ON THREE TYPES OF UNAMBIGUITY OF CONTEXT-FREE LANGUAGE

A.Ja.Dikovskiĭ and Larisa S.Modina
Institute of Mathematics,
Novosibirsk 630090, USSR

Introduction.

As it is readily seen, the wording of the problem of unambiguous description of a language depends not only on involved grammars, but also on types of structures in terms of which these grammars characterise their sentences. Up to last years the problem of unambiguous grammatical description was considered only with respect to context-free grammars, whose structural descriptions are constituent-structure trees. In the monography [1] this problem was raised with respect to dependency grammars whose generative capacities are the same as those of context-free grammars, but whose structural descriptions (dependency trees) are different from constituent-structure trees. In this wording the problem got its final solution in papers [2-4] even for a more general notion of dependency grammar than in [1]. In our paper the very same problem is considered and resolved with respect to the class of the finite state Δ-systems, introduced in [5], and as it is shown there, having the same generative capacity as context-free grammars. This model is a prime formalization of the surface-syntactic component of linguistic systems of the type "Meaning$\Leftrightarrow$Text" (see monography [6]). Finite-state Δ-systems characterise their sentences in terms of pure dependency trees widely used as the tool of syntactic description of sentences in natural languages. We compare the capacities of unambiguous description of context-free languages in terms of each of the three structures mentioned above. As it turns out, the classes of languages generated by unambiguous dependency grammars and unambiguous finite-state Δ-systems are incomparable and both strictly broader than the class of unambiguous context-free languages. Nevertheless their union does not coincide with the class of all context-free languages.

For the reason of space economy we prove two first theorems only (however most relevant technics is involved in their proofs). For the same reason an informal discussion of the main concepts is abridged and put in the appendix. Our paper is closely interrelated to still

unpublished papers [2,4] which are abstracted in [3]. Nevertheless it is selfcontained. Its complete version is submitted for publication in the Transactions of Kalinin State University.

Basic definitions.

Let Σ and M be some disjoint finite alphabets (of word-forms and of types of surface structure relations in natural language interpretation).

Definition. A dependency (d-) tree in Σ, M is a pair $\psi = \langle T, \pi \rangle$, where T is some finite oriented tree with nodes labelled by symbols in Σ and arrows - by symbols in M, and π is some linear ordering of all nodes of T. Let $v_1, \ldots, v_k$ be the π-ordered sequence of all nodes of T, and $x(v_i)$ be a label of v_i. Then we say that ψ is a dependency tree of $x = x(v_1)x(v_2)\ldots x(v_k)$ (notation $x = w(\psi)$).

Definition. Let T be a finite oriented tree with node labels in Σ and arrow labels in M. We call T a pure dependency (pd-) tree, if any two arrows in T leading from a node v to different and identically labelled nodes v_1, v_2, have different arrow labels. We say that T is a pure dependency tree of a string x if there is such a linear ordering π of all nodes of T that $\langle T, \pi \rangle$ is a dependency tree of x.

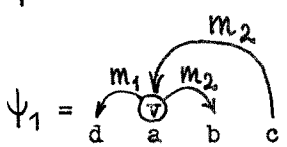
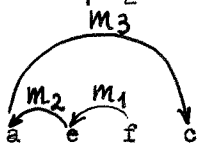
Notation. The set of all d-trees (pd-trees) in Σ and M is denoted by $D(\Sigma, M)$ ($\Delta(\Sigma, M)$). Subsets of $D(\Sigma, M)$ and of $\Delta(\Sigma, M)$ will be called d-languages and pd-languages respectively.

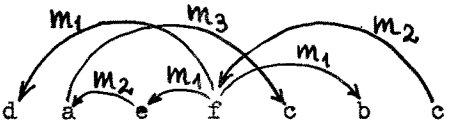
To provide background for our comparative study we recall some notions and notation from the papers [2-4,7].

Notation. 1. Let T be a finite oriented tree and v be a node of T, T(v) denotes the subtree of T which results from T by deletion of all nodes $v' \neq v$ not depending on v. (We call it a complete subtree of T generated by v.)

2. Let $\psi_1 = \langle T_1, \pi_1 \rangle$ and $\psi_2 = \langle T_2, \pi_2 \rangle$ be in $D(\Sigma, M)$, $X_1 = (v_1, \ldots, v_t, v, v_{t+1}, \ldots, v_n)$ the π_1-ordered sequence of all nodes of T_1, and $X_2 = (u_1, \ldots, u_m)$ the π_2-ordered sequence of all nodes of T_2. $A(\psi_1, v, \psi_2)$ denotes the d-tree $\langle T, \pi \rangle$, where 1) the set of all nodes of T is $X = (X_1 \cup X_2) - \{v\}$; 2) π orders X into the sequence $(v_1, \ldots, v_t, u_1, \ldots, u_m, v_{t+1}, \ldots, v_n)$; 3) v' and v'' are connected in T (in indicated direction) by an arrow of type m in M if either a) v', v'' are in $X_1 - \{v\}$

and connected in T_1 by the same arrow; or b) v',v'' are in X_2 and connected in T_2 by the same arrow; or c) v' is in $X_1-\{v\}$, v'' is the root of T_2 and v' and v are connected in T_1 by the same arrow; or d) v'' is in $X_1-\{v\}$, v' is the root of T_2 and v and v'' are connected by the same arrow in T_1; 4)corresponding nodes in T and T_1,T_2 have identical labels.

Example. Let  and  . Then

$A(\psi_1,v,\psi_2)=$  .

Definition [3,7]. A generalized dependency dendrogrammar (gdd-grammar) $\mathcal{T}=(V,M,W,I,R)$, where V,W and M are pairwise disjoint finite alphabets (of terminals, nonterminals and arrow labels), I in W (an axiom) and R is a finite system of productions of the form $B\to\varphi$, where B is in W and φ in $D(V\cup W,M)$.

Notation. For ψ_1,ψ_2 in $D(V\cup W,M)$ we write $\psi_1\models_{\mathcal{T}}\psi_2$ if there is a node v in ψ_1 labelled by B and a production $B\to\varphi$ in R such that $\psi_2=A(\psi_1,v,\varphi)$. The transitive closure of relation $\models_{\mathcal{T}}$ is denoted by $\vdash_{\mathcal{T}}$. $L_D(\mathcal{T})=\{\psi \text{ in } D(V,M) \mid I\vdash_{\mathcal{T}}\psi\}$ is a d-language generated by $\mathcal{T}$ and $L(\mathcal{T})=\{w(\psi)\mid \psi \text{ in } L_D(\mathcal{T})\}$ is a language generated by $\mathcal{T}$.

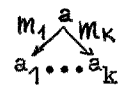
From these definitions immediately follows that the class of languages generated by gdd-grammars coincides with the class of all cf-languages.

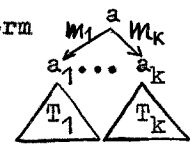
Definition [5]. A finite-state (fs-) Δ-grammar is a system G=(V,M, W,I,R), where V,M and W are finite pairwise disjoint alphabets (of terminals, arrow labels and nonterminals), I is in W (an axiom) and R is a finite set of productions of the form A→T, where A is in W and T is in $\Delta(V\cup W,M)$, all nonbottom nodes of T being labelled by terminals.

Notation. For T_1,T_2 in $\Delta(V\cup W,M)$ we write $T_1\models_G T_2$ if there is a production A→T in R and a bottom node v in T_1 labelled by A, such that T_2 results from T_1 by substitution of T in T_1 for v. The transitive closure of $\models_G$ we denote by $\vdash_G$. A Δ-language generated by G is the set $L_\Delta(G)=\{T \text{ in } \Delta(V,M) \mid I\vdash_G T\}$.

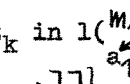
Fs-Δ-grammars form the simplest type of grammars generating Δ-languages. General definition of Δ-grammars may be found in [5]. One of basic notions in [5] is the notion of Δ-system, i.e. a two-level

formal system consisting of a Δ-grammar generating a class Δ of pd-trees and a so called linearization operator which maps pd-trees in Δ into strings they characterize. We consider in our paper the simplest type of Δ-systems - the so called finite-state Δ-systems.

Definition. Let V and M be finite disjoint alphabets. A local linearization operator in V and M is a function l which associates with every tree T in $\Delta(V,M)$ of the form  a nonempty set l(T) of strings $z_0 \dots z_p \dots z_k$ such that for some p, $0 \leq p \leq k$, $z_p = a$ and $z_0, \dots, z_{p-1}, z_{p+1}, \dots, z_k$ is a permutation of the sequence $(a_1,m_1),\dots,(a_k,m_k)$.

Definition. The linearization operator O_l related to a local linearization operator l is the function from $\Delta(V,M)$ to 2^{V^+} defined by induction as follows. a)For a one-node tree T with a label a in V $O_l(T) = \{a\}$. b)Let T be a pd-tree in $\Delta(V,M)$ of the form 

Then

$$O_l(T) = \left\{ y_0 \dots y_p \dots y_k \;\middle|\; (\exists z_0 \dots z_p \dots z_k \text{ in } l(\text{  }))\Big[z_p = y_p = a \;\&\; (\underset{0 \leq j \leq k}{\forall j \neq p}) \Big[z_j = (a_{i_j}, m_{i_j}) \;\&\; y_j \text{ is in } O_l(T_{i_j}) \Big] \Big] \right\}.$$

Definition. A finite-state (fs-) Δ-system is a pair (G,l), where G = (V,M,W,I,R) is a fs-Δ-grammar and l is a local linearization operator in V,M. The set $L(G,l) = \bigcup_{T \text{ in } L_\Delta(G)} O_l(T)$ is called a language specified by the system (G,l).

In the paper [5] it is proved in fact that the class of all languages specified by fs-Δ-systems coincides with the class of all cf-languages. This enables us to introduce in parallel to classical notions of unambiguity, ambiguity, degree of ambiguity of cf-language analogous notions in terms of gdd-grammars and fs-Δ-systems.

Definition [4]. Let $\mathcal{T}$ be a gdd-grammar and x in $L(\mathcal{T})$. $\Xi_D(\mathcal{T},x)$ denotes the number of d-trees ψ in $L_D(\mathcal{T})$ such that $w(\psi)=x$. A degree of d-ambiguity of $\mathcal{T}$ is the quantity $\Xi_D(\mathcal{T}) = \sup\{ \Xi_D(\mathcal{T},x) \mid x \text{ is in } L(\mathcal{T}) \}$ (if $\Xi_D(\mathcal{T})$ is not a number, we set $\Xi_D(\mathcal{T}) = \infty$). Gdd-grammar $\mathcal{T}$ is d-unambiguous if $\Xi_D(\mathcal{T}) \leq 1$, and d-ambiguous if $\Xi_D(\mathcal{T}) > 1$ (it is supposed that $n < \infty$ for all numbers n). A degree of d-ambiguity of a cf-language L is $\Xi_D(L) = \min\{ \Xi_D(\mathcal{T}) \mid L(\mathcal{T}) = L \}$. A cf-language L is d-unambiguous if $\Xi_D(L) \leq 1$, and d-ambiguous if $\Xi_D(L) > 1$.

Definition. Let (G,l) be a fs-Δ-system and x in L(G,l). $\Xi_\Delta((G,l),$

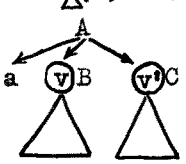
x) will denote the number of pd-trees T in $L_\Delta(G)$ such that x is in $O_1(T)$. A <u>degree of Δ-ambiguity</u> of (G,1) is the quantity $\Xi_\Delta(G,1) = \sup\{\Xi_\Delta((G,1),x) \mid x \text{ is in } L(G,1)\}$. Fs-Δ-system (G,1) is <u>Δ-unambiguous</u> if $\Xi_\Delta(G,1) \leq 1$, and <u>Δ-ambiguous</u> if $\Xi_\Delta(G,1) > 1$. For a cf-language L the characteristic $\Xi_\Delta(L) = \min\{\Xi_\Delta(G,1) \mid L(G,1)=L\}$ is called a <u>degree of Δ-ambiguity of L</u>. A cf-language L is <u>Δ-unambiguous</u> if $\Xi_\Delta(L) \leq 1$, and <u>Δ-ambiguous</u> if $\Xi_\Delta(L) > 1$.

<u>Results</u>.

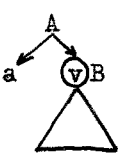
Each unambiguous cf-language is trivially d-unambiguous. An analogous proposition is true for Δ case.

<u>Theorem 1</u>. Each unambiguous cf-language is Δ-unambiguous.

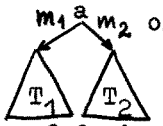
<u>Proof</u>. Let us consider an unambiguous cf-language L. As it is known [8], there is an unambiguous cf-grammar in normal form $\Gamma=(V,W,I,R)$ generating L. We relate to Γ the following Δ-system (G,1). We set $G=(V,\{m_1,m_2\},W,I,R')$, where $R' = \{A \to m_1 a m_2 \mid A \to aBC \text{ is in } R\} \cup \{A \to a\downarrow m_1 \mid A \to aB \text{ is in } R\} \cup \{A \to a \mid A \to a \text{ is in } R\}$, and define the operator 1 as follows: $1(m_1 a m_2) = \{a(a_1,m_1)(a_2,m_2)\}$ and $1(a\downarrow m_1) = \{a(a,m_1)\}$ for all a,a_1,a_2 in V. Let us use the notation t(T)=x for a string x in V^+ and for a derivation tree T of x in Γ. We may introduce the following function g from derivation trees in Γ to pd-trees in $L_\Delta(G)$: 1. $g(\overset{A}{\downarrow}{a}) = a$. 2. Let a derivation tree T be of the form



or of the form



with t(T) in V^+, and let by induction T_1 and T_2 be the single pd-trees such that $g(T(v))=T_1$ and $g(T(v'))=T_2$. Then g(T) is the pd-tree



or respectively

Let $\Delta(\Gamma)$ denote the set of all trees of derivations of the form $I \overset{*}{\Longrightarrow}_\Gamma x$, where x is in V^+, and let g^I be the restriction of g to $\Delta(\Gamma)$. By the choice of G g^I is the function from $\Delta(\Gamma)$ <u>onto</u> $L_\Delta(G)$. By definition $O_1(g(T)) = \{t(T)\}$ for all T in $\Delta(\Gamma)$. From this it follows that $L(G,1)=L(\Gamma)=L$. Now, let us assume that for a string x in L there exist different pd-trees T_1 and T_2 such that $O_1(T_1)=O_1(T_2)=x$. Then we choose derivation trees T_1', T_2' such that $g(T_1')=T_1$ and $g(T_2')=T_2$.

$T_1' \neq T_2'$ because g is a function. Hence we have $t(T_1')=0_1(g(T_1'))=0_1(T_1)=x=0_1(T_2)=0_1(g(T_2'))=t(T_2')$ which contradicts unambiguity of Γ. Q.e.d.

Now we furnish an example showing that there are Δ-unambiguous ambiguous cf-languages.

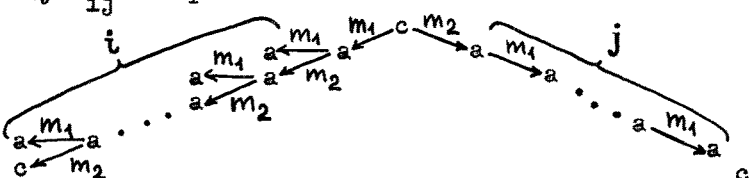
Example 1. It is clear that the cf-language $L_{ac}=\{a^ica^ica^j \mid i,j\geqslant 1\}\cup\{a^ica^ja^j \mid i,j\geqslant 1\}$ is ambiguous. Consider fs-Δ-system (G_1,l_1), where $G_1=(\{a,c\},\{m_1,m_2\},\{I,A,B\},I,R_1)$, R_1 is the set of productions

$$\{I\to \overset{m_1}{A}\ \overset{c}{}\ \overset{m_2}{B},\quad A\to \overset{m_1}{a}\ \overset{a}{}\ \overset{m_2}{A},\quad A\to \overset{m_1}{a}\ \overset{a}{}\ \overset{m_2}{c},\quad B\to \overset{a}{\underset{B}{\downarrow}}m_1,\quad B\to a\},$$

and l_1 is defined as follows (here and below we suppose that l is defined arbitrarily on those pd-trees of height 1 which are not listed):

$l_1(\overset{m_1}{a}\ \overset{c}{}\ \overset{m_2}{a})=\{(a,m_1)c(a,m_2),\ (a,m_2)c(a,m_1)\}$, $l_1(\overset{m_1}{a}\ \overset{a}{}\ \overset{m_2}{a})=\{(a,m_1)(a,m_2)a\}$, $l_1(\overset{m_1}{a}\ \overset{a}{}\ \overset{m_2}{c})=\{(a,m_1)(c,m_2)a\}$, and $l_1(\overset{a}{\underset{a}{\downarrow}}m_1)=\{a(a,m_1)\}$.

Let us denote by T_{ij} the pd-tree



$L(G_1,l_1)=L_{ac}$. Indeed, let $l_1'(T)=l_1''(T)=l_1(T)$ for all $T\neq \overset{m_1}{a}\ \overset{c}{}\ \overset{m_2}{a}$ and let $l_1'(\overset{m_1}{a}\ \overset{c}{}\ \overset{m_2}{a})=\{(a,m_1)c(a,m_2)\}$ and $l_1''(\overset{m_1}{a}\ \overset{c}{}\ \overset{m_2}{a})=\{(a,m_2)c(a,m_1)\}$. It is clear then, that each string $a^ica^ica^j$ $(a^ica^ja^j)$ in L_{ac} falls into $0_{l_1'}(T_{ij})$ (respectively into $0_{l_1''}(T_{ji})$). As it is readily seen, $L_\Delta(G_1)=\{T_{ij} \mid i,j\geqslant 1\}$. Thus $L_{ac}\subseteq L(G_1,l_1)$. At the same time, $L(G_1,l_1)\subseteq L_{ac}$ because $0_{l_1}(T_{ij})\subset L_{ac}$ for all i,j. Finally, Δ-unambiguity of fs-Δ-system (G_1,l_1) follows from the fact that $0_{l_1}(T_{ij})\cap 0_{l_1}(T_{lk})=\emptyset$ for all $i,j,k,l\geqslant 1$ such that either $i\neq l$, or $j\neq k$. Our example together with the theorem 1 gives

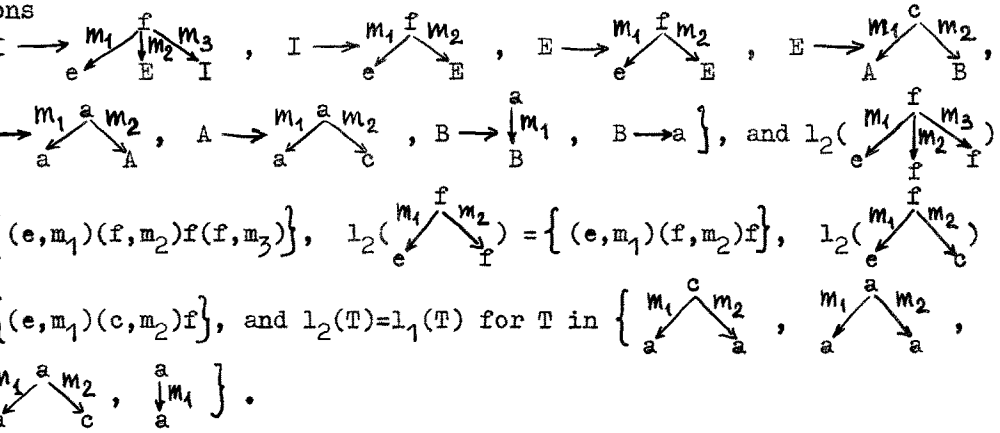
Corollary 1. The class of all unambiguous cf-languages is a proper subclass of the class of all Δ-unambiguous cf-languages.

As it is shown in [4] (cf.[3]), the language $\{a^ib^jc^j \mid i,j\geqslant 1\}\cup\{a^ib^ic^j \mid i,j\geqslant 1\}$ is d-ambiguous. The proof of this theorem (with minory changes) is fit to prove d-ambiguity of L_{ac}. Thus we have

Corollary 2. There exist Δ-unambiguous d-ambiguous cf-languages.

Let us consider one more example of Δ-unambiguous ambiguous cf-language (Δ-unambiguity may be verified as in example 1).

Example 2. The cf-language L^{+}_{acef}, where $L_{acef} = \{ e^i x f^i \mid i \geqslant 1, x \text{ in } L_{ac} \}$, is specified by Δ-unambiguous fs-Δ-system (G_2, l_2) with $G_2 = (\{a,c,e,f\}, \{m_1,m_2,m_3\}, \{I,A,B,E\}, I, R_2)$, where R_2 is the set of productions

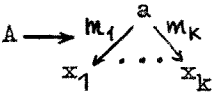


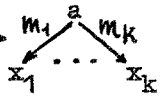
In the paper [4] it is shown that $\sqsupset_D(L^{+}_{acef}) = \infty$. At the same time the proof of this theorem shows that the degree of ambiguity of L^{+}_{acef} is ∞ (notation $\sqsupset(L) = \infty$). Thus we obtain the following stronger form of corollary 2.

Corollary 3. There exist Δ-unambiguous cf-languages of infinite degrees of ambiguity and d-ambiguity.

Nevertheless, as it might be expected, there exist Δ-ambiguous cf-languages. We expose several such languages.

Theorem 2. The cf-language $L_{ab} = \{ a^n b^m a^n b^k \mid k,m,n \geqslant 1 \} \cup \{ a^m b^n a^k b^n \mid k,m,n \geqslant 1 \}$ is Δ-ambiguous.

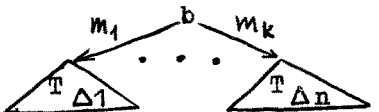
Proof. Let $(G',1')$ be a fs-Δ-system specifying L_{ab}. As it is noted in [5] (remarks 1,2 to the proof of theorem 3) $(G',1')$ may be reduced to a fs-Δ-system $(G,1)$ with the properties a) $L_\Delta(G) = L_\Delta(G')$ and b) $1 = 1'$, in the following canonical form: 1) the axiom I cannot occur on the right side of any production in R; 2) each production in R is either of the form  , A in W, a in V, $x_1, \ldots, x_k$ in $V \cup W$, or of the form $A \longrightarrow a$, A in W, a in V, or of the form $I \longrightarrow A$, A in W; 3) for all A in W any two productions $A \longrightarrow T_1$ and $A \longrightarrow T_2$ in R have identical root labels, if these labels are terminals. The conditions a),b) imply not only $L(G',1') = L(G,1) = L_{ab}$ but also $\sqsupset_\Delta(G',$

$l') = \Xi_{\Delta}(G,l)$. Thus it suffice to show that (G,l) is Δ-ambiguous. Let $G = (V,M,W,I,R)$. First of all we effect the following simple construction used in the theorem 3 in [5]. By the canonical form condition 3) there is a function $r: W \to V$ such that $r(A)=a$ iff a is in V and there is a production $A \to T$ in R with the root of T labelled by a. Let $P = A \to$  be a production in R. We relate the

tree $T_P =$  to P, in which $x_i' = x_i$ if x_i is in V, and $x_i' = r(x_i)$ if x_i is in W for all i, $1 \leq i \leq k$. It is clear that $l(T_P)$ is always defined. So, we may introduce the cf-production $A \to x_{i_1} \ldots x_{i_l} a x_{i_{l+1}} \ldots x_{i_k}$ for each string $(x'_{i_1}, m_{i_1}) \ldots (x'_{i_l}, m_{i_l}) a (x'_{i_{l+1}}, m_{i_{l+1}}) \ldots (x'_{i_k}, m_{i_k})$ in $l(T_P)$. Assembling all such productions, adding to them all the productions of the form $A \to a$ in R, eliminating productions of the form $I \to A$, A in W, and finally, setting I to be the axiom, we obtain a cf-grammar $\Gamma(G,l)$ generating L_{ab}. Each derivation tree in $\Delta(\Gamma(G,l))$ (notation $\Delta(\Gamma)$ is introduced in theorem 1) is quasinormal, i.e. every its nonbottom node has an immediate successor labelled by a terminal. This provides correctness to the inductive definition of the following function $h: \Delta(\Gamma(G,l)) \to 2^{\Delta(V,M)}$. Let T be a derivation tree in $\Delta(\Gamma(G,l))$ and v be a node in T (the definition proceeds by induction on complete subtrees of T). Then

a) If v is a bottom node of T labelled by a terminal a in V, $h(T(v))$ consists of single one-node tree with the label a.

b) Let v be a nonbottom node in T. A pd-tree T_Δ is in $h(T(v))$ iff there exist a sequence $v_1, \ldots, v_k, v_0, v_{k+1}, \ldots, v_n$ of all immediate successors of v in T with v_0 labelled by a terminal b in V, and a sequence of arrow labels $m_1, \ldots, m_n$ in M, and a sequence of pd-trees $T_{\Delta 1}, \ldots, T_{\Delta n}$, such that $T_{\Delta i}$ is in $h(T(v_i))$ for all $1 \leq i \leq n$, and

$T_\Delta =$ 

c) $h(T) = h(T(\bar{v}))$, where $\bar{v}$ is the root of T.

Some simple properties of this function we state below using the following

Notation. Let T be a tree with labelled nodes and a be a symbol. $|T|_a$ denotes the number of nodes in T labelled by a.

Lemma 1 (obvious). Let T be a derivation tree of a terminal string in

$\Gamma(G,1)$, and T_Δ be a pd-tree in h(T). Then 1) $|T|_a = |T_\Delta|_a$ for each terminal a in V; 2) For each node v in T there is some complete subtree of T_Δ in the set h(T(v)); 3) For each complete subtree T'_Δ of T_Δ there is a node v in T such that T'_Δ is in h(T(v)).

Moreover, from the construction of $\Gamma(G,1)$ immediately follows

Lemma 2. For any derivation tree T in $\Delta(\Gamma(G,1))$ of a terminal string x there is a pd-tree T_Δ in $h(T) \cap L_\Delta(G)$ such that x is in $O_1(T_\Delta)$.

Using lemmas 1 and 2 we reduce theorem 2 to the following

Proposition. There exist a string Y in L_{ab}, derivation trees T_1 and T_2 of Y in $\Delta(\Gamma(G,1))$ and a complete subtree T'_1 of T_1 such that, whatever would be a complete subtree T'_2 of the tree T_2, either $|T'_1|_a \neq |T'_2|_a$, or $|T'_1|_b \neq |T'_2|_b$.

The reduction proceeds as follows. Let the proposition be true. In this case by lemma 2 we may choose for our string Y two its pd-trees $T_{\Delta 1}$ in $h(T_1) \cap L_\Delta(G)$ and $T_{\Delta 2}$ in $h(T_2) \cap L_\Delta(G)$ such that Y is in $O_1(T_{\Delta 1}) \cap O_1(T_{\Delta 2})$. Let us show that $T_{\Delta 1} \neq T_{\Delta 2}$ (this would imply Δ-ambiguity of (G,1)). Indeed, by conditions 1) and 2) of the lemma 1 we find in the tree $T_{\Delta 1}$ a complete subtree $T'_{\Delta 1}$ such that $|T'_{\Delta 1}|_a = |T'_1|_a$ and $|T'_{\Delta 1}|_b = |T'_1|_b$. Therefore, if there had been a complete subtree $T'_{\Delta 2}$ in $T_{\Delta 2}$ such that $|T'_{\Delta 2}|_a = |T'_{\Delta 1}|_a$ and $|T'_{\Delta 2}|_b = |T'_{\Delta 1}|_b$, then by conditions 1) and 3) of the lemma 1 there would have been a complete subtree T'_2 in the tree T_2 such that $|T'_2|_a = |T'_{\Delta 2}|_a = |T'_{\Delta 1}|_a = |T'_1|_a$ and $|T'_2|_b = |T'_{\Delta 2}|_b = |T'_{\Delta 1}|_b = |T'_1|_b$, a contradiction.

We prove the proposition by successive applications of the following

Lemma 3([1], theorem 4.7; cf.also [9]). For each cf-grammar Γ there is an integer $s(\Gamma)$ such that for any string X and for any its full derivation D=(I,...,X), if X=xyz with $|y| \geq s(\Gamma)$, the string y may be segmented so that $y=y_1vy_2$, $v \neq \Lambda$, and the derivation D drops either a)into subderivations

$$I \overset{*}{\Longrightarrow} x_1Ay_2z \overset{*}{\Longrightarrow} x_1uAvy_2z \overset{*}{\Longrightarrow} x_1ux_2vy_2z = X,$$

or b)into subderivations

$$I \overset{*}{\Longrightarrow} xy_1Az_2 \overset{*}{\Longrightarrow} xy_1vAwz_2 \overset{*}{\Longrightarrow} xy_1vz_1wz_2 = X.$$

(A derivation is full if it is a derivation of a terminal string from the axiom.) Application of the lemma 3 to a string X distinguishes in every derivation D of X a subderivation of the form $A \overset{*}{\Longrightarrow} uAv$ (in the first case) or of the form $A \overset{*}{\Longrightarrow} vAw$ (in the second case)

which could be either iterated or removed to get a new full derivation. About this subderivation we shall write that it is factored from D by the given application of the lemma 3.

Now, so that to prove our proposition let us consider the value $s = s(\Gamma(G,1))$ of the parameter $s(\Gamma)$ in lemma 3 and set $q = (2s)!$. Besides this let us consider the string $X = a^{s+q}b^s a^s b^s$ in L_{ab}, a full derivation D of X in the grammar $\Gamma(G,1)$ and the constituent-structure C of X induced by D. We shall effect two applications of lemma 3 to different substrings of X and number them by Roman numerals.

I. First we apply lemma 3 to X with $x=a^{s+q}$, $y=b^s$ and $z=a^s b^s$. It is easy to check that for such application of lemma 3 only the case b) in its wording is possible. So, in this case the substring w has the form $w=b^d$, $d>0$, and is not a substring of y. Thus in C we may distinguish the following constituents

(1) $X = a^{s+q}b^e(b^d(b^f a^s b^j)b^d)b^i$, where $e,i < s$.

II. The second application of lemma 3 to X we effect with $x=a^{s+q}b^s$, $y=a^s$ and $z=b^s$. If this application leads to the case b) the string w is either empty or is a substring of the distinguished occurrence of the string y. The same is true for the string u in the case a) because, as we have proved already, the structure C contains constituents of the form (1). Let $c=|u|+|v|$ in the case a) and $c=|v|+|w|$ in the case b) (for u and v or v and w resulting from the second application of lemma 3). Besides this, let D_I and D_{II} be the subderivations factored from the derivation D by first and second applications of lemma 3. We choose positive numbers r and p such that $rd=pc=q$ and iterate subderivations D_I and D_{II} in D r and p times respectively. As a result we obtain a derivation D_1 of the string

$$Y = a^{s+q}b^{e+d+rd+f}a^{s+pc}b^{j+d+rd+i} = a^{s+q}b^{s+q}a^{s+q}b^{s+q} .$$

In the constituent structure C_1 induced by D_1 on Y we may distinguish the constituent

(2) $Y = a^{s+q}b^e(b^{s+q-e}a^{s+q}b^{s+q-i})b^i$, where $e,i < s$.

Let us now take the string $Z = a^s b^s a^s b^{s+q}$ in L_{ab} and consider some full derivation D' of Z and the constituent structure C' induced by D' on Z. We apply lemma 3 twice to the string Z, first with $x=\Lambda$, $y=a^s$ and $z=b^s a^s b^{s+q}$, and afterwards with $x=a^s$, $y=b^s$ and $z=a^s b^{s+q}$. Then by similar construction we obtain a full derivation D_2 of the same string Y, but this time we find in the constituent structure C_2 induced on Y by D_2 the constituent

(3) $Y = a^g(a^{s+q-g}b^{s+q}a^{s+q-h})a^h b^{s+q}$, where $g,h < s$.

Let T_1 and T_2 be the derivation trees of D_1 and D_2 respectively.

By (2) we have a complete subtree T_1' in T_1 such that $t(T_1')=b^{s+q-e}a^{s+q}b^{s+q-i}$. It should be noted that $|T_1'|_b > 2q$ and $|T_1'|_a = s+q$. But if there is a complete subtree T_2' in T_2 such that $|T_2'|_b > 2q$, then by (3) the constituent $t(T_2')$ contains the substring $a^{s+q-g}b^{s+q}a^{s+q-h}a^h = a^{s+q-g}b^{s+q}a^{s+q}$ which implies $|T_2'|_a \geqslant s+q-g+s+q \neq s+q$. This verifies our proposition and hence the theorem. Q.e.d.

As it is noted in [1], the language L_{ab} is d-unambiguous (a d-unambiguous gdd-grammar generating L_{ab} may be found in [3]). Therefore theorem 2 implies

Corollary 4. There exist Δ-ambiguous d-unambiguous cf-languages.

Next theorem gives an example of a cf-language of infinite degree of Δ-ambiguity.

Theorem 3. The cf-language $L_{abc} = \{ a^n b a^{i_1} c a^{i_2} c \ldots c a^{i_k} \mid k,n,i_j = 1,2,\ldots, (\exists\, j)_{1 \leqslant j \leqslant k} [i_j = n] \}$ has the property $\Xi_\Delta(L_{abc}) = \infty$.

The proof of this theorem shows also that $\Xi(L_{abc}) = \infty$ (this, however, being known already - cf. [1], p.154). At the same time this language is shown in [2] to be d-unambiguous. Thus we have

Corollary 5. There exists a d-unambiguous cf-language L such that $\Xi(L) = \infty$ and $\Xi_\Delta(L) = \infty$.

The last theorem of this paper shows that the union of Δ-unambiguous and d-unambiguous languages is itself a proper subfamily of cf-languages.

Theorem 4. The cf-language $L_{abd} = \{ a^n b^{2m} d^n \mid m,n \geqslant 1 \} \cup \{ a^n b^{n+m} d^m \mid m,n \geqslant 1 \}$ is both Δ-ambiguous and d-ambiguous. (D-ambiguity of this language is shown in [4].)

REFERENCES

1. Gladkiĭ A.V., Formal grammars and languages, M., Nauka, 1973 (russ.) - in translation into English by North-Holland P.C°.
2. Modina L.S., On dependency ambiguity of context-free languages, in print (russ.)
3. Modina L.S., On some formal grammars generating dependency trees, Lecture Notes in Computer Sci., 32, Berlin-Heidelberg-New-York, 1975, 326-329.
4. Modina L.S., On degrees of dependency ambiguity, unpublished manuscript.
5. Modina L.S., Dendrogrammars and dendrolanguages, Kibernetika (Kiev), 1975, N5, 86-93 (russ.)
6. Melčuk I.A., Experiment of the theory of linguistic models "Mean-

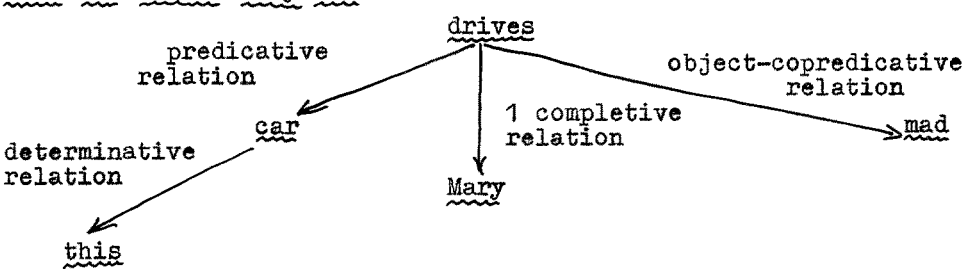
ing⇔Text", M., Nauka, 1974 (russ.)

7. Modina L.S., On some formal systems generating dependency trees, Problemy Peredači Informacii, 1976, 12, N2, 83-94 (russ.)
8. Greibach S.A., A new normal-form theorem for context-free phrase structure grammars, J.Assoc.Comput.Mach., 1965, 12, N1, 42-52.
9. Ogden W., A helpfull result for proving inherent ambiguity, Math. Syst.Theory, 1968, 2, N3, 191-194.

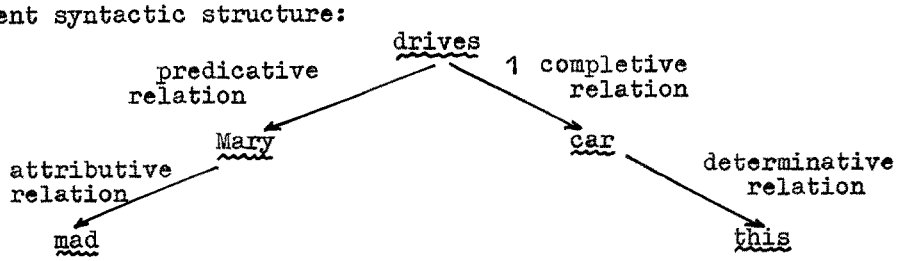
A P P E N D I X

(Few words about pure dependency trees and fs-Δ-systems.)

Pd-tree is essentially an explicit representation of syntactic relations in a sentence. The main principle of syntactic analysis is to the effect that all the formal means of manifestation of syntactic relations ought to be removed from the initial sentence description and transformed into corresponding components of a syntactic structure. Thus, the linear order of words in a sentence must also be removed, because it is, undoubtedly, one of the means of manifestation of syntactic relations. This is very peculiar to English. We relate the reader to [6] for a more detailed discussion and confine ourselves to the following example. In the notation of [6], the sentence This car drives Mary mad has the structure



However, the sentence Mad Mary drives this car which differs from the sentence above only in linear order of words, has absolutely different syntactic structure:



This is why linear order is divorsed from pd-trees. As a consequence, the procedure of syntactic synthesis drops naturally into two phases, first of them being a process of generation of a pd-tree (unordered structure), and the second being a transformation of the pd-tree into

a sentence (linearized sequence of words). This approach is adopted in [6] and formalized in [5] through the notion of Δ-system (Δ from the Greek δένδρον). So that to make this principle transparent we deliberately confine ourselves in this paper to the simplest class of Δ-systems (fs-Δ-systems). That is why the examples of Δ-unambiguous ambiguous languages turn to be artificial. However, fs-Δ-systems may be essentially enriched (without augmenting the class of specified languages) by means of generalization of linearization operators. We may, for example, allow relabelling in the course of (top-down) linearization. Even such a weak generalization provides Δ-unambiguity to more interesting cf-languages, such as $\{a^i b^i c^j \mid i,j \geqslant 1\} \cup \{a^i b^j c^j \mid i,j \geqslant 1\}$ or $\{x\hat{c}\hat{x}cy\} \cup \{zcu\hat{c}\hat{u}\}$ and many others. Nevertheless, theorem 2 would still remain true which shows that overlaping constituents lead to unavoidable Δ-ambiguity. It must be noted that fs-Δ-system with relabelling is itself a very natural model because in general the linearization of deep structure is accompanied by morphological transformations.