseteq F(\Sigma)$ be nonempty and finite. Then, for every $G \in \Omega$, $D \subseteq L_f(G)$ iff there exists a substitution σ such that $GF(D)\sigma \subseteq G$.

The 'if' part is easy: $D \subseteq L_f(\mathrm{GF}(D)) \subseteq L_f(\mathrm{GF}(D)\sigma) \subseteq L_f(G)$. For the 'only if' part, observe that any grammar G such that $D \subseteq L_f(G)$ assigns some types to all substructures of the structures from D. More strictly, for any $X \in D$, we fix a derivation of $X \mapsto_G \mathrm{S}$. We define a substitution σ in the following way: if x is the variable assigned to Y by the construction of $\mathrm{GF}(D)$, Y is a substructure of X, and $Y \mapsto_G A$ is a part of the fixed derivation of $X \mapsto_G \mathrm{S}$, then $x\sigma = A$. Since (fl), (fr) reverse the rules defining $\mapsto_G$, one can show $\mathrm{GF}(D)\sigma \subseteq G$.

We employ standard notions of the theory of unification in a slightly modified version.

Let $\mathcal{T}$ be a nonempty, finite family of finite sets of types. A substitution σ is called *a unifier* of $\mathcal{T}$, if, for any $T \in \mathcal{T}$ and all $A, B \in T$, we have $A\sigma = B\sigma$ (so, σ unifies each set $T \in \mathcal{T}$ but not necessarily the join of these sets). If a unifier of $\mathcal{T}$ exists, then $\mathcal{T}$ is said to be *unifiable*. σ is called *a most general unifier* (m.g.u.) of $\mathcal{T}$, if it is a unifier of $\mathcal{T}$ and, for any unifier α of $\mathcal{T}$, there is a substitution β, such that $\alpha = \sigma\beta$ (in our notation, $A(\sigma\beta) = (A\sigma)\beta$). A standard algorithm of unification (see e.g. [23]) can easily be adapted to yield an m.g.u. of $\mathcal{T}$, if $\mathcal{T}$ is unifiable, or to reply NO, if $\mathcal{T}$ is not unifiable.

For a categorial grammar G with the initial type assignment I_G, we define a family: $\mathcal{T}(G) = \{I_G(v) : v \in \Sigma\}$. For a nonempty, finite set $D \subseteq F(\Sigma)$, we define $\mathcal{T}(D) = \mathcal{T}(\mathrm{GF}(D))$. Let $\mathcal{G}_k$ denote the class of $k-$valued categorial grammars (on Σ). In [6, 11], the following fact has been proved.

Proposition 4.1. Let $D \subseteq F(\Sigma)$ be nonempty and finite. There exists a $1-$valued categorial grammar G such that $D \subseteq L_f(G)$ if and only if $\mathcal{T}(D)$ is unifiable. If σ is an m.g.u. of $\mathcal{T}(D)$, then $L_f(\mathrm{GF}(D)\sigma)$ is the smallest f-language L in the class $\{L \in L_f(\mathcal{G}_1) : D \subseteq L\}$.

The proof of proposition 1 uses lemma 1. Since an m.g.u. of $\mathcal{T}$ is unique up to variants, then the resulting grammar $\mathrm{GF}(D)\sigma$ is unique up to isomorphism (renaming of variables). If $\mathcal{T}(D)$ is unifiable, then one defines $\mathrm{RG}(D) = \mathrm{GF}(D)\sigma$, where σ is an m.g.u. of $\mathcal{T}(D)$. The grammar $\mathrm{RG}(D)$ is called *the rigid (categorial) grammar determined by* D. By proposition 1, $L_f(\mathrm{RG}(D))$ is the smallest f-language in the class of all f-languages of rigid (1-valued) categorial grammars which contain D.

We return to the example at the beginning of this section. $\mathcal{T}(D)$ consists of the sets: $\{x, y\}$, $\{z, x\backslash \mathrm{S}\}$, $\{z\backslash(y\backslash \mathrm{S})\}$. It is unifiable, and its m.g.u. σ is given by: $y\sigma = x$, $z\sigma = x\backslash \mathrm{S}$. $\mathrm{RG}(D)$ yields the assignment:

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

The class $L_f(\mathcal{G}_1)$ (with a fixed Σ) admits no infinite ascending chains [18, 19]. This class enriched with the total f-language $F(\Sigma)$ is a closure system [10], so it has finite elasticity. The function $\varphi((X_0, \ldots, X_n)) = \mathrm{RG}(\{X_0, \ldots, X_n\})$ learns $\mathcal{G}_1$ [18] (caution: isomorphic grammars are treated as equal). To avoid the latter assumption one can modify the definition of φ:

(φ1) $\varphi((X_0) = \mathrm{RG}(\{X_0\})$.

(φ2) $\varphi((X_0,\ldots,X_{n+1})) = \varphi((X_0,\ldots,X_n))$, if X_{n+1} belongs to the f-language of $\varphi((X_0,\ldots,X_n))$, and it equals RG($\{X_0,\ldots,X_{n+1}\}$), otherwise.

A binary relation R is said to be *finitely valued* if, for all x, the set $\{y : (x,y) \in R\}$ is finite. Let $R \subseteq E \times E'$. For $L \subseteq E'$, one defines $R^{-1}[L] = \{e \in E : (\exists e' \in E')(e,e') \in R\}$. For a class $\mathcal{L}$, one defines $R^{-1}[\mathcal{L}] = \{R^{-1}[L] : L \in \mathcal{L}\}$. Using the König lemma, Kanazawa [18, 19] proves the following theorem on finite elasticity: if $R \subseteq E \times E'$ is finitely valued, and $\mathcal{L}$ has finite elasticity, $\mathcal{L}$ is a class of subsets of E', then $R^{-1}[\mathcal{L}]$ has finite elasticity.

As a consequence, each class $L_f(\mathcal{G}_k)$, $k \geq 1$, has finite elasticity [18, 19]. Let $\Sigma = \{v_1,\ldots,v_n\}$. Each v_i is replaced by k new symbols v_i^j, $j = 1,\ldots,k$. Let Σ' consist of all new symbols v_i^j. A relation $R \subseteq F(\Sigma) \times F(\Sigma')$ is defined as follows: $(X,X') \in R$ iff X arises from X' by replacing each v_i^j by v_i. Clearly, R is finitely valued. One easily shows $L_f(\mathcal{G}_k) = R^{-1}[L_f(\mathcal{G}_1)]$ (here $\mathcal{G}_k$ consists of grammars on Σ, while $\mathcal{G}_1$ consists of grammars on Σ').

Consequently, $\mathcal{G}_k$ is learnable (from f-structures). A concrete learning algorithm, given in [19], uses the following construction.

Let $\mathcal{T}$ be a family of sets. A family $\mathcal{T}'$ of nonempty subsets of sets from $\mathcal{T}$ is called *a partition* of $\mathcal{T}$ if, for any $T \in \mathcal{T}$, the family $\{T' \in \mathcal{T}' : T' \subseteq T\}$ is a partition of T in the standard sense. If $\mathcal{T}'$ divides each $T \in \mathcal{T}$ in at most k parts, then $\mathcal{T}'$ is called a $k-$partition of $\mathcal{T}$. One defines the set $\mathrm{VG}_k(D)$ as the set of all grammars $\mathrm{GF}(D)\sigma$ such that σ is an m.g.u. of some $k-$partition of $\mathcal{T}(D)$. A partial computable function ψ (the domain of ψ is recursive) is defined as follows: $\psi(D)$ picks a grammar $G \in \mathrm{VG}_k(D)$ such that $L_f(G)$ is minimal among all f-languages $L_f(G')$, $G' \in \mathrm{VG}_k(D)$, if $\mathrm{VG}_k(D)$ is nonempty. Then, a learning function φ for $\mathcal{G}_k$ is defined by (φ1), (φ2) with replacing RG by ψ.

In a similar way one shows that $\mathcal{G}_k$ is learnable from p-structures and from strings [19]. The crucial observation is that the class of p-languages (resp. languages) of grammars from $\mathcal{G}_k$ has finite elasticity, which follows from the finite elasticity of the class of f-languages of these grammars and the theorem on finite elasticity.

5. Main results

In this section we show that unification-based learning procedures can compute both $\mathrm{LA}^{\mathcal{G}}(L,D)$ and $\mathrm{UA}^{\mathcal{G}}(L,D)$, whenever the class $\mathcal{G}$ satisfies (G2) and some additional constraints.

A substitution α induces an equivalence relation $\equiv_\alpha$ on types: $A \equiv_\alpha B$ iff $A\alpha = B\alpha$. Each equivalence relation $\equiv$ on types partitions any set T, of types, in equivalence classes (restricted to T). We define $T/\equiv = \{[A]_\equiv \cap T : A \in T\}$. For a family $\mathcal{T}$ of sets of types, we define:

$$\mathcal{T}/\equiv = T_1/\equiv \cup \cdots \cup T_n/\equiv, \text{ for } \mathcal{T} = \{T_1,\ldots,T_n\}.$$

This family is called *the partition* of $\mathcal{T}$ *determined* by $\equiv$.

Let Ω be the class of categorial grammars on Σ. We assume that $\mathcal{G} \subseteq \Omega$ satisfies (G2) and the following conditions:

(G1') $\mathcal{G}$ is recursive and closed under isomorphism (renaming of variables),

(G3) if $G \subseteq G'$ and $G' \in \mathcal{G}$ then $G \in \mathcal{G}$,

(G4) for any finite set $D \subseteq F(\Sigma)$ and any substitution α, if $\mathrm{GF}(D)\alpha \in \mathcal{G}$ then $\mathrm{GF}(D)\sigma \in \mathcal{G}$, for any m.g.u. σ of $\mathcal{T}_D/\equiv_\alpha$.

(G3) and (G4) are generalizations of some conditions, considered in [24]. Notice that α is a unifier of $\mathcal{T}/\equiv_\alpha$, whence an m.g.u. of the latter family always exists. No result of this section depends on (G2), but this constraint is needed for standard learning algorithms (also see conditions (φ1),(φ2) in section 4).

Each class $\mathcal{G}_k$, $k \geq 1$, satisfies these conditions. It is easy to prove (G1'), (G2), (G3). We show (G4). Assume $\mathrm{GF}(D)\alpha \in \mathcal{G}_k$. Then, $\mathcal{T}(D)/\equiv_\alpha$ is a $k-$partition of $\mathcal{T}(D)$ (see section 4). If σ is an m.g.u. of this $k-$partition, then $\mathrm{GF}(D)\sigma \in \mathcal{G}_k$.

Classes of grammars, satisfying (G1')-(G4), are closed under intersection (of classes).So, one can restrict the classes $\mathcal{G}_k$ by additional constraints.

The order of type A ($o(A)$) is defined as follows [7]:

$$o(p) = 0, \text{ for variables } p;\ o(A\backslash B) = o(B/A) = \max(o(B), o(A)+1).$$

The order of G ($o(G)$) is the maximal order of types in I_G. The class of all $G \in \Omega$ such that $o(G) \leq n$, for a fixed $n \geq 0$, satisfies (G1')-(G4) (for (G4), use the inequality $o(A) \leq o(A\alpha)$, for all types A and substitutions α). Then, the class of $k-$valued grammars of order not greater than n satisfies our conditions. For $n \geq 1$, the class of categorial grammars of order not greater than n is not learnable, since it yields all f-languages of categorial grammars [7].

Let $\mathcal{G} \subseteq \Omega$ and $D \subseteq F(\Sigma)$ be nonempty and finite. We define a set of grammars $\mathrm{OUT}^{\mathcal{G}}(D)$ as the set of all grammars $\mathrm{GF}(D)\sigma$ such that σ is an m.g.u. of some partition of $\mathcal{T}(D)$ and $\mathrm{GF}(D)\sigma \in \mathcal{G}$. Up to isomorphism, it is a finite set of grammars.

Lemma 5.1. If $\mathcal{G}$ satisfies (G1'), then $\mathrm{OUT}^{\mathcal{G}}(D)$ is effectively computable from D.

Theorem 5.1. Let $\mathcal{G}$ satisfy (G3), (G4). For any $L \in L_f(\mathcal{G})$ and any nonempty, finite $D \subseteq L$, there hold the equalities:

$$\mathrm{LA}^{\mathcal{G}}(L, D) = \bigcap\{L_f(G) : G \in \mathrm{OUT}^{\mathcal{G}}(D)\},$$

$$\mathrm{UA}^{\mathcal{G}}(L, D) = \{X \in F(\Sigma) : \mathrm{OUT}^{\mathcal{G}}(D \cup \{X\}) \neq \emptyset\}.$$

Proof:
We prove the first equality. By lemma 1, $D \subseteq L_f(G)$, for any $G \in \mathrm{OUT}^{\mathcal{G}}(D)$, whence the inclusion $\subseteq$ holds. We prove the converse inclusion. Let $X \notin \mathrm{LA}^{\mathcal{G}}(L, D)$. Then, $X \notin L_f(G)$, for some $G \in \mathcal{G}$ such that $D \subseteq L_f(G)$. By lemma 1, there is a substitution α such that $\mathrm{GF}(D)\alpha \subseteq G$. By (G3), $\mathrm{GF}(D)\alpha \in \mathcal{G}$. Let σ be an m.g.u. of $\mathcal{T}(D)/\equiv_\alpha$. By (G4), $\mathrm{GF}(D)\sigma \in \mathcal{G}$, and consequently, the grammar belongs to $\mathrm{OUT}^{\mathcal{G}}(D)$. There exists a substitution β such that $\alpha = \sigma\beta$. Consequently, $L_f(\mathrm{GF}(D)\sigma) \subseteq L_f(\mathrm{GF}(D)\alpha) \subseteq L_f(G)$, whence $X \notin L_f(\mathrm{GF}(D)\sigma)$, which finishes this part of proof.

We prove the second equality. Let $X \in \mathrm{UA}^{\mathcal{G}}(L, D)$. Then, $X \in L_f(G)$, for some $G \in \mathcal{G}$ such that $D \subseteq L_f(G)$. By lemma 1, there is a substitution α such that $\mathrm{GF}(D \cup \{X\})\alpha \subseteq G$. By (G3), $\mathrm{GF}(D \cup \{X\})\alpha \in \mathcal{G}$, and by (G4), $\mathrm{GF}(D \cup \{X\})\sigma \in \mathcal{G}$, for any m.g.u. σ of $\mathcal{T}(D \cup \{X\})/\equiv_\alpha$. So, $\mathrm{OUT}^{\mathcal{G}}(D \cup \{X\}) \neq \emptyset$. The converse inclusion is obvious. □

If $\mathcal{G}$ satisfies (G1'), (G3) and (G4), then both $\mathrm{LA}^{\mathcal{G}}(L, D)$ and $\mathrm{UA}^{\mathcal{G}}(L, D)$ are computable from $D \subseteq L$. Since f-languages of categorial grammars are effectively closed under finite meets, then $\mathrm{LA}^{\mathcal{G}}(L, D)$ can be presented as $L_f(G)$, for some categorial grammar G. Caution: for $\mathcal{G} = \mathcal{G}_k$, this grammar G need not be $k-$valued. $\mathrm{UA}^{\mathcal{G}}(L, D)$ is a recursive f-language; it can be presented by means of a Turing machine.

There appears a danger of triviality. If $\mathcal{G}$ is the class of all categorial grammars or the class of all categorial grammars of order not greater than n, then $\mathrm{LA}^{\mathcal{G}}(L, D) = D$, for all $L \in L_f(\mathcal{G})$ and all nonempty, finite $D \subseteq L$. It is not the case if $L(\mathcal{G})$ has finite elasticity. The next proposition uses the general terminology and notation of section 3.

Proposition 5.1. Let $L(\mathcal{G})$ satisfy the condition (D) from section 3. Let $L \in L(\mathcal{G})$, $L = \{s_n\}_{n\in\omega}$. Then, there exists $n \in \omega$ such that $\mathrm{LA}^{\mathcal{G}}(L, \{s_0, \ldots, s_m\}) = L$, for all $m \geq n$.

Proof:
By (D), there exists a finite set $D_L \subseteq L$ such that L is the smallest language in $L' \in L(\mathcal{G})$ such that $D_L \subseteq L'$. There exists $n \in \omega$ such that $D_L \subseteq \{s_0, \ldots, s_n\}$. The least integer n, having this property, fulfills the demands of the proposition. □

We consider an example. Let D consist of three English sentences (in the form of f-structures):

1. (Kazimierz teaches)$_2$,
2. (Kazimierz (teaches (a logician)$_1$)$_1$)$_2$,
3. ((a logician)$_1$(teaches Kazimierz)$_1$)$_2$.

$\mathrm{GF}(D)$ yields the following type assignment:

$$\text{Kazimierz} \mapsto x, y, z, \text{ teaches} \mapsto x\backslash\mathrm{S},\ (x\backslash\mathrm{S})/u,\ (w\backslash\mathrm{S})/z,$$

$$\text{logician} \mapsto p, q, \text{ a} \mapsto u/p,\ w/q.$$

$\mathcal{T}(D)$ is not unifiable (see the first two types of 'teaches'). Let Σ be the set of four words, appearing in these sentences. Let $\mathcal{G}$ be the set of categorial grammars on Σ which assign at most two types to 'teaches' and at most one type to each of the remaining words. Since $\mathcal{G} \subseteq \mathcal{G}_2$, then $L_f(\mathcal{G})$ has finite elasticity (this property is preserved by subclasses). By proposition 2, our algorithm, computing $\mathrm{LA}^{\mathcal{G}}(L, D)$ acts nontrivially on $\mathcal{G}$. For the set D, as above, $\mathrm{OUT}^{\mathcal{G}}(D)$ consists of a unique grammar H:

$$\text{Kazimierz} \mapsto x, \text{ teaches} \mapsto x\backslash\mathrm{S},\ (x\backslash\mathrm{S})/x,$$

$$\text{logician} \mapsto p, \text{ a} \mapsto x/p.$$

One can treat x as the type of noun phrase and p as the type of common noun. Then, $x\backslash\mathrm{S}$ is the type of intransitive verb and $(x\backslash\mathrm{S})/x$ of transitive verb. For any $L \in L_f(\mathcal{G})$ such that $D \subseteq L$, we have $\mathrm{LA}^{\mathcal{G}}(L, D) = L_f(H)$; this set consists of six f-structures: the three f-structures from D, ((a logician)$_1$ teaches)$_2$, (Kazimierz (teaches Kazimierz)$_1$)$_2$, ((a logician)$_1$(teaches (a logician)$_1$)$_2$. These structures must belong to every f-language $L \in L_f(\mathcal{G})$ such that $D \subseteq L$. It can also be shown that (teaches Kazimierz)$_1$ does not belong to $\mathrm{UA}^{\mathcal{G}}(L, D)$. Let X denote this f-structure. $\mathrm{GF}(D \cup \{X\})$ assigns a new

type S$/r$ to 'teaches' (type r to 'Kazimierz'). Now, we have four types of 'teaches', and three of them are pairwise not unifiable. Consequently, $\mathrm{OUT}^{\mathcal{G}}(D \cup \{X\}) = \emptyset$.

Let us notice that (a logician)$_1$ belongs to $\mathrm{UA}^{\mathcal{G}}(L, D)$, according to the above example (one can unify x and S). This is a linguistic nonsense. One can eliminate this possibility by introducing a negative constraint (a logician)$_2 \not\rightarrow$S. Then, one considers classes of categorial grammars, fulfilling this constraint. In [24, 14], it has been shown that the unification-based learning algorithms can be relativized to classes of grammars, restricted by various negative constraints. Also, the results of the present paper remain true, if one admits positive and negative data, say of the form $D = (D^p, D^n)$, where D^p, D^n are disjoint, finite sets. We write $D \sqsubseteq L$ if $D^p \subseteq L$ and $D^n \cap L = \emptyset$. This relation can be used instead of $D \subseteq L$, and all definitions and results can be modified in this style. We omit details. It is noteworthy that computing LA and UA with the aid of negative data requires similar assumptions on $\mathcal{G}$ as for the case of positive data only. So, it is not the case that every class of grammars, fulfilling (G1) and (G2), admits an effective computation of LA and UA by unification (this contrasts with general results on learnability, discussed in section 3).

We naturally come to an idea of approximate learning. *An approximate learning function* for $\mathcal{G}$ is defined as a function φ, which to any nonempty, finite set $D \subseteq E$ assigns a finite set of grammars from $\mathcal{G}$, satisfying the conditions:

(AL1) for any $G \in \varphi(D)$, $D \subseteq L(G)$,

(AL2) for every $L \in L(\mathcal{G})$, if $D \subseteq L$ then there exists $G \in \varphi(D)$, fulfilling $L(G) \subseteq L$.

If $\mathcal{G}$ fulfills (G3), (G4), then the function $\varphi(D) = \mathrm{OUT}^{\mathcal{G}}(D)$ satisfies (AL1), (AL2) (see the proof of theorem 1). If φ is an approximate learning function for $\mathcal{G}$, then, for any $L \in L(\mathcal{G})$ and all $D \subseteq L$, we have:

$$\mathrm{LA}^{\mathcal{G}}(L, D) = \bigcap\{L(G) : G \in \varphi(D)\}.$$

We say that an approximate learning function φ for $\mathcal{G}$ *learns* $\mathcal{G}$, if, for any $L \in L(\mathcal{G})$ and any text $(s_n)_{n\in\omega}$ for L, $\bigcap\{L(G) : G \in \varphi(\{s_0, \ldots, s_n\})\} = L$, for all but finitely many $n \in \omega$.

Theorem 5.2. Let φ be an approximate learning function for $\mathcal{G}$. Then, φ learns $\mathcal{G}$ if and only if $L(\mathcal{G})$ satisfies (D).

Proof:
Assume that φ learns $\mathcal{G}$. Then, $\mathrm{LA}^{\mathcal{G}}(L, \{s_0, \ldots, s_n\}) = L$, for any $L \in L(\mathcal{G})$, any text (s_n) for L, and for some n, depending on the text. The set $D_L = \{s_0, \ldots, s_n\}$ fulfills the demands of (D). Assume that $L(\mathcal{G})$ satisfies (D). Fix $L \in L(\mathcal{G})$ and a text (s_n) for L. Let D_L be as in (D). There exists $n \in \omega$ such that $D_L \subseteq \{s_0, \ldots, s_n\}$. Clearly, $\mathrm{LA}^{\mathcal{G}}(L, \{s_0, \ldots, s_m\}) = L$, for all $m \geq n$. □

An analogue of theorem 1 is true for string languages. We must modify the construction. If $D \subseteq \Sigma^+$ is nonempty and finite, then we consider all possible sets $D' \subseteq F(\Sigma)$ arising from D after one has defined some f-structure on each string from D (this is a finite family of sets of f-structures). We define $\mathrm{OUT}^{\mathcal{G}}(D)$ as the join of all sets $\mathrm{OUT}^{\mathcal{G}}(D')$, the latter being defined as above.

Theorem 5.3. Let $\mathcal{G}$ satisfy (G3), (G4). Then, for any $L \in L(\mathcal{G})$ and any nonempty, finite $D \subseteq L$, there hold the equalities:

$$\mathrm{LA}^{\mathcal{G}}(L, D) = \bigcap\{L(G) : G \in \mathrm{OUT}^{\mathcal{G}}(D)\},$$

$$\mathrm{UA}^{\mathcal{G}}(L, D) = \{a \in \Sigma^{+} : \mathrm{OUT}^{\mathcal{G}}(D \cup \{a\}) \neq \emptyset\}.$$

Proof:
The proof is similar to the proof of theorem 1. □

Assuming (G1'), one can effectively compute $\mathrm{OUT}^{\mathcal{G}}(D)$ from D, and consequently, LA and UA. String languages of categorial grammars are not closed under meet, whence the values of LA are recursive languages but not necessarily CF-languages. (It would be interesting to find some examples.) Learning from p-structures can be handled in a similar way (now, LA yields p-languages of categorial grammars).

Unification-based learning is not restricted to categorial grammars. For CF-grammars, we can use similar methods. We briefly consider learning from p-structures. Ω denotes the set of all CF-grammars with the terminal alphabet Σ. E denotes the set of all p-structures on Σ.

Let $D \subseteq E$ be nonempty and finite. We assign S to all structures from D and different variables to proper substructures of these structures. $\mathrm{GF}(D)$ is a CF-grammar whose production rules are determined by these labeled structures.

For example, if D contains (Joan (works hardly)), then one assigns x to 'Joan', y to (works hardly), z to 'works', and u to 'hardly'. $\mathrm{GF}(D)$ contains the productions:

$$S \mapsto xy,\ x \mapsto \text{Joan},\ y \mapsto zu,\ z \mapsto \text{works},\ u \mapsto \text{hardly}.$$

In general, $\mathrm{GF}(D)$ contains all production rules, obtained in this way from p-structures in D. The set of these rules is denoted by $R(D)$.

Substitutions are restricted to variable-to-variable substitutions. $\mathrm{GF}(D)\sigma$ arises from $\mathrm{GF}(D)$ by replacing each variable x by $x\sigma$ (in rules from $R(D)$). An analogue of lemma 1 is true. Partitions are defined on $R(D)$. A substitution α determines an equivalence relation $\equiv_\alpha$ on $R(D)$: two rules are equivalent iff α makes them identical. One naturally defines a unifier and an m.g.u. of a partition of $R(D)$. Then, conditions (G1), (G1'), (G2)-(G4) can be adapted to the case of CF-grammars in a straightforward way, and similarly for the construction of $\mathrm{OUT}^{\mathcal{G}}(D)$. All results of this section and analogues of learnability results from section 4 can easily be proved. In particular, the class of CF-grammars on Σ, admitting at most k production rules, is learnable both from p-structures and from strings, and one can compute LA and UA for this class.

Context-sensitive grammars (CS-grammars) admit productions of the form $bAc \mapsto bac$, where A is a nonterminal, a, b, c are strings of symbols, $a \neq \epsilon$. CS-grammars determine p-structures on derivable strings, but p-structures (even labeled by nonterminals) do not uniquely determine derivations. The method, described above for CF-grammars, can be extended to CS-grammars, if the number of production rules is bounded.

It is known that categorial grammars based on the (associative) Lambek calculus are not learnable from strings, nor structures [3]. On the other hand, approximate learning can lead to interesting results also for grammars based on different variants of the Lambek calculus. We leave this problem for further research.

References

[1] Ajdukiewicz, K.: Die syntaktische Konnexitaet, *Studia Philosophica*, **1**, 1935, 1.

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

[3] Bechet, D., Foret, A.: $k-$valued non-associative Lambek grammars are learnable from generalized functor-argument structures, *Theoretical Computer Science*, **355**(2), 2006, 139–152.

[4] van Benthem, J.: *Language in Action. Categories, Lambdas and Dynamic Logic*, Studies in Logic and The Foundations of Mathematics, North-Holland, Amsterdam, 1991.

[5] van Benthem, J., ter Meulen, A., Eds.: *Handbook of Logic and Language*, Elsevier Science B. V., 1997.

[6] Buszkowski, W.: Discovery Procedures for Categorial Grammars, in: *Categories, Polymorphism and Unification* (E. Klein, J. van Benthem, Eds.), Universiteit van Amsterdam, Amsterdam, 1987, 36–64.

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

[8] Buszkowski, W.: Mathematical Linguistics and Proof Theory, in: van Benthem and ter Meulen [5], 683–736.

[9] Buszkowski, W.: Algebraic structures for categorial grammars, *Theoretical Computer Science*, **199**, 1998, 5–24.

[10] Buszkowski, W.: Type logics in grammar, in: *Trends in Logic. 50 years of Studia Logica* (V. F. Hendricks, J. Malinowski, Eds.), Kluwer, Dordrecht, 2003, 337–382.

[11] Buszkowski, W., Penn, G.: Categorial Grammars Determined from Linguistic Data by Unification, *Studia Logica*, **XLIX**(4), 1990, 431–454.

[12] Demri, S., Orłowska, E.: *Incomplete Information: Structure, Inference, Complexity*, EATCS Monographs in Theoretical Computer Science, Springer, 2002.

[13] Doherty, P., Łukaszewicz, W., Skowron, A., Szałas, A.: *Knowledge Representation Techniques. A Rough Set Approach*, Studies in Fuzziness and Soft Computing, Springer, 2006.

[14] Dziemidowicz, B.: Optimal Unification and Learning Algorithms for Categorial Grammars, *Fundamenta Informaticae*, **49**, 2002, 297–308.

[15] Fulop, S. A.: Semantic Bootstrapping of Type-Logical Grammar, *Journal of Logic, Language and Information*, **14**, 2005, 49–86.

[16] Gecség, F., Steinby, M.: *Tree Automata*, Akademiai Kiadó, Budapest, 1984.

[17] Gold, E. M.: Language Identification in the Limit, *Information and Control*, **10**, 1967, 447–474.

[18] Kanazawa, M.: Identification in the Limit of Categorial Grammars, *Journal of Logic, Language and Information*, **5**(2), 1996, 115–155.

[19] Kanazawa, M.: *Learnable Classes of Categorial Grammars*, Studies in Logic, Language and Information, CSLI Publications & FoLLI, Stanford, California, 1998.

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

[21] Lambek, J.: The mathematics of sentence structure, *American Mathematical Monthly*, **65**, 1958, 154–170.

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

[23] Lloyd, J. W.: *Foundations of Logic Programming*, Springer-Verlag, Berlin, 1987.

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

[25] Marciniec, J.: Infinite Set Unification with Application to Categorial Grammar, *Studia Logica*, **LVIII**(3), 1997, 339–355.

[26] Marciniec, J.: Optimal Unification of Infinite Sets of Types, *Fundamenta Informaticae*, **62**(3,4), 2004, 395–407.

[27] Moortgat, M.: Categorial Type Logics, in: van Benthem and ter Meulen [5], 93–177.

[28] Moortgat, M.: Structural Equations in Language Learning, in: *Logical Aspects of Computational Linguistics* (P. de Groote, G. Morrill, C. Retore, Eds.), vol. 2099 of *Lecture Notes in Artificial Intelligence*, Springer, 2001, 1–16.

[29] Osherson, D., de Jongh, D., Martin, E., Weinstein, S.: Formal Learning Theory, in: van Benthem and ter Meulen [5], 737–775.

[30] Pawlak, Z.: Rough sets, *International Journal of Information and Computer Science*, **11**(5), 1982, 341–356.

[31] Pawlak, Z.: *Rough Sets. Theoretical Aspects of Reasoning about Data*, Kluwer, Dordrecht, 1991.

[32] Pawlak, Z., Skowron, A.: Rudiments of Rough Sets, *Information Sciences*, to appear.

[33] Polkowski, L. T.: *Rough Sets. Mathematical Foundations*, Advances in Soft Computing, Springer, 2002.

[34] Shinohara, T.: Inductive Inference of Monotonic Data, in: *Algorithmic Learning Theory* (S. Arikawa, S. Goto, S. Ohsuga, T. Yokomori, Eds.), Springer, 1990, 339–351.

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