T.A.L., 2000, vol. 41,n°1, pp. 67–96

DEPENDENCIES ON THE OTHER SIDE OF THE CURTAIN

Alexander DIKOVSKY* **Larissa MODINA****

Résumé - Abstract

A short survey is presented of formal models, methods and results in the domain of dependency syntax of natural languages, which were in the focus of the research in the USSR in the period from the early 60ies to the late 70ies.

Cet article passe en revue des modèles formels, méthodes et résultats dans le domaine de la syntaxe de dépendances développés en URSS du début des années 60 à la fin des années 70.

Mots Clefs - Keywords

Formal language, dendrolanguage, dependency grammar, categorial grammar, dependency ambiguity, structural complexity.

Langage formel, dendrolangage, grammaire de dépendance, grammaire catégorielle, ambiguïté de dépendance, complexité structurelle.

*Université de Nantes, IRIN, 2, rue de la Houssinière, BP 92208, 44322 Nantes cedex 3, France (Email: Alexandre.Dikovsky@irin.univ-nantes.fr) and Keldysh Institute for Applied Math., Russian Academy of Sci., Moscow

**Institute of Oriental Studies, Russian Academy of Sci., 12, Rozhdestvenka St., Moscow 103753, Russia

1. About this survey

Why we have written it. For only one reason : to familiarize the people working in the domain of formal syntax of natural languages with the milestones of the research in this domain in the USSR in the period from the early 60ies to the late 70ies. During this period the papers could be published exclusively in the USSR and in Russian. So with minor exclusions they are still unknown in the West. Meanwhile, even after such a long time, they may be of interest at least for three reasons :

- the formal syntax models designed for natural languages in the USSR were mostly based on the concept of **syntagmatic dependency** rather than **constituency** ;
- from the very beginning the research was well founded mathematically ;
- along with this, its motivation has always remained linguistic.

Dependency based syntactic description has a long tradition in many natural languages. Explicitly and rather formally it was presented by Tesnière (Tesnière L. 1959) and Hays (Hays D. 1961). We address the reader to the book (Mel'čuk I. 1988) for the discussion and linguistic foundations of dependency-based syntax. Here we limit ourselves to mentioning that dependencies are manifested explicitly as binary relations between wordforms. If one abstracts from the relations between inflectional forms (morphological dependencies) and from semantic relations between wordforms (semantic dependencies), the remaining syntactic dependency graphs are trees : dependency trees. Dependency trees have various advantages as compared to the more extensively used constituent trees. They are more succinct, explicit, and expressive. Very importantly, in the strict sense they are also invariant with respect to the word order. In languages with the so called "free word order", and developed inflection, such as Russian, differently ordered instances of the same dependency tree reflect extra-syntactic distinctions. For instance, for the Russian phrase *' Včera avtory zakončili obzor '* (*Yesterday the authors finished the survey*) all the 24 reorderings are correct and syntactically equivalent[1]. They have the same dependency tree depicted in Fig. 1, whereas they are stylistically or emphatically different, or they have different communicative structures.

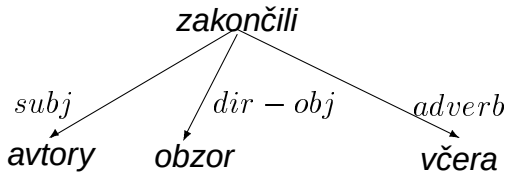


Figure 1.

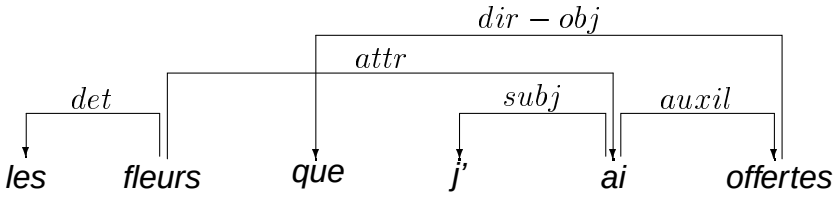
[1]Some of them are not syntactically equivalent if one follows the point of view of (Gladkij A. 1985) that communicative structure has to do with syntactic structure.

The popularity of constituent structure also introduced by linguists (cf. (Bloomfield L. 1933; Wells R. 1947; Harris Z. 1961)), may be explained by the fact that it is well suited to the frozen structure of programming languages, that it is isomorphic to the derivation trees in context-free grammars of Chomsky, and so it serves for fast syntactic analysis, and then for semantic transformations (optimization, code generation) of these languages.

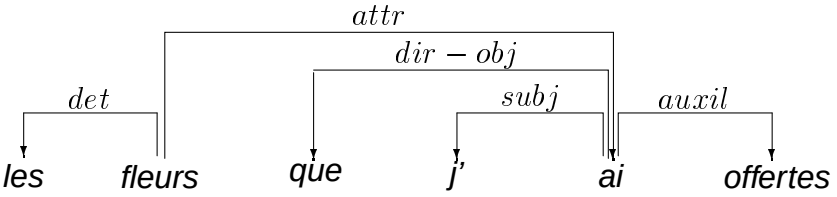
The research in the domain of the theory of formal grammars was very intensive in the USSR. We cannot even pretend to present here a survey that would be at any extent complete, even of that branch which was influenced by applications to automatic processing of natural language (some information can be inferred from (Gladkij A. 1973; Gladkij A. 1985; Gladkij A. & Dikovskij A. 1972)). Instead we will sketch the broad outlines of the formal models and methods of dependency syntax investigated in the USSR in the period from the early 60ies to the late 70ies.

2. Preliminaries

What is a dependency tree? Basically, it is a tree with labelled edges on a set of *lexical units*. We consider surface structure trees resulting from syntactic analysis, so the edges are labelled by names of surface structure relations.



(a) a natural surface structure



(b) less adequate structure

Figure 2.

The most important feature of dependency trees is the partial *dependency* order they induce on the wordforms.

Notation 1. *For a dependency tree* d, *a node* v_2 $(r-)$ depends *on a node* v_1 *(*v_1 $(r-)$ governs v_1*) in* d *if there is an edge in* d *with the name* r *from* v_1 *to* v_2 [2]*. We denote this relation by* $DEP(v_1, v_2, r)$ *or by* $dep(v_1, v_2)$, *when the dependency name is of no concern. The* dominance relation *is the reflexive and*

[2]To be more precise, we should distinguish between the nodes and the occurrences of the lexical units which label them. To simplify notation we won't do it.

transitive closure of the relation dep. v_1 dominates v_2 (*v_2 is a* descendent of v_1*, or else v_1 is an* ancestor *of v_2) is denoted* $DOM(v_1, v_2)$. *For example, in the dependency tree* d *in Fig. 2* $dep($*fleurs*$, $*les*$), DEP($*ai*$, $*offertes*$, auxil), $ *and* $DOM($*fleurs*$, $*que*$)$.

Word order. The considerations of relations between word order and dependencies are rather delicate. If we compare the trees in Fig. 1 and 2, we see that the first one does not impose a total order on its nodes, whereas those in Fig. 2 do. The choice is not arbitrary. According to the Meaning-Text theory (Mel'čuk I. 1974; Mel'čuk I. 1981) there should be no total order of wordforms in dependency trees. One of the central ideas of this theory is that when passing from a level of description l_1 to the next depth level l_2 the means specific of the level l_1 are eliminated and transformed into combinations of means of the level l_2. Word order is one of the means of the deep-morphological level, so it should not be present in the syntactic structure resulting from the syntactic analysis. Meanwhile, if we think about the phase of surface-syntactic analysis, which applies to the totally ordered deep-morphological structure and finishes by constructing a dependency tree, we easily represent the hybrid rules in terms of both : dependencies and word order. Moreover, the definitions, the rules, the algorithms of syntactic analysis are often recursive. So it is natural and sometimes inevitable to keep the initial order till the very end. Thus the inclusion of word order into dependency trees is problem-dependent. The fundamental difference between the dependency trees and the constituent trees is that the latter are inseparable from word order, whereas the former can be considered with or without word order. For example, one can eliminate word order after the phase of syntactic analysis, because it becomes redundant. As we will be interested exclusively in the phase of the surface-syntactic analysis, we will mostly use the ordered dependency trees as the uniform syntactic structure. The natural word order they keep from the analyzed sentences is denoted by $<$.

Some refinements.[3] We will fix three alphabets : $\mathbf{W}$ (of terminal symbols, i.e. word-forms), $\mathbf{C}$ (of nonterminal symbols, i.e. syntactic categories), and $\mathbf{N}$ (of dependency relation names), and consider some kinds of structure (s-) trees, i.e. trees with nodes labeled by symbols in $\mathbf{W} \cup \mathbf{C}$ and optionally, edges labelled by names in $\mathbf{N}$.

Notation 2. *A set of s-trees will be called a* dendrolanguage*. For an s-tree* t *and its node* v, $t(v)$ *denotes the full subtree of* t *with the root* v[4]*. An s-tree is* minimal *if it has exactly one non leaf node.*

Constituent structures. $\mathbf{CS}(\mathbf{W}, \mathbf{C})$ *will denote the set of all* constituent (c-) trees*, i.e. locally ordered s-trees whose non leaf nodes are labeled by nonterminals in* $\mathbf{C}$. Locally ordered *means, that a linear order* $\prec$ *is fixed on daughters*

[3] We use the notation and the basic notions conventional for the literature in mathematical linguistics in the USSR (see e.g. (Gladkij A. 1973; Gladkij A. 1985)).

[4] $t(v)$ is the least subtree of t containing all nodes of t which belong to paths from v to a leaf.

of each non leaf node. $\prec$ *is naturally extended to the leaves of the tree. So for a c-tree* t*, its* yield $w(t)$ *is the string of the labels of its* $\prec$ *−ordered leaves.* $\mathbf{CS^t(W,C)}$ *is the set of* terminal *c-trees, i.e. c-trees* t *with* $w(t) \in \mathbf{W}^*$*. The set* $\{< l(v), w(t(v)) > \mid v$ *a node of* t*,* $l(v)$ *its label*$\}$ *is a* constituent (c-) structure *of* $w(t)$ *(*$w(t(v))$ *being a* $l(v)$*−*constituent *of* $w(t)$*). We omit the labels* $l(v)$ *when they are of no concern, regarding* constituent intervals $w(t(v))$ *as constituents. For example, the following is a c-structure of the phrase in Fig. 2.*

$$(\ (\textit{les fleurs})_{NP}\ (\ \textit{que}\ (\textit{j' (ai offertes})_{V}\ )_{Cl}\ )_{Rc}\ )_{NP}.$$

The corresponding c-tree[5] *is shown in Fig. 3.*

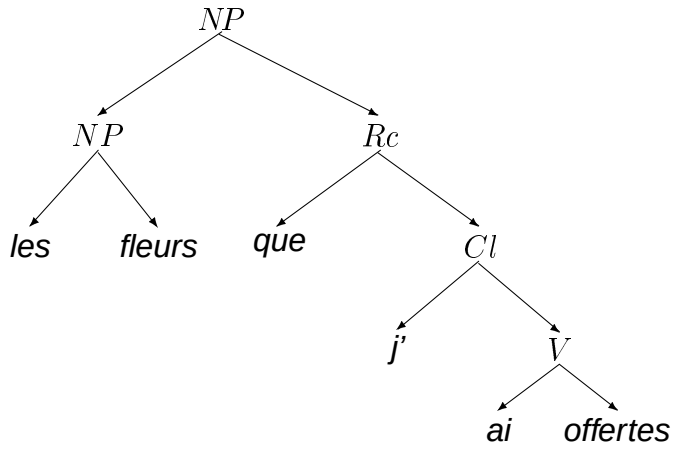


Figure 3.

For a non-unit constituent s*, the constituents corresponding in the c-tree to the daughters of* s *are called* immediate constituents of s*. For instance, in the c-tree of Fig. 3, the unit constituent 'que' and the* Cl*−constituent 'j' ai offertes' are immediate constituents of the* Rc*−constituent 'que j' ai offertes' .*

Subsets $S \subseteq \mathbf{CS^t(W,C)}$ *are called* c-dendrolanguages*.*

Dependency structures. Dependency trees *are s-trees with labeled edges. In order to distinguish between unordered and ordered dependency trees we will use lowercase letters for the former (d-trees) and uppercase letters for the latter (D-trees).* $\delta(\mathbf{W,C,N})$ *and* $\mathbf{\Delta(W,C,N)}$ *will denote respectively the set of all d-trees and of all D-trees in fixed alphabets. Dependency trees with nodes labeled only by terminals in* $\mathbf{W}$ *are* terminal*. The corresponding sets will be denoted by* $\delta^{\mathbf{t}}(\mathbf{W,N})$ *and* $\mathbf{\Delta^t(W,N)}$*.*

For a D-tree d*, the string of labels of its* $<$ *−ordered nodes will be denoted* $w(d)$ *(we will say that* d *is a D-tree on* $w(d)$*).*

Subsets of $\delta^{\mathbf{t}}(\mathbf{W,N})$ *and* $\mathbf{\Delta^t(W,N)}$ *are called respectively* d-dendrolanguages *and* D-dendrolanguages*.*

The order $<$ *and the partial dependency order* DOM *induce some additional structure in D-trees. For a D-tree* d*, let* $w(d) = a_1...a_n$*. A closed (c-) interval* $[a_i, a_j]$*,* $i \leq j$*, is the sequence* $a_i...a_j$*. For instance, in the D-trees of Fig. 2, [fleurs, ai] = 'fleurs que j' ai'. A sequence* I *of occurrences of sym-*

[5]We will not distinguish between c-structures and the isomorphic c-trees.

bols in the string $w(d)$ *is a* ring interval *if it is either a c-interval or* $I = I_1 \cup I_2$ *for two c-intervals* I_1, I_2 *such that* I_1 *starts with the first word* a_1 *of* w*, and* I_2 *ends with the last word* a_n *of* w*. For instance, the sequence of occurrences ' les fleurs ai offertes ' is a ring interval. A c-interval* $[a_i, a_j]$ *of* $w(d)$ *is an* arrow interval *if* $dep(a_i, a_j)$ *or* $dep(a_j, a_i)$ *(the governor being the* interval head *). An arrow interval is* correct *if all its elements are dominated by its head. Two arrow intervals* $[a_i, a_j]$ *and* $[a_l, a_k]$ interlace *if* $a_i < a_l < a_j < a_k$ *or* $a_l < a_i < a_k < a_j$.

For a node v *of* d*, the ordered set of nodes of the full subtree* $d(v)$ *is called the* maximal projection *of* v *(*v *being its head) and is denoted by* $gr(v)$*. Its subset* $g = gr(v) \setminus \bigcup_{v_1 \in B} gr(v_1)$*, where* B *is some set of nodes dependent on* v*, is called a* projection[6] *of* v*. For instance, in D-tree (b) in Fig. 2,* $gr($*que*$)$ *is the unit interval* $[$*que,que*$]$*,* $gr($*ai*$) = [$*que,offertes*$]$*, and among the projections of ' ai ' there are* $[$*ai,offertes*$]$ *and* $[$*ai,ai*$]$.

The (unordered) multiset[7] *of all occurrences of symbols in* $w(d)$ *is called the* support *of* $w(d)$ *and is denoted by* $\lambda(w(d))$ *or just* $\lambda(d)$.

We will use a simple operator of composition *of D-trees, similar to substitution of strings. Let* d_1, d_2 *be two D-trees,* v *be a node of* d_1*,* v_0 *be the root of* d_2*, and* $w(d_1) = u_1 v u_2$*. Then the* composition *of* d_1 *and* d_2 *in* v*, denoted* $d_1[v \backslash d_2]$*, is the D-tree on* $w(d_1[v \backslash d_2]) = u_1 w(d_2) u_2$*, such that* $DEP(v_1, v_2, r)$ *in* $d_1[v \backslash d_2]$ *iff* $v_1, v_2 \in \lambda(w(d_1)) \setminus \{v\}$ *and* $DEP(v_1, v_2, r)$ *in* d_1*, or* $v_1, v_2 \in \lambda(w(d_2))$ *and* $DEP(v_1, v_2, r)$ *in* d_2*, or* $v_1 = v_0, v_2 \in \lambda(w(d_1))$ *and* $DEP(v, v_2, r)$ *in* d_1*, or* $v_2 = v_0, v_1 \in \lambda(w(d_1))$ *and* $DEP(v_1, v, r)$ *in* d_1*. The result of the parallel composition into a D-tree* d *of D-trees* $d_1, ..., d_k$ *in pairwise different nodes* $v_1, ..., v_k$ *is denoted by* $d[v_1 \backslash d_1, ..., v_k \backslash d_k]$.

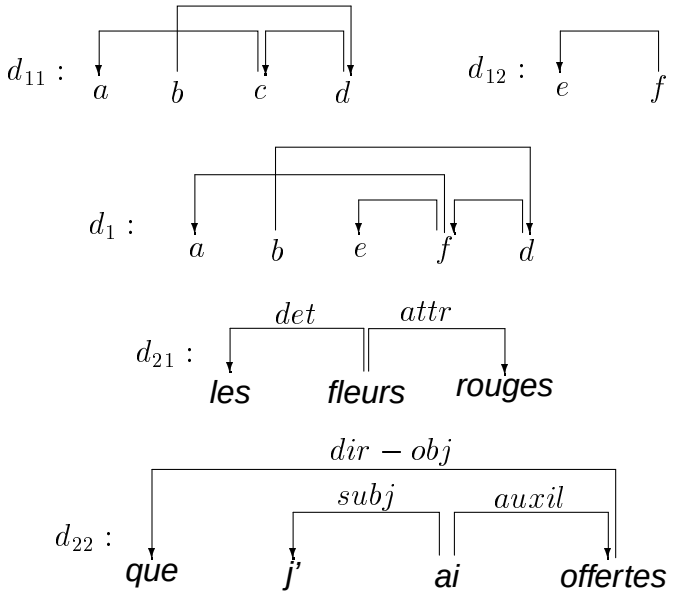


Figure 4.

[6] In russian literature they are called *dependency group* and *cutdown dependency group* respectively.

[7] I.e. a "set" with possible repetitions of elements.

For example, D-tree d_1 *in Fig. 4 is a composition of D-trees* d_{11}, d_{12} : $d_1 = d_{11}[c \backslash d_{12}]$, *and D-tree* d_2 *in Fig. 2 (a) is a composition of D-trees* d_{21}, d_{22} *in Fig. 4 :* $d_2 = d_{21}[$*rouges*$\backslash d_{22}]$.

The D-tree composition we use is different from the operators of substitution and adjunction of TAGs (tree adjoining grammars) (Joshi A. *et al.* 1975) at least in two respects : the latter apply to unordered s-trees, and they preserve the out-degree (i.e., the number of daughters) of nodes.

Dendrolanguages and languages. An ordered s-tree describes both : a phrase and its structure. So each dendrolanguage of ordered s-trees determines the corresponding language : for a D-dendrolanguage $D \subseteq \mathbf{\Delta^t}(\mathbf{W}, \mathbf{N})$ (or a c-dendrolanguage $D \subseteq \mathbf{CS^t}(\mathbf{W}, \mathbf{C})$), $L(D) = \{w(t) \mid t \in D\}$ is the language it determines. This means in particular, that we should distinguish between weak and strong equivalence of grammars (or automata) describing D- or c-dendrolanguages. For two such grammars G_1, G_2, they are *strongly equivalent* if they describe the same dendrolanguages : $D(G_1) = D(G_2)$, and they are *weakly equivalent* if they describe the same languages : $L(D(G_1)) = L(D(G_2))$.

3. Comparison of D-trees and c-trees

3.1. Projectivity

Tseitin (Tseitin G. 1959; Tseitin G. & Zasorina L. 1961) and Harper and Hays (Harper K. & Hays D. 1959) were first to observe that node projections in D-trees have a tendency to fill continuous intervals. This is the case of D-tree (b) (but not (a)) in Fig. 2 : the maximal projections $gr($*les*$), gr($*que*$), gr($*j'*$)$, and $gr($*offertes*$)$ are unit intervals, $gr($*fleurs*$) = w(d) =$ [*les,offertes*], and $gr($*ai*$) =$ [*que,offertes*]. This property of D-trees was called *projectivity* because under this condition they can be so placed that the projections of the nodes to the axis never intersect the edges. Fig. 5 shows non projectivity in this sense of the D-tree in Fig. 2 (a).

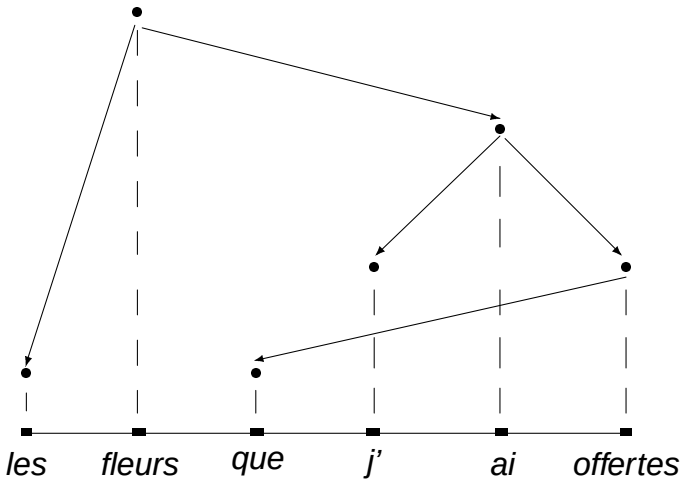


Figure 5.

The first precise definitions of this property and of its variants are due to Lecerf (Lecerf Y. 1960), Beletskij (Beleckij M. 1961), and Fitialov (Fitialov S. 1962) (we present the definitions in the form of (Fitialov S. 1962)). It should be noted that the requirement of projectivity was explicitly included by Gaifman (Gaifman H. 1961) in the definition of dependency structure established in the course of analysis of a sentence, and it was implicitly supported by the procedures of constructing D-trees, that appear in RAND publications of Hays and Ziehe (Hays D. 1960; Hays D. & Ziehe T. 1960).

Definition 3. *A D-tree* d *is* projective *if all its arrow intervals are correct. Otherwise, it is* non projective*.* d *is* weakly (w-) non projective *if it is non projective but it has no interlacing arrow intervals. And* d *is* strongly (s-) non projective *if it has interlacing arrow intervals.*

Following to this definition, D-tree (b) in Fig. 2 is projective, whereas D-tree (a) is s-non projective, and D-tree d_{22} in Fig. 4 is w-non projective.

Statistically, the projective D-trees are much more frequent than those non projective, at least in technical prose. In languages like Russian or German the non projectivity serves as one of the means of setting up the communicative structure of sentences (see Fig. 6)[8].

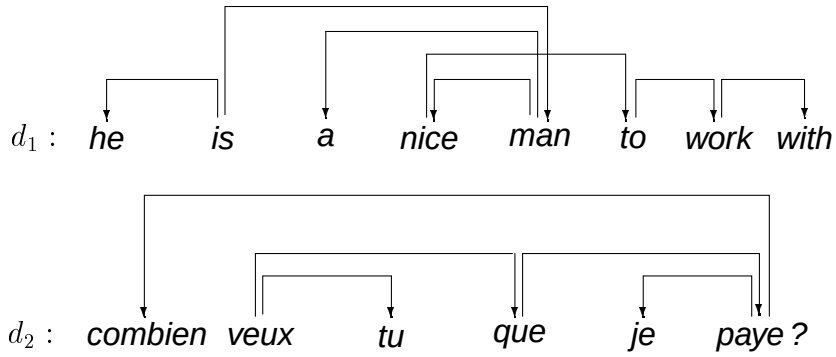


Figure 6.

The projectivity properties can be reformulated equivalently in terms of interval and dependency structure as follows.

Theorem 1. *A D-tree* d *is projective iff the maximal projections of its nodes are intervals, and it is not s-non projective iff they are ring intervals.*

In this form, the theorem was proven by Gladkij (Gladkij A. 1966). Its first part concerning projectivity was stated by Lecerf (Lecerf Y. 1961).

Corollary 2. *A D-tree which is not s-non projective, is not projective iff any of its incorrect arrow intervals contains the root (and there are some incorrect intervals).*

From this follow clear “geometric definitions" of non projectivity used by linguists (cf. (Iordanskaja L. 1963; Iordanskaja L. 1964; Padučeva E. 1964)). A

[8] An interesting investigation of style, topicalization and other kinds of extra-syntactic analysis in terms of projectivity can be found in (Sevbo I. 1981).

D-tree is non projective iff some of its arrow intervals contain the root or some arrows cross over. For example, D-tree d_1 in Fig. 6 and D-tree (a) in Fig. 2, are non projective because some arrows intersect, and D-tree d_2 in Fig. 6, is non projective because the arrow interval $[$*combien,paye*$]$ contains the root *'veux'*.

3.2. The linkage between c-trees and D-trees

In order to compare the expressive power of constituent and dependency structures, Padučeva (Padučeva E. 1964) has proposed an interesting informal notion of *conformity* between the two. The idea was that the constituents conform with the dependencies if they coincide with projections of words. This notion was put into a precise form by Gladkij (Gladkij A. 1966; Gladkij A. 1973; Gladkij A. 1985)[9].

Definition 4. *Let t be a c-structure of a string w and d be a D-tree on w. t and d* conform *if*
1) for each node v of d, $gr(v)$ is a constituent in t,
2) each constituent of t is a projection of a node of d.

For instance, it is easy to verify that the c-tree in Fig. 3, conforms with D-tree (b) in Fig. 2. In particular, the constituent $[$*que,offertes*$]$ is the maximal projection of *'ai'*, and the constituent $[$*j', offertes*$]$ is a projection of *'ai'* : $[$*j', offertes*$] = gr($*ai*$) \setminus gr($*que*$)$.

It is an evident consequence of Theorem 1 that any D-tree which conforms with some c-structure is projective. On the other hand, for each projective D-tree d, any c-structure t with constituent intervals $\{gr(v) \mid v \in \lambda(d)\} \cup \lambda(d)$ conforms with d. This c-structure is the simplest among those, which conform with d. Gaifman (Gaifman H. 1961) calls it a *ramification induced by* d and uses it in order to express his idea of "equivalence" of D-trees and c-structures. A very simple extension of c-structures, proposed by Padučeva (Padučeva E. 1964), has allowed to establish a transparent correspondence between projective D-trees and all conformable c-structures.

Definition 5. *Let t be a c-tree of a string w. A function σ, which assigns to each non unit constituent $c \in t$, one of its immediate constituents $\sigma(c)$, is called a* selection function*. $\sigma(c)$ is called the* head *of c.*

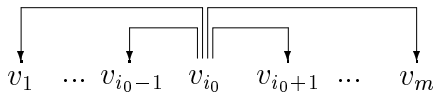
The idea is that the selected constituent $\sigma(c)$ is the head of all implicit dependencies among the immediate constituents of c. Fig. 7 presents a selection function for the c-structure in Fig. 3. The heads are marked by circles.

It is no surprise that each selection function σ on a c-tree t of a string w defines a unique D-tree on w[10]. This D-tree $d(t, \sigma)$ is said to be *induced* by σ and is defined as follows.

[9] It turns out that Hays (Hays D. 1960) has described this criterion in equivalent terms in 1960. He uses the term "correspond" in the place of "conform". Gaifman has used this notion in (Gaifman H. 1961) in order to establish the relationship between CF-grammars and dependency grammars (see below).

[10] A c-tree together with some selection function on it is sometimes called a *headed c-tree*.

Definition 6. *For a leaf c in t, the induced D-tree is the unit tree $d(c, \sigma) = c$ on c. Let a non unit constituent c have the immediate constituents $c_1, ..., c_m$, and $c_{i_0} = \sigma(c)$ be the head. For each $j, 1 \leq j \leq m$, let d_j be the D-tree $d(c_j, \sigma)$ on c_j induced by σ. Then the D-tree on c induced by σ is the D-tree $d(c, \sigma) = d_0[v_1 \backslash d_1, ..., v_m \backslash d_m]$, where d_0 is the D-tree :*



For example, the D-tree induced by the selection function on the c-tree in Fig. 7, is exactly that depicted in Fig. 2 (b).

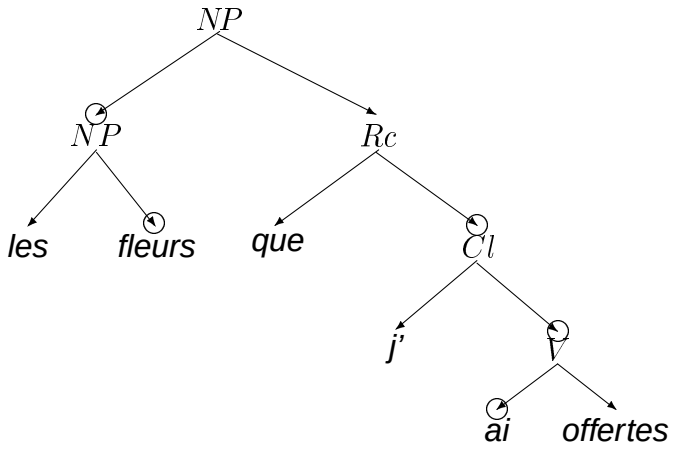


Figure 7.

The following theorem of Gladkij's (Gladkij A. 1966; Gladkij A. 1973; Gladkij A. 1985) proves what Padučeva aimed at in the cited paper : there is a one-to-one correspondence preserving conformity between c-trees with selection functions and projective D-trees.

Theorem 3.
1) For each c-tree t and for any selection function σ on t, there is a unique D-tree $d(t, \sigma) = d$; d is projective and conforms with t.
2) For each projective D-tree d and for each c-tree t which conforms with d, there is a unique selection function σ on t such that $d = d(t, \sigma)$[11]*.*

So projective D-trees are equivalent to c-trees in which the dependencies are made explicit through selection functions. As far as non projective D-trees are concerned, they cannot be expressed in this way.

[11]The idea behind this point is that among the immediate constituents of a non unit constituent c there is exactly one g, which is a non maximal projection of a node. It is g, which is to be selected in c by selection function σ_0. This selection function induces the initial D-tree.

4. Dependency grammars

4.1. Classes of dependency grammars

Ordered dependency grammars. Grammars generating sentences together with their dependency structures have emerged shortly after the appearance of phrase-structure grammars of Chomsky's[12]. They were introduced by Hays (Hays D. 1960) and explored by Gaifman (Gaifman H. 1961). We give a somewhat more general definition due to Beletskij (Beleckij M. 1967).

Definition 7. *A* basic dependency (BD-) grammar (BDG) *is a pair* $\Gamma = (G, \sigma)$, *where* G *is a reduced*[13] *CF-grammar (CFG), and* σ *is a selection function on the right-hand sides of its rules*[14]*. A complete derivation tree* t *of a string* $w \in L(G)$ *being a terminal c-tree of* w, *the selection function* σ *induces the D-tree* $d = d(t, \sigma)$ *on* w. *The set of all such induced trees is the D-dendrolanguage* $D(\Gamma)$ generated by Γ. Γ *is a* lexicalized dependency (LD-) grammar *if* $\sigma(r)$ *is a terminal for each rule* r[15]*. The classes of BD-dendrolanguages and LD-dendrolanguages, i.e. D-dendrolanguages generated respectively by BD-grammars and LD-grammars are denoted by* $\mathcal{D}(BDG)$ *and* $\mathcal{D}(LDG)$.

LD-grammars are strongly equivalent to the D-grammars initially defined by Hays. In his original definition, D-grammars are analyzing and not generating as here. Among the two BD-grammars shown in Fig. 8, the first Γ_1 is lexicalized and the other is not. We underline the occurrences selected by σ, and drop the selection where it is trivial (in unit right-hand sides).

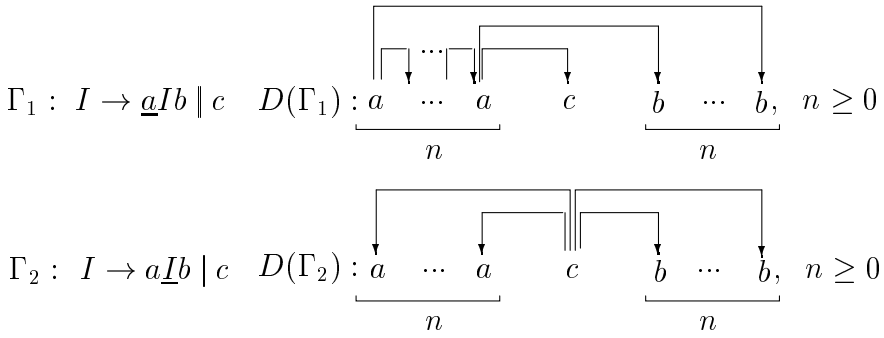


Figure 8.

A more general definition of D-grammars was proposed by the authors of this survey. It was investigated in several papers of Modina (Modina L. 1975b; Modina L. 1976; Modina L. 1978).

Definition 8. *A* dependency tree (DT-) grammar (DTG) *is a system* $G =$

[12] See however the discussion of categorial grammars below.

[13] A CF-grammar is reduced if it is Λ-free and for each nonterminal A, $I \Rightarrow^* xAy$ and $A \Rightarrow^* w$ is true for the axiom I, some strings x, y, and some terminal string w.

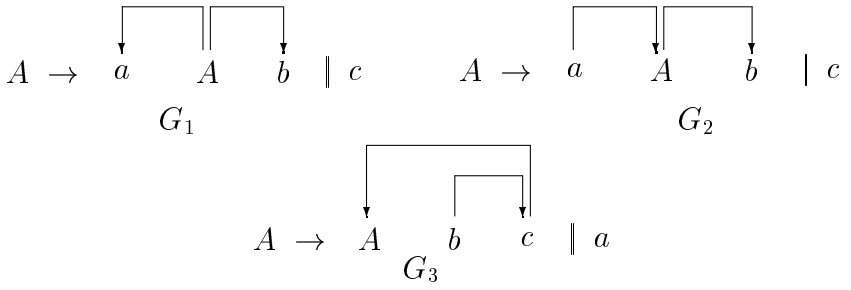
[14] In fact, Beletskij allows a set-valued selection function, i.e. several symbols may be selected in the right-hand sides of the rules. Grammars with selection functions were also introduced by Robinson (Robinson J. J. 1970).

[15] This means in particular, that each rule has a terminal in its right-hand side.

$< W, C, N, I, R >$, *where* $W \subseteq \mathbf{W}$, $C \subseteq \mathbf{C}$, $N \subseteq \mathbf{N}$ *are alphabets respectively of word-forms, categories and dependency names, and* R *is a set of composition rules of the form* $A \to d$, *where* $A \in C$ *and* $d \in \mathbf{\Delta}(W, C, N)$.

Let d_1, d_2 *be two D-trees in* $\mathbf{\Delta}(W, C, N)$. d_2 *is* immediately derivable *from* d_1 *in* G *(denoted* $d_1 \Rightarrow d_2$*) if* $w(d_1) = u_1 A u_2$ *for some nonterminal* $A \in C$, *and there is a rule* $A \to d$ *in* R *such that* $d_2 = d_1[A \backslash d]$. *The reflexive and transitive closure of* $\Rightarrow$ *is denoted by* $\Rightarrow^*$. *The* D-dendrolanguage generated *by* G *is the set* $D(G) = \{d \in \mathbf{\Delta}^{\mathbf{t}}(W, N) \mid I \Rightarrow^* d\}$. *The class of DT-dendrolanguages, i.e. D-dendrolanguages generated by DT-grammars is denoted by* $\mathcal{D}(DTG)$.

The following are examples of DTGs :



These grammars generate the following D-dendrolanguages :

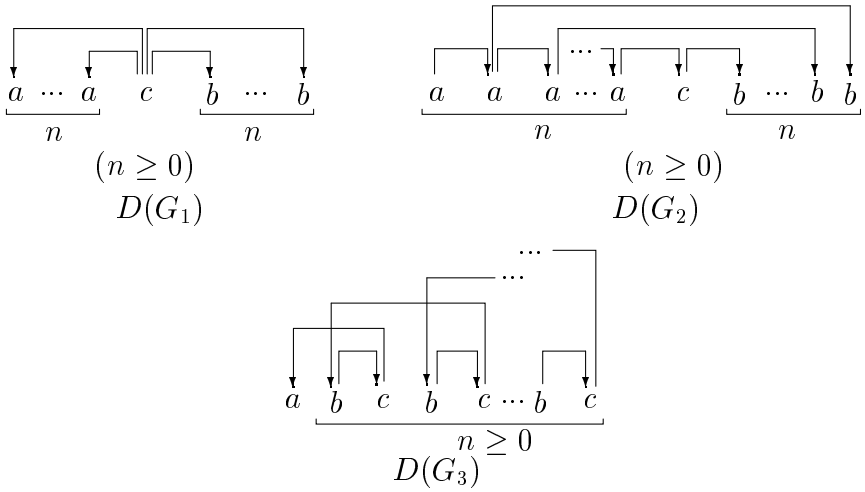


Figure 9.

Both BD-grammars Γ_1, Γ_2 in Fig. 8, are weakly equivalent to each of the DT-grammars G_1, G_2 in Fig. 9. Meanwhile, they are not strongly equivalent as D-grammars. BD-grammar Γ_2 in Fig. 8, is strongly equivalent to DT-grammar G_1 in Fig. 9, but Γ_1 is strongly equivalent neither to G_1 nor to G_2.

We distinguish several subclasses of DTGs. A DTG is *projective* if the D-trees in the right-hand sides of its rules are projective (otherwise it is *non projective*). The corresponding class of DT-dendrolanguages is denoted by $\mathcal{D}(DTG^p)$. A DTG is *bottom-recursive* if in each of its rules $A \to d$ the non leaf nodes in d are labelled by terminals in W. The corresponding class of bottom-recursive DT-dendrolanguages is denoted by $\mathcal{D}(DTG^{br})$.

Among the DT-grammars shown above, G_3 is bottom-recursive, and G_1, G_2 are not. Meanwhile, G_1 and G_2 are projective, and G_3 is not.

Evidently, each BD-dendrolanguage is also a DT-dendrolanguage. In fact, DT-grammars such that D-trees in the right-hand sides of their rules have the height 1 or 0, are exactly BD-grammars. If besides this, the roots of the D-trees are labelled by terminals, these DT-grammars are lexicalized. Theorem 3 shows that the D-trees generated by BD-grammars are always projective. So $\mathcal{D}(BDG) \subseteq \mathcal{D}(DTG^p)$. Modina (Modina L. 1976) points out the simple but important fact that BD-grammars are strongly equivalent to projective DT-grammars and LD-grammars are strongly equivalent to projective bottom-recursive DT-grammars.

Theorem 4.
1) $\mathcal{D}(BDG) = \mathcal{D}(DTG^p)$.
2) $\mathcal{D}(LDG) = \mathcal{D}(DTG^{p,br})$.

Unordered dependency grammars. General tree grammars generating d-dendrolanguages (*GdT-grammars*) were introduced by Gladkij and Mel'čuk (Gladkij A. & Mel'čuk I. 1971; Gladkij A. & Mel'čuk I. 1983) and applied in (Gladkij A. & Mel'čuk I. 1974) in order to transform Russian deep syntax rules into surface ones. The GdT-grammars are defined in terms of the composition operator $C(t_0; \alpha_1, ..., \alpha_n \mid t_1, ..., t_n)$, which for given d-trees $t_0, t_1, ..., t_n$ and some nodes $\alpha_1, ..., \alpha_n$ of t_0, gives the d-tree derived from t_0 by identification of the roots of $t_1, ..., t_n$ with respective nodes $\alpha_1, ..., \alpha_n$ in t_0. The rules of GdT-grammars are of the form $t_1 \to t_2 \mid f$, where t_1, t_2 are d-trees and f is a function from nodes of t_1 into nodes of t_2. An application of $t_1 \to t_2 \mid f$ to a d-tree T giving a d-tree T' is defined by equations : $T = C(T_0; \alpha_0 \mid C(t_1; \alpha_1, ..., \alpha_n \mid T_1, ..., T_n))$ and $T' = C(T_0; \alpha_0 \mid C(t_2; f(\alpha_1), ..., f(\alpha_n) \mid T_1, ..., T_n))$, for some d-tree T_0, its leaf α_0 and an enumeration $\alpha_1, ..., \alpha_n$ of nodes of t_1. TAGs are a special kind of GdT-grammars. Gladkij and Mel'čuk introduce several subclasses of GdT-grammars. Among them are the class of *context free* GdT-grammars (CFdG), whose rules have minimal d-trees in their left-hand sides, and whose context functions f are identities, and the subclass of CFdG of bounded out-degree (called *regular (RdG)* in (Gladkij A. & Mel'čuk I. 1971)). CFdG and RdG generate CFd- and Rd-dendrolanguages respectively.

Heterogeneous dependency systems. They are combinations of a grammar G generating a dendrolanguage S, and an operator M transforming s-trees in S into D-trees. In some systems, G generates c-trees, in the other it generates d-trees. We give an example of each kind.

A c-trees based system. Clearly, CFGs can serve as grammars generating c-trees : for a CFG G, $CS(G)$ is the c-dendrolanguage of its complete derivation trees. It is well known that the class $\mathcal{CS}(CFG) = \{CS(G) \mid G \in CFG\}$ of all such c-dendrolanguages is exactly the set of *regular c-dendrolanguages*, definable by regular expressions on trees or recognized by finite top-down and bottom-up tree automata (Rounds W. 1970; Thatcher J. 1970; Modina

L. 1975a)[16]. In the simplest c-trees based dependency systems, CFGs define c-dendrolanguages, which are transformed into D-dendrolanguages by finite state tree transducers.

Definition 9. *A fs-transducer M applies to a c-tree t, whose leaves are marked by initial state q_0. The definition of this transformation M proceeds by induction as follows. Suppose that a node v of the input c-tree is labelled by a symbol α, has daughters $v_1, ..., v_m$, and the D-trees $M(v_1, q_1), ..., M(v_m, q_m)$ are already defined and marked respectively by states $q_1, ..., q_m$. Then there is an instruction of $M : (a; q_1, ..., q_m) \to (d'; q') \mid g$, which relates to $(a; q_1, ..., q_m)$ a new state q', a D-tree d', and a subtrees distribution injection g of a set $\{u_1, ..., u_k\}$ of some leaves of d' into $\{1, ..., m\}$. Application of this instruction gives $M(v, q') = d'[u_1 \backslash M(v_{g(u_1)}, q_{g(u_1)}), ..., u_k \backslash M(v_{g(u_k)}, q_{g(u_k)})]$. M succeeds on the input c-tree t, if it reaches $M(t, q_f)$ in one of its final states q_f. Such M is called a* D-transducer *if for any c-tree t in its domain, $M(t, q_f)$ is a D-tree on $w(t)$ labelled by a final state q_f. It is* projective *if besides this, the resulting D-tree is always projective.*

Definition 10. *A* cD-system *is a pair $\Gamma = (G, M)$, where G is a CFG and M is a dependency transducer on $CS(G)$. Γ is* projective *if M is. $D(\Gamma) = \{d \in M(t, q_f) \mid t \in CS(G),\ q_f\ final\}$ is the* D-dendrolanguage defined by Γ. *$\mathcal{D}(cDS)$ is the set of all such D-dendrolanguages, and $\mathcal{D}(cDS^p)$ is the one of those projective.*

Modina (Modina L. 1976) has proven that cD-systems are strongly equivalent to DT-grammars, this equivalence preserving projectivity.

Theorem 5.
1) $\mathcal{D}(cDS) = \mathcal{D}(DTG)$,
2) $\mathcal{D}(cDS^p) = \mathcal{D}(DTG^p)$.

Corollary 6.
1) $\mathcal{D}(DTG) = \{M(D) \mid D \in \mathcal{CS}(CFG),\ M\ is\ a\ dependency\ transducer\ on\ D\}$,
2) $\mathcal{D}(DTG^p) =$
$\{M(D) \mid D \in \mathcal{CS}(CFG),\ M\ is\ a\ projective\ dependency\ transducer\ on\ D\}$.

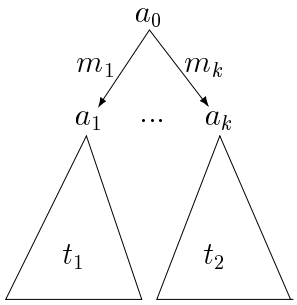
A d-trees based system. The simplest system of this kind generates d-dendrolanguages using regular d-grammars, and then applies some ordering rule in order to transform generated d-trees into D-trees.

Definition 11. *A* dD-system *is a pair $\Gamma = (G, l)$, where G is an RdG and l is a function, which maps each minimal d-tree t in the signature of G into a set of linear orderings of t. l is extended to d-trees in the natural way :*

1. For a unit tree $a \in \mathbf{W}$, $l(a) = \{a\}$.

2. For a d-tree t of the form

[16]We remind that a finite bottom-up tree automaton recognizes a tree, starting from its leaves marked by initial state q_0. Each its instruction corresponds a state q to a minimal tree t with leaves marked by some states (after applying this instruction the state q will mark the root of t). The input tree is recognized if its root is marked by one of final states at some step. The top-down automata proceed in the opposite direction.



with the top minimal sub-tree e, *let* l *map* e *into a set of D-trees* $l(e) = \{d_{01}, ..., d_{0r}\}$, *and let it map the subtrees* $t_1, ..., t_k$ *into D-dendrolanguages* $l(t_1), ..., l(t_k)$. *Then* l *maps* t *into* $l(t) = \{d_{0i}[a_1\backslash d_1, ..., a_k\backslash d_k] \mid 1 \leq i \leq r,\ d_1 \in l(t_1), ..., d_k \in l(t_k)\}$. *Now,* $D(\Gamma) = \{d \in l(t) \mid t \in d(G)\}$, $d(G)$ *being the* d-dendrolanguage generated by G.

For instance, the RdG :

$$G_{fs} : I \to \overset{c}{\underset{A\ \ B}{m_1 \diagup\diagdown m_2}} \quad A \to \overset{a}{\underset{a\ \ A}{m_1 \diagup\diagdown m_2}} \quad A \to \overset{a}{\underset{a\ \ c}{m_1 \diagup\diagdown m_2}} \quad B \to \overset{a}{\underset{B}{\downarrow m_1}} \quad B \to a$$

generates the d-dendrolanguage $d(G_{fs}) = \{t_{ij} \mid i, j \geq 1\}$ depicted in Fig10.

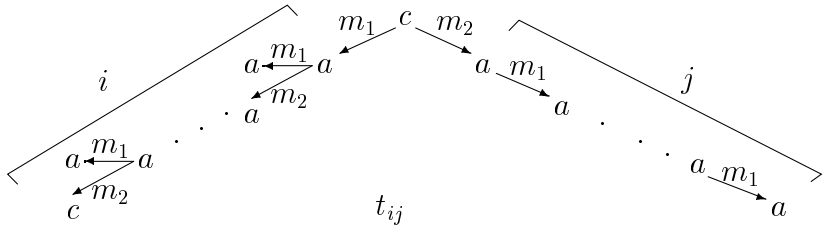


Figure 10.

If we define the order function l_0 on d-trees

$$e_1 = \overset{c}{\underset{a\ \ a}{m_1 \diagup\diagdown m_2}} \quad e_2 = \overset{a}{\underset{a\ \ a}{m_1 \diagup\diagdown m_2}} \quad e_3 = \overset{a}{\underset{a\ \ c}{m_1 \diagup\diagdown m_2}} \quad e_4 = \overset{a}{\underset{a}{\downarrow m_1}}$$

as : $l_0(e_1) = \{(a, m_1)c(a, m_2), (a, m_2)c(a, m_1)\}$, $l_0(e_2) = \{(a, m_1)(a, m_2)a\}$, $l_0(e_3) = \{(a, m_1)(c, m_2)a\}$, and $l_0(e_4) = \{a(a, m_1)\}$, then d-tree t_{ij} in Fig. 10, is mapped into the language $L(l_0(t_{ij})) = \{a^i ca^i ca^j, a^j ca^i ca^i\}$, for all $i, j \geq 1$. So the dD-system $\Gamma_{ac} = (G_{fs}, l_0)$ generates the CF-language $L_{ac} = L(D(\Gamma_{ac})) = \{a^i ca^j ca^j \mid i, j \geq 1\} \cup \{a^i ca^i ca^j \mid i, j \geq 1\}$.

BDGs vs CFGs. The notion of conformity leads to another way of comparison between CFGs and BDGs by their structural expressivity.

Definition 12. *A BDG* $\Gamma = (G_0, \sigma)$ conforms *with a CFG* G *if :*

a) for each c-tree $t \in CS(G)$, *there is a D-tree* $d \in D(\Gamma)$ *on* $w(t)$, *which conforms with* t, *and conversely,*

b) for each D-tree $d \in D(\Gamma)$ *on a string* w, *there is a c-tree* $t \in CS(G)$ *of* $w(t) = w$, *which conforms with* d[17].

Fitialov and Gladkij have investigated necessary and sufficient conditions of conformity and lexicalization. It turns out that they have reestablished in an equivalent form the results already obtained by Gaifman (Gaifman H. 1961)[18].

Definition 13. *Let* $\Gamma = (G, \sigma)$ *be a BDG. Let* $\mathcal{R}(\Gamma)$ *be the graph on terminals and nonterminals of* G, *in which there is an edge from* α *to* β *iff there is a rule* $\alpha \to x\beta y$, *where* β is selected *by* σ. Γ *is* D-loop free *if this graph does not have loops*[19].

For example, BDG Γ_1 in Fig. 8, is D-loop free and lexicalized, whereas BDG Γ_2 is neither D-loop free nor lexicalized. Clearly, each LDG is D-loop free. Fitialov (Fitialov S. 1968) proves that D-loop free BDGs are strongly equivalent to LDGs.

Theorem 7. *(Gaifman H. 1961; Fitialov S. 1968) For each D-loop free BDG, there is a strongly equivalent LDG.*

Then he gives in (Fitialov S. 1968) a necessary and sufficient condition in complexity terms, of existence of a strongly equivalent LDG. For a CFG G, let us denote by $width(G)$ the maximal length of right-hand side of its rules applied in some complete derivation. For a D-tree d, its width $width(d)$ is the maximal out-degree of nodes in d. The width of a BDG Γ is the number $width(\Gamma) = sup\{width(d) \mid d \in D(\Gamma)\}$ if such a number exists. Otherwise it is ∞[20].

Theorem 8. *(Gaifman H. 1961; Fitialov S. 1968)*
1) If $\Gamma = (G, \sigma)$ *is a D-loop free BDG,* h *is the maximal path length in* $\mathcal{R}(\Gamma)$, *and* $g = width(G)$, *then* $width(\Gamma) \leq h \cdot max\{1, g-1\}$.
2) If $\Gamma = (G, \sigma)$ *is a BDG of bounded width and* G *is a reduced CFG, then* Γ *is D-loop free.*
3) If in 1) G *is reduced, then* $width(\Gamma) \geq h$.

Corollary 9. *(Gaifman H. 1961; Fitialov S. 1968)*
1) For each BDG of bounded width, there exists a strongly equivalent LDG.
2) If a BDG Γ *is strongly equivalent to an LDG, then* Γ *has a bounded width.*

For example, this shows that BDG Γ_2 in Fig. 8, is not strongly equivalent to any LDG. In order to find conditions of conformity of CFGs and LDGs, Gladkij considers the following complexity measure due to Fitialov[21]. Let G be a CFG. Its terminals have the *degree* $dgr(a) = 0$. For a nonterminal A, $dgr(A) = n$ if $dgr(A) \neq j$ for $j = 0, ..., n-1$, and for each rule of the form $A \to x$ in G there is

[17] Clearly, this means that conformable BDG and CFG are weakly equivalent.
[18] We are grateful to the anonymous reviewer who has pointed out this fact.
[19] In terms of Gaifman (Gaifman H. 1961), this means that G is of finite degree.
[20] We set the supremum of an unbounded set of integers be equal ∞.
[21] And to Gaifman.

an occurrence of a symbol α in x of a degree $dgr(\alpha) \leq n-1$. Symbols which do not have a finite degree have the degree ∞. $dgr(G)$ is the maximal degree of its symbols. For example, the following three CFGs are weakly equivalent, but $dgr(G_1) = 2, dgr(G_2) = 3$, and $dgr(G_3) = \infty$.

$$G_1 = \{I \to CI,\ I \to aCb,\ I \to ab,\ C \to aCb,\ C \to ab\},$$
$$G_2 = \{I \to CI,\ I \to aCb,\ I \to ab,\ C \to ACB,\ C \to ab,$$
$$C \to aabb,\ A \to aa,\ B \to bb\},$$
$$G_3 = \{I \to II,\ I \to aIb,\ I \to ab\}.$$

In these terms, Gladkij (Gladkij A. 1973) gives the following conditions.

Theorem 10. *(Gaifman H. 1961; Gladkij A. 1973)*
1) For each CFG of bounded degree, there is a conformable LDG.
2) If a reduced CFG has a conformable LDG, then it has a bounded degree.

It is worth noting that theorems 7 – 10 are constructive (that is their conditions are decidable).

To summarize, DTGs are more expressive than BDGs because they express non projective D-trees. Those which are projective, are in fact, strongly equivalent to BDGs and more expressive than LDGs. The latter are strongly equivalent to BDGs of finite width and conformable with CFGs of finite degree.

BDGs vs categorial grammars. Interestingly enough, grammars which induce dependency structures on sentences were defined very long before generative grammars. We mean *categorial grammars* introduced and used in (Leśniewski S. 1929; Ajdukiewicz K. 1935) as a means of formalization of logical languages, and proposed by Bar-Hillel in (Bar-Hillel Y. 1953) as a means of natural language syntax description.

Definition 14. *Let* $\mathbf{K}$ *be a finite set of symbols called* elementary categories *and* $I \in \mathbf{K}$ *be a distinguished category called* principal*. The set* $\Psi(\mathbf{K})$ *of categories over* $\mathbf{K}$ *is defined recursively as follows. A* category *is either an elementary category or an expression of the form* $[\phi\backslash\psi]$ *or of the form* $[\phi/\psi]$*, where* ϕ, ψ *are categories. The depth* $dp(\alpha)$ *of a category* α *is the number of symbols* $\backslash$ *and* $/$ *in* α*. Let* s_1, s_2 *be sequences of categories.* s_1 reduces *to* s_2 *(denoted* $s_1 \models s_2$*) if* $s_1 = u_1, \phi, [\phi\backslash\psi], u_2$ *and* $s_2 = u_1, \psi, u_2$*, or* $s_1 = u_1, [\psi/\phi], \phi, u_2$ *and* $s_2 = u_1, \psi, u_2$*. The reflexive and transitive closure of* $\models$ *is denoted by* $\vdash$.

A categorial grammar $\mathfrak{C}$ *is a system* $(W, \mathbf{K}, \lambda)$*, where* $W \subseteq \mathbf{W}$ *is a finite set of words, and* λ *(*a lexicon*) is a function from* W *to finite subsets of* $\Psi(\mathbf{K})$*. The* language $L(\mathfrak{C})$ defined by $\mathfrak{C}$ *is the set* $\{x \in W^+ \mid (\exists s)(s \in \lambda(x), s \vdash I)\}$*.* $\mathfrak{C}$ *is* left *(*right*) if its categories use only* $\backslash$ *(respectively* $/$*).*

Let $\Psi(\mathfrak{C})$ denote the set of all subcategories of categories in $\bigcup_{a \in W} \lambda(a)$. If to each reducible pair of categories $\phi, [\phi\backslash\psi]$ and $[\psi/\phi], \phi$ such that $\phi, \psi \in \Psi(\mathfrak{C})$, we assign the rules $\psi \to \phi[\phi\backslash\psi]$ and $\psi \to [\psi/\phi]\phi$, and if we add all rules of the form $\alpha \to a$, where $a \in \lambda(\alpha)$, then we arrive at a CFG $G(\mathfrak{C})$, called *kernel CFG* of $\mathfrak{C}$, with nonterminals in $\Psi(\mathfrak{C})$ and axiom I. From definition of categorial grammar it follows that $L(\mathfrak{C}) = L(G(\mathfrak{C}))$. In fact, each complete derivation tree of a

string $x \in W^+$ in $G(\mathfrak{C})$ is also a reduction tree of some sequence $s \in \lambda(x)$ (see t in Fig. 11). Clearly, this tree is binary. So there exist two natural selection functions on such reduction trees : *standard selection function* σ^s which always selects the complex category[22] : $\sigma^s(\{\phi, [\phi\backslash\psi]\}) = [\phi\backslash\psi], \quad \sigma^s(\{[\psi/\phi], \phi\}) = [\psi/\phi]$, and *inverse selection function* σ^i which selects the simpler one[23]. Depending on the choice of one of the two selection functions, categorial grammar $\mathfrak{C}$ assigns to each string x with a reduction tree t either D-tree $d(t, \sigma^s)$ or D-tree $d(t, \sigma^i)$ (see d^s and d^i in Fig. 11). The corresponding D-dendrolanguages $D^s(\mathfrak{C}) = \{d(t, \sigma^s) \mid t \in CS(G(\mathfrak{C}))\}$ and $D^i(\mathfrak{C}) = \{d(t, \sigma^i) \mid t \in CS(G(\mathfrak{C}))\}$ are called *standard and inverse D-dendrolanguages* defined by $\mathfrak{C}$.

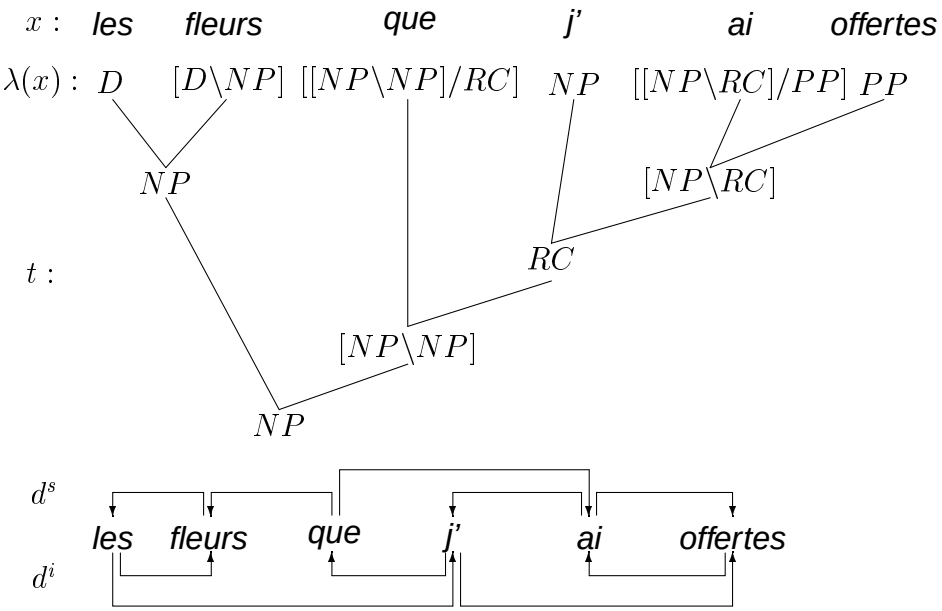


Figure 11.

The categorial grammars are proven in (Bar-Hillel Y. *et al.* 1960) to be weakly equivalent to CFGs. The fact of equivalence can be refined in terms of complexity of categories. Let $\mathcal{L}(CG^{ij})$ denote the set of languages defined by categorial grammars of depth i (i.e., using the categories of maximal depth i), left if $j = l$, right if $j = r$, and general if $j = g$. Depth 2 is enough to define all CF-languages[24]. Depth 1 gives either all linear CF-languages LIN, or all finite-state languages RAT according to orientation of categories.

Theorem 11.
1. $\mathcal{L}(CG) = \mathcal{L}(CFG)$ *(in (Bar-Hillel Y. et al. 1960)).*
2. $\mathcal{L}(CG^{2l}) = \mathcal{L}(CG^{2r}) = \mathcal{L}(CFG)$ *(in (Bar-Hillel Y. et al. 1960; Dikovskij A. &*

[22]This choice corresponds to understanding the selected complex category as a function and another category as its argument.

[23]So it selects the argument.

[24]In fact, the equivalence was proven in (Bar-Hillel Y. *et al.* 1960) in this strong form. The authors of this review have given in (Dikovskij A. & Modina L. 1968) another and quite a simple proof of this theorem by a transformation of pushdown automata to categorial grammars.

Modina L. 1968))[25]*.*

Corollary 12.
1. $\mathcal{L}(CG^{1l}) = \mathcal{L}(CG^{1r}) = \mathcal{L}(RAT)$ *(in (Gladkij A. 1973)).*
2. $\mathcal{L}(CG^{1g}) = \mathcal{L}(LIN)$ *(in (Gladkij A. 1973)).*

By definition, BDG $\Gamma^s(\mathfrak{C}) = (G(\mathfrak{C}), \sigma^s)$ is d-loop free. So by theorem 7, $\Gamma^s(\mathfrak{C})$ is strongly equivalent to a LDG, i.e. $\mathcal{D}^s(CG) \subseteq \mathcal{D}(LDG)$. The inverse inclusion is also true.

Theorem 13. $\mathcal{D}^s(CG) = \mathcal{D}(LDG)$.

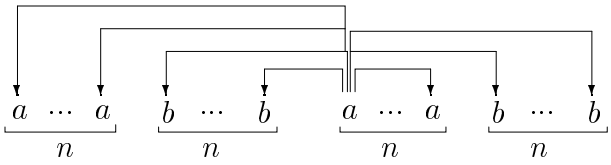
It seems that categorial grammars under inverse selection function are not of much interest because they assign unnatural D-trees (see for example, d^i in Fig. 11).

4.2. Dependency ambiguity

Gladkij was the first who discovered that there are inherently ambiguous CF-languages generated by unambiguous BDGs (Gladkij A. 1973). He has remarked that the inherently ambiguous CF-language $L_{ab} = \{a^n b^k a^n b^m \mid k, n, m \geq 1\} \cup \{a^k b^n a^m b^n \mid k, n, m \geq 1\}$ is generated by the following BDG Γ_{ab} :

$$G_{ab} = \{I \to \underline{A}b,\ A \to a\underline{C},\ A \to \underline{A}b,\ A \to a\underline{B}a,\ B \to a\underline{B}a,\ B \to a\underline{C},$$
$$C \to b\underline{C},\ C \to b\underline{a},\ I \to a\underline{D},\ D \to \underline{F}b,\ D \to a\underline{D},\ D \to b\underline{E}b,$$
$$E \to b\underline{E}b,\ E \to \underline{F}b,\ F \to \underline{F}a,\ F \to b\underline{a}\}.$$

The underlying CFG relates a unique c-tree to each string with a pair of different degrees, and it relates two different c-trees, and the unique D-tree of the form



to the strings of the form $a^n b^n a^n b^n$. It is easy to understand that unambiguous CF-languages are generated by unambiguous BDGs. So this example shows that BDGs are “less ambiguous" than CFGs. This observation led us to explore several types of dependency ambiguity.

Definition 15. *Let* G *be a DT-grammar and* $w \in L(G)$. *We define* D-ambiguity of w *in* G *as the number* $amb_D(G, w)$ *of different D-trees* $d \in D(G)$ *such that* $w(d) = w$. D-ambiguity *of* G *is defined as* $amb_D(G) = sup\{amb_D(G, w) \mid w \in L(G)\}$, *if this supremum exists. Otherwise,* $amb_D(G) = \infty$. *A CF-language* L *has* D-ambiguity $amb_D(L) = inf\{amb_D(G) \mid L(G) = L\}$. L *is* D-unambiguous *if* $amb_D(L) = 1$. L *is* inherently D-ambiguous *if* $amb_D(L) > 1$, *and it is* inherently ∞-D-ambiguous *if* $amb_D(L) = \infty$.

[25]Together with Theorem 7 this implies the Greibach normal form theorem (Greibach S. 1965).

The first example of inherently D-ambiguous CF-language was found by Modina (Modina L. 1978) :

Theorem 14. *The CF-language*
$L_{abcd} = \{a^n b^k c^l d^m \mid (n,k,l,m \geq 1) \ \wedge \ ((m = n \ \wedge \ k = l) \vee k + l > m)\}$
is both inherently ambiguous and inherently D-ambiguous.

In (Modina L. 1975a; Dikovskij A. & Modina L. 1977) we have set ourselves the problem to find a class of unordered dependency grammars less ambiguous than CFGs. It turned out that the dD-systems have this property. Evidently, dD-systems are weakly equivalent to CFGs. This makes possible the following new notion of structural ambiguity of CF-languages.

Definition 16. *Let* $\Gamma = (G, l)$ *be a dD-system and* w *be a string in* $\mathbf{W}$. *We define d-*ambiguity of w *in* Γ *as the number* $amb_d(\Gamma, w)$ *of different d-trees* $t \in d(G)$ *such that* $w \in L(l(t))$. d-ambiguity *of* Γ *is defined as* $amb_d(\Gamma) = sup\{amb_d(\Gamma, w) \mid w \in L(\Gamma)\}$, *if this supremum exists. Otherwise,* $amb_d(\Gamma) = \infty$. *A CF-language* L *has* d-ambiguity $amb_d(L) = inf\{amb_d(\Gamma) \mid L(\Gamma) = L\}$. L *is* d-unambiguous *if* $amb_d(L) = 1$. L *is* inherently d-ambiguous *if* $amb_d(L) > 1$, *and it is* inherently ∞-d-ambiguous *if* $amb_d(L) = \infty$.

It is evident that the dD-system Γ_{ac} above is d-unambiguous. So the language L_{ac} is d-unambiguous. Meanwhile, it is inherently ambiguous. In (Dikovskij A. & Modina L. 1977) we prove that unambiguity implies d-unambiguity.

Theorem 15. *Each unambiguous CF-language is d-unambiguous.*

Then in (Dikovskij A. & Modina L. 1977) we have clarified the relations between D-unambiguity and d-unambiguity.

Theorem 16.
1) The language L_{ac} *is D-ambiguous.*
2) The language L^+_{acef}, *where* $L_{acef} = \{e^i x f^i \mid i \geq 1, x \in L_{ac}\}$, *is d-unambiguous, inherently* ∞*-ambiguous and inherently* ∞*-D-ambiguous.*
3) The language L_{ab} *above is inherently d-ambiguous.*
4) The language $L_{abc} = \{a^n b a^{i_1} c ... c a^{i_k} \mid k, n, i_j = 1, 2, ..., (\exists j, 1 \leq j \leq k)[i_j = n]\}$ *is inherently* ∞*-d-ambiguous.*
5) The language $L_{abd} = \{a^n b^{2m} d^n \mid m, n \geq 1\} \cup \{a^n b^{n+m} d^m \mid m, n \geq 1\}$ *is both inherently D-ambiguous and inherently d-ambiguous.*

So the following relations hold between different properties of structural ambiguity of CF-languages.

Corollary 17.
1) The class of unambiguous CF-language is a proper subclass of the classes of D-unambiguous CF-languages and of d-unambiguous CF-languages.
2) There exist d-unambiguous inherently D-ambiguous CF-languages.
3) There exist D-unambiguous inherently d-ambiguous CF-languages.
4) There exist CF-languages, which are both inherently D-ambiguous and inherently d-ambiguous.

Once again, now in terms of structural ambiguity, it turns out that the dependency structures are more expressive than c-structures : all c-unambiguous grammars can be naturally simulated by dependency-unambiguous grammars, and there are inherently c-ambiguous languages definable by dependency-unambiguous grammars.

4.3. Structural complexity

Once we have imposed a tree-like structure on sentences, we can classify the sentences by structural complexity. If we dispose of a grammar defining sentence structures, then we can stratify the generated language into layers of bounded structure complexity. Sometimes such upper bounds serve as filters reducing the choice in syntactic analysis algorithms.

First measures of structural complexity were introduced by Yngve (Yngve V. 1960) and Bar-Hillel, Kasher, and Shamir (Bar-Hillel Y. *et al.* 1963). They have introduced several measures of branching of constituent structures.

Let t be a c-structure on a string w, and $c \in t$ be some constituent. The *left branching depth* $y^l(c)$, *right branching depth* $y^r(c)$, and *nesting depth* $\phi(c)$ of c are defined as follows.

1) For $c = w,\ \ y^l(c) = y^r(c) = 0$.

2) Let $y^l(c), y^r(c)$ be defined for a constituent $c,$ which has the immediate constituents $c_0, c_1, ..., c_k$ (in natural order). Then $y^l(c_i) = y^l(c) + k - i,\ \ y^r(c_i) = y^r(c) + i$.

3) $\phi(c)$ is the maximal length of a sequence of constituents $c_1, ..., c_k \in t$ such that $c = [l_0, r_0],\ c_i = [l_i, r_i]$ and $l_1 < ... < l_k < l_0 \leq r_0 < r_k < ... < r_1$.

For example, for the c-structure in Fig. 3., the left and right branching depth values are as follows (y^l labels the left parentheses and y^r the right) :

$$^0(\ ^1(^2\textit{les}^0\ ^1\textit{fleurs}^1)^0_{NP}\ ^0(\ ^1\textit{que}^1\ ^0(^1\textit{j}'^2\ ^0(^1\textit{ai}^3\ ^0\textit{offertes}^4)^3_V\ )^2_{Cl}\ )^1_{Rc}\ )^0_{NP}.$$

As for the nesting depth, the unit constituents *'fleurs'*, *'que'*, *' j' '* , *'ai'* have depth 1, other constituents having depth 0.

When Yngve has introduced his branching depth functions, he conjectured that because of non-symmetry of word order in natural languages, the c-structure branching to the left forces memorization for each constituent $c,$ of all sister constituents following c (as if they were put on the stack). From this he has concluded that the left (but not right) branching depth is bounded in all natural languages by the size of human short dynamic memory (i.e., 7 ± 2) (Miller G. 1956). The hypothesis has not stood, at least for Hungarian (Vargha D. 1964), Armenian, all Tiurk and Baltic languages but it is credible (not the stack idea) for English and French.

Complexity measures m on c-trees induce the corresponding complexity functions on CFGs in the following standard way. For a CFG $G,$

$$m_G(w) = min\{m(t) \mid w = w(t),\ t \in CS(G)\},$$
$$m_G(n) = max\{m_G(w) \mid |w| \leq n\}.$$

For a class of functions $\mathcal{F}$,

$$\mathcal{L}_m(\mathcal{F}) = \{L(G) \mid \exists f \in \mathcal{F}\ (m_G = \boldsymbol{O}(f))\}.$$

The interesting thing about the structure complexity functions is that constant upper bounds on them lead to “good" subclasses of CF-languages. As far as the three depth functions are concerned, Gladkij (Gladkij A. 1973) has proven that constant upper bounds on them lead to finite state languages.

Theorem 18. $\mathcal{L}_{y^l}(\mathbf{const}) = \mathcal{L}_{y^r}(\mathbf{const}) = \mathcal{L}_{\phi}(\mathbf{const}) = \mathcal{L}(RAT)$.

All the three functions are trivially bounded by linear functions, so $\mathcal{L}_{y^l}(\mathbf{lin}) = \mathcal{L}_{y^r}(\mathbf{lin}) = \mathcal{L}_{\phi}(\mathbf{lin}) = \mathcal{L}(CFG)$. Meanwhile, if for a CFG G, any of these complexity functions (say y^l) cannot be bounded by a constant, then one can prove[26] that there is a complexity gap : for some $c > 0,\ \ y^l(n) \geq c \cdot \log\log\log n$ almost everywhere. As far as we know, nobody has investigated the existence of classes of strict depth complexity between $\mathbf{lin}$ and $\log\log\log n$.

We have already seen the examples of complexity measures of D-trees : in particular, the width. Several dependency complexity measures similar to branching depth were used for style analysis of Russian prose and poetry (for example, Sevbo (Sevbo I. 1981) has made a comparative analysis of style of Pushkin, Tsvetaeva, Platonov and other russian authors in terms of structure complexity). Dikovsky (Dikovskij A. 1970) has introduced the following measure μ of balanced branching of trees, called *density*.

1) For a unit tree $t = a,\ \ \mu(t) = 0$.

1) Let t be a tree whose root has r daughters $a_1, ...a_r, r > 0,\ m_1 = \mu(t(a_1)), ..., m_r = \mu(t(a_r)),$ and $m = max\{m_1, ..., m_r\}$. Then $m(t) = m + 1$ if there exist $1 \leq i < j \leq r$ such that $m_i = m_j = m$, and $m(t) = m$ otherwise.

Evidently, μ does not depend on the order of nodes. So this measure applies to c-trees as well as to D-trees and d-trees. For example, the D-trees in Fig. 2 and 6, have all density 1, the c-tree in Fig. 3, has density 2, the BD-grammars in Fig. 8, have both dependency density 1 as well as DT-grammars G_1, G_2 in Fig. 9. DT-grammar G_3 in Fig. 9, has dependency density 0.

In (Dikovskij A. 1972a) Dikovsky exposes several attractive properties of this measure (for example, its additivity with respect to tree composition), and shows that for each tree $t,\ \ m(t) \leq \log|t|$[27]. Then he proves that conformable c-trees and D-trees have the same order of density (Dikovskij A. 1972c).

Theorem 19. *If a c-tree t conforms with a D-tree d, then*

1) $\left\lceil \frac{\mu(t)}{2} \right\rceil \leq \mu(d) \leq \mu(t)$.

2) Both inequalities in 1) are optimal : the lower and upper bounds are attained on some pairs of conformable c-trees and D-trees.

Dikovsky proves that the density measure has very strong classification power. For example, from general complexity results obtained in (Dikovskij A. 1974; Dikovskij A. 1977) follows the existence of dense hierarchies of DTGs and of CF-languages of different dependency density.

Theorem 20. *There is an infinite class $\mathcal{F}$ of unbounded numeric functions below $\log n$ such that for each $f \in \mathcal{F}$, there exists a CF-language L_f of strict*

[26] A proof of this fact was communicated to one of the authors by G.Tseitin.

[27] $|t|$ is the size of t (e.g., the number of nodes).

dependency density f, i.e.,
1) for some DTG (BDG, LDG) G_f generating L_f, $\mu_{G_f} = \boldsymbol{O}(f)$, and
2) for each function g of lower order of magnitude than f, $L_f \in \mathcal{L}_\mu(f) \setminus \mathcal{L}_\mu(g)$.

We conjecture that in any natural language the dependency density is bounded by a small constant (2 or 3). Theoretically, this might lower the computational complexity of syntactic analysis of natural languages. In (Dikovskij A. 1972b) Dikovsky shows that imposing constant upper bounds on density function leads to a very well known family of languages : the cone of CF-languages generated by linear CF-languages.

Theorem 21. *$\mathcal{L}_\mu(\mathbf{const})$ is the smallest family of languages containing all linear CF-languages and closed under substitution.*

And in this family, the density measure serves well for infinite classifications on c-structure and dependency structure complexity, as it shows the following theorem implied by results of (Dikovskij A. 1972b; Dikovskij A. 1972c).

Theorem 22. *For each $k = 1, 2, 3, ...,$*
1) there exists a CF-language L_k of strict c-structure density k : it is generated by a CFG of c-structure density k and by no CFG of c-structure density $l < k$.
2) there exists a CF-language L_k^d of dependency density 2^k, which cannot be generated by any DTG (BDG, LDG) of dependency density $l < 2^{k-1}$.

5. Problems with dependency syntax

We will mention three main sources of problems with dependency syntax :

- dependency adequacy problems,
- dependency grammar expressivity problems,
- complexity problems.

1. Dependency adequacy problems. These are the problems of adequacy of dependency structure with respect to one or other linguistic theory of syntax. Mel'čuk (Mel'čuk I. 1988) groups the inadequacy criticisms under four headings : 'double dependency', 'mutual dependency', 'no dependency', and 'insufficient dependency', and for each group describes the factors leading to the inadequacies of this group, i.e, either confusion of different kinds of dependencies (morphological, semantic or syntactic), or ignoring dependency relation names. Meanwhile, we should note that there are syntactic constructions in each language, for which it is difficult to chose a unique head according to some non ad hoc criterion. In English this is e.g. the case with WH-clauses (*' [Which possible chain to select] has become the theme of his work '*), some prepositive nominal attribute groups (*' [the man in the street's] argument '*), some *'and'*-connected composite nominal groups (*' [John and Mary's] presentation '*), etc. In such cases a very natural solution is to treat the constructions **as one unit** and to analyze separately their inner and external dependencies.

In this manner one can treat not only continuous groups, but also discontinuous ones. So one arrives at hybrid dependency-constituent structures. Several syntactic formalisms of this kind were proposed (e.g. (Martem'janov J. 1970; Martem'janov J. 1971)). The most elaborated and well founded hybrid structure system was proposed by Gladkij (Gladkij A. 1969; Gladkij A. 1985). He proposes a specific type of syntactic structures which he calls *syntactic groups (sg-) systems* and defines axiomatically as follows.

A graph $\gamma =< C; \rightarrow >$ is an sg-system on a string w if C is a set of c-intervals of w and $\rightarrow$ is a binary relation on C, which meet the following conditions :

$\mathfrak{C}1.$ w and all unit intervals of w belong to C.
$\mathfrak{C}2.$ If $E_1, E_2 \in C$, then $E_1 \cap E_2 = \emptyset$, or $E_1 \subseteq E_2$, or else $E_2 \subseteq E_1$.
$\mathfrak{C}3.$ If $E_1, E_2 \in C$, then E_1 and E_2 do not interlace[28].
$\mathfrak{R}1.$ If $E_1 \rightarrow E_2$, then E_1, E_2 are immediate[29] subgroups of a group in C.
$\mathfrak{R}2.$ Relation $\rightarrow$ is loop-free.
$\mathfrak{R}3.$ If $E_1 \rightarrow E$ and $E_2 \rightarrow E$, then $E_1 = E_2$.
$\mathfrak{R}4.$ If $E_1 \rightarrow E_2$, then $E_1 \cup E_2$ does not interlace with E, for each $E \in C$.
$\mathfrak{R}5.$ If $E_1 \rightarrow E_2$ and $E_3 \rightarrow E_4$, then the sets $E_1 \cup E_2, E_3 \cup E_4$ do not interlace.

In Fig. 12, 13, we see two examples of syntactic groups graphs.

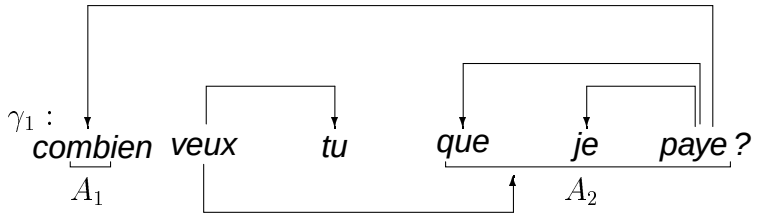


Figure 12.

In graph γ_1, the group $E_1 = A_1 \cup A_2$ depends on *'veux'*. γ_1 is an sg-system because it satisfies all axioms. In graph γ_2, the group $E_2 = B_1 \cup B_2$ depends on *'man'*. This dependency contradicts axiom $\mathfrak{R}4$. So γ_2 is not an sg-system.

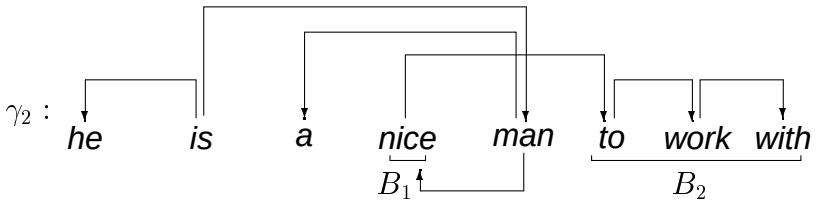


Figure 13.

In his book (Gladkij A. 1985), Gladkij demonstrates numerous examples of syntactic constructions in Russian, for which the sg-system representation simplifies grammatical description.

[28] Unions of c-intervals E_1, E_2 of w interlace if there exist $a, b \in E_1$ and $c, d \in E_2$ such that $a < c < b < d$ and $E_1 \cap E_2 = \emptyset$.

[29] In the same sense as for constituents.

2. Dependency grammar expressivity problems. These are problems specific to each class of dependency grammars in question. In fact, we have compared several classes of dependency grammars, and shown that non projectivity is the property which underlies the difference in expressive power of BDGs and DTGs. Non projective DTGs partially solve the problem of local non projective dependencies, for example, of noun-phrase shift or of prepositional phrase extraction. However, in many languages, for instance, in English, German and French, there are regular constructions, such as “raising", in which a dependency group is shifted unlimited distance up to one of its ancestors (e.g., WH-clause shift or elective constructions shift in English). DT-grammars might prove to be a good basis for developing a class of dependency grammars which cover such long distance shift constructions[30].

3. Complexity problems. Paradoxically, the expressivity of the dependency structure has caused hard complexity problems. The fact that dependencies are separable from word order presents an opportunity for linguists to define surface syntax in terms of individual binary dependency syntagmas, i.e. in terms of edges of dependency trees. For English, as far as we know, the most complete definition of this kind may be found in (Mel’čuk I. & Pertsov N. 1987). Such kind of syntax description is very attractive because, being complete, it can be very succinct. Meanwhile, computational complexity considerations show that such a “free" dependency grammar leads to brute-force complete-search syntactic analysis (see, for example, the paper (Neuhaus P. & Bröker N. 1997), where the analysis problem for such grammars is proven to be NP-complete). This is why some automatic translation systems based on “free" dependency grammar syntax description use filter based algorithms of syntactic analysis. However, this problem seems to be dramatized. For DT-grammars (without constraint of projectivity) the classical Early bottom-up algorithm gives the complexity of analysis $\boldsymbol{O}(n^3)$.

6. Conclusion

This survey is dedicated mostly to the research in mathematical foundations of dependency syntax in the USSR of 60ies-70ies. Meanwhile, during the same period very many interesting results were obtained in the linguistic theory of dependency based syntax and its applications. This might be the subject of another survey (by some other authors). Even so short as it is, this survey shows that the theory of dependencies is rich and fruitful, closely interrelated with the theory of grammars, tree grammars in particular, and at the same time well founded linguistically. Very importantly, it has not sunk into oblivion. In

[30] It is not difficult to see that in the particular class of DT-grammars we discuss in this review, the *ancestor conflict degree*, i.e. the maximal length of a path from an edge A to another edge D (a descendant of A), which covers A or interlaces incorrectly with A, is bounded by the maximum of this degree in the rules of the grammar. However a generalization of derivability leads to another class of DTGs, where this is already not the case, and which nevertheless has the same complexity of analysis.

the last four-five years we observe the growing interest of researchers in computational linguistics to dependency grammars, dependency based syntactic analysis, formal semantics based on dependencies etc. The new research in categorial grammars lays the bridge between dependency grammars and formal semantics (see e.g. (Abrusci V. *et al.* 1999; Retoré C. 1996)). The genre of this survey does not give us the opportunity to present these results.

7. Acknowledgments

We wish to thank our colleagues A. Gladkij, I. Mel'čuk, N. Pertsov, Ch. Retoré, Z. Shalyapina, and anonymous reviewers for their criticism, remarks and comments.

REFERENCES

ABRUSCI, V.M. ; FOUQUERÉ, Ch. ; VAUZEILLES, J. (1999) : "Tree Adjoining Grammars in a Fragment of the Lambek Calculus", *Computational Linguistics*, vol. 25, n ° 2, pp. 209–236.

AJDUKIEWICZ, K. (1935) : "Die syntaktische Konnexität", *Studia Philosophica*, n ° 1, pp. 1–27.

BAR-HILLEL, Y. ; GAIFMAN, H. ; SHAMIR, E. (1960) : "On categorial and phrase structure grammars", *Bull. Res. Council Israel*, vol. 9F, pp. 1–16.

BAR-HILLEL, Y. ; KASHER, A. ; SHAMIR, E. (1963) : *Measures of syntactic complexity*, Techn. Report n ° 13, Appl. Logic Branch, The Hebrew University of Jerusalem.

BAR-HILLEL, Y. (1953) : "A quasi-arithmetical notation for syntactic description", *Language*, vol. 29, n ° 1, pp. 47–58.

BELECKIJ, M.I. (1961) : "Model' jazyka algoritma grammatičeskogo analiza [A model of a language for an algoritm of grammatical analysis]", *in Tezisy Konferencii po obrabotke informacii, mašinnomu perevodu i avtomatičeskomu čteniju teksta [Proc. of the Conf. on Inform. Retriv., Automat. Translation and Automatic Interpretation of the Text]*, VINITI.

BELECKIJ, M.I. (1967) : "Beskontekstnye i dominacionnye grammatiki i svjazannye s nimi algoritmičeskie problemy [Context-free and dependency grammars, and some related algorithmic problems]", *Kibernetika [Cybernetics]*, n ° 4, pp. 90–97.

BLOOMFIELD, L. (1933) : *Language*, NY, Holt, Rinehart and Winston.

DIKOVSKIJ, A.Ja. ; MODINA, L.S. (1968) : "Minimizacija odnoj funkcii složnosti v klasse mp-avtomatov i kategorial'nye grammatiki složnosti tri [Minimization of a complexity function of pushdown automata and categorial grammars of complexity 3]", *Algebra i Logika [Algebra and Logics]*, vol. 7, n ° 3, pp. 23–37.

DIKOVSKIJ, A.Ja. ; MODINA, L.S. (1977) : “On Three Types of Unambiguity of Context-Free Languages”, *in Proc. of the 4th Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science,* A. Salomaa ; M. Steinby (ed.), pp. 193–205, Turku, Finland.

DIKOVSKIJ, A.Ja. (1970) : “Jazyki ograničennoj aktivnoj ëmkosti [Languages of bounded active capacity]”, *Doklady Akademii nauk SSSR*, vol. 192, n° 5, pp. 966–968, Eng. transl. : Soviet Math. Dokl. 1970, v.11, pp.748-750.

DIKOVSKIJ, A.Ja. (1972a) : “Gustota - mera složnosti vyvoda v kontekstnosvobodnoj grammatike [Density as a derivation complexity measure of cf-grammars]”, *Problemy peredači informacii [Problems of Information Transmission]*, vol. 8, n° 2, pp. 90–102.

DIKOVSKIJ, A.Ja. (1972b) : “Gustota dereva vyvoda i aktivnaja ëmkost’ grammatiki [On density and index of grammars]”, *Problemy peredači informacii [Problems of Information Transmission]*, vol. 8, n° 4, pp. 88–98.

DIKOVSKIJ, A.Ja. (1972c) : “O gustote dominacii [On domination density]”, *Doklady Akademii nauk SSSR*, vol. 204, n° 3, pp. 525–528, Eng. transl. : Soviet Math. Dokl. 1972, v.13, pp. 678-682.

DIKOVSKIJ, A.Ja. (1974) : “K obsčemu ponjatiju složnosti vyvoda v kontekstnosvobodnoj grammatike [On a general notion of complexity of derivation in a context-free grammar]”, *Doklady Akademii nauk SSSR*, vol. 214, n° 2, pp. 257–260, Eng. transl. : Soviet Math. Dokl. 1974, v.15, pp.98-102.

DIKOVSKIJ, A.Ja. (1977) : “K obščemu ponjatiju složnosti vyvoda i sintaksičeskogo opisanija v beskontekstnoj grammatike [A general notion of derivation complexity and syntactic structure in cf-grammar]”, *in Semiotika i Informatika [Semiotics and Informatics], Vol.9*, VINITI, pp. 83–105.

FITIALOV, S.Ja. (1962) : “O modelirovanii sintaksisa v strukturnoj lingvistike [On syntax representations in structural linguistics]”, *in Problemy strukturnoj lingvistiki [Problems of Structural Linguistics]*, Nauka, pp. 100–114.

FITIALOV, S.Ja. (1968) : “Ob èkvivalentnosti grammatik ns i grammatik zavisimostej [On the equivalence of ps-grammars and dependency grammars]”, *in Problemy strukturnoj lingvistiki [Problems of Structural Linguistics]*, Nauka, pp. 71–102.

GAIFMAN, H. (1961) : *Dependency systems and phrase structure systems*, Report p-2315, RAND Corp. Santa Monica (CA), Published in : Information and Control, 1965, v. 8, n° 3, pp. 304-337.

GLADKIJ, A.V. ; DIKOVSKIJ, A.Ja. (1972) : “Teorija formal’nyx grammatik [Theory of formal grammars]”, *in Itogi nauki [Scientific Surveys], Vol.10*, Moscow, VINITI.

GLADKIJ, A.V. ; MEL’ČUK, I.A. (1971) : “Grammatiki derev’ev. I. Opyt formalizacii preobrazovanij sintaksičeskix struktur estestvennogo jazyka [Tree

grammars. I. A formalism for syntactic transformations in natural languages]", *in Informacionnye voprosy semiotiki, lingvistiki i avtomatičeskogo perevoda [Informational Problems of Semiotics, Linguistics and Automatic Translation]*, pp. 16–41, Published In "Linguistics. An International Review", n ° 150. April 15, 1975, The Hague : Mouton, pp. 47-82.

GLADKIJ, A.V. ; MEL'ČUK, I.A. (1974) : "Grammatiki derev'ev. II. K postroeniju Δ-grammatiki dlja russkogo jazyka [Tree grammars. II. Towards a Δ-grammar for russian]", *in Informacionnye voprosy semiotiki, lingvistiki i avtomatičeskogo perevoda [Informational Problems of Semiotics, Linguistics and Automatic Translation]*, pp. 4–29.

GLADKIJ, A.V. ; MEL'ČUK, I.A. (1983) : *Elements of Mathematical Linguistics*, The Hague : Mouton.

GLADKIJ, A.V. (1966) : *Matematičeskaja lingvistika [Mathematical Linguistics]*, Novosibirskij Gosudarstvennyj Universitet [Novossibirsk State University].

GLADKIJ, A.V. (1969) : "Ob opisanii sintaksičeskoj struktury predloženija [On description of sentence syntactic structure]", *in Computational Linguistics, Vol.VII*, Budapest, pp. 21–44.

GLADKIJ, A.V. (1973) : *Formal'nye grammatiki i jazyki [Formal Grammars and Languages]*, Moscow, Nauka.

GLADKIJ, A.V. (1985) : *Sintaksičeskie struktury estestvennogo jazyka v avtomatizirovannyx sistemax obščenija [Syntactic Structures of Natural Languages in Automatic Interactive Systems]*, Moscow, Nauka.

GREIBACH, Sh.A. (1965) : "A new normal-form theorem for context-free phrase structure grammars", *Journ. Assoc. Computing Machinary*, vol. 12, n ° 1, pp. 42–52.

HARPER, K.E. ; HAYS, D.G. (1959) : "The Use of Machines in the Construction of a Grammar and Computer Programm for Structural Analysis", *in Proc. of the IFIP. Information Processing 1959*, pp. 188–194, Paris.

HARRIS, Z.S. (1961) : *Structural Linguistics*, Chicago, The University of Chicago Press.

HAYS, D.G. ; ZIEHE, T.W. (1960) : *Studies in Machine Translation-10 : Russian Sentence-Structure Determination*, Research memorandum RM-2538, The RAND Corporation.

HAYS, D.G. (1960) : *Grouping and dependency theories*, Research memorandum RM-2646, The RAND Corporation, Published in : Proc. of the National Symp. on Machine Translaion, Englewood Cliffs (N.Y.), 1961, pp. 258-266.

HAYS, D. (1961) : "Basic principles and technical variations in sentence structure determination", *in Information Theory,* C.Cherry (eds.), Washington, pp. 367–374.

IORDANSKAJA, L.N. (1963) : "O nekotoryx svojstvah pravil'noj sintaksičeskoj struktury [On some properties of correct syntactic structure]", *Voprosy yazykoznanija [The problems of linguistics]*, n ° 4, pp. 102–112.

IORDANSKAJA, L.N. (1964) : “Svojstva pravil’noj sintaksičeskoj struktury i algoritm eë obnaruženija (na materiale russkogo jazyka) [Properties of correct syntactic structure and an algorithm of its extraction]”, *in Problemy kibernetiki [The problems of cybernetics]*, Nauka, pp. 217–244.

JOSHI, A.K. ; LEVY, L.S. ; TAKAHASHI, M. (1975) : “Tree adjunct grammars”, *Journ. of Comput. and Syst. Sci.*, vol. 10, n ° 1, pp. 136–163.

LECERF, Y. (1960) : “Programme des conflits, modèle des conflits”, *Traduction Automatique*, vol. 1, n ° 4, pp. 11–18, n ° 5, pp. 17-36.

LECERF, Y. (1961) : “Une représentation algébrique de la structure des phrases dans diverses langues naturelles”, *C.R. Acad. Sci. Paris*, n ° 252, pp. 232–324.

LEŚNIEWSKI, S. (1929) : “Grundzüge eines neuen Systems der Grundlagen der Mathematik”, *Fundam. Math.*, vol. 14, pp. 1–81.

MARTEM’JANOV, Ju.S. (1970) : “K opisaniju teksta : jazyk valentno-junktivno-emfaznyx otnošenij [On text description : a language of valence-junction-emphasis relations], part 1”, *in Mašinnyj perevod i prikladnaja lingvistika [Automatic translation and applied linguistics], Vol.13*, Moscow, MGPIIJa.

MARTEM’JANOV, Ju.S. (1971) : “K opisaniju teksta : jazyk valentno-junktivno-emfaznyx otnošenij [On text description : a language of valence-junction-emphasis relations], part 2”, *in Mašinnyj perevod i prikladnaja lingvistika [Automatic translation and applied linguistics], Vol.14*, Moscow, MGPIIJa.

MEL’ČUK, I. ; PERTSOV, N.V. (1987) : *Surface Syntax of English. A Formal Model Within the Meaning-Text Framework*, Amsterdam/Philadelphia, John Benjamins Publishing Company.

MEL’ČUK, I. (1974) : *Opyt teorii lingvističeskix modelej "Smysl $\Longleftrightarrow$ Tekst" [Towards a Theory of Meaning-Text Linguistic Models]*, Moscow, Nauka.

MEL’ČUK, I. (1981) : “Meaning-Text Models : A Recent Trend in Soviet Linguistics”, *Annual Review of Antropology*, vol. 10, pp. 27–62.

MEL’ČUK, I. (1988) : *Dependency Syntax*, Albany, NY, SUNY Press.

MILLER, G.A. (1956) : “The magical number seven, plus or minus two : Some limits on our capacity for processing information”, *Psychological Review*, vol. 63.

MODINA, L.S. (1975a) : “Drevesnye grammatiki i jazyki [Dendrogrammars and dendrolanguages]”, *Kibernetika [Cybernetics]*, n ° 5, pp. 86–93.

MODINA, L.S. (1975b) : “On Some Formal Grammars Generating Dependency Trees”, *in Proc. of the MFCS’75, Lecture Notes in Computer Science*, pp. 326–329.

MODINA, L.S. (1976) : “O nekotoryx isčislenijax, poroždajuščix derev’ja podčinenija [On some calculus generating dependency trees]”, *Problemy peredači informacii [Problems of Information Transmission]*, vol. 12, n ° 2, pp. 83–94.

MODINA, L.S. (1978) : “O dominacionnoj neodnoznačnosti ks-jazykov [On dependency ambiguity of cf-languages]”, *Problemy peredači informacii [Problems of Information Transmission]*, vol. 14, n ° 2, pp. 86–98.

NEUHAUS, P. ; BRÖKER, N. (1997) : “The Complexity of Recognition of Linguistically Adequate Dependency Grammars”, *in Proc. of 35th ACL Annual Meeting and 8th Conf. of the ECACL*, pp. 337–343.

PADUČEVA, E.V. (1964) : “O sposobax predstavlenija sintaksičeskoj struktury predloženija [On some means of representing syntactic structure of sentences]”, *Voprosy yazykoznanija [The problems of linguistics]*, n ° 2, pp. 99–113.

RETORÉ, Ch. (1996) : “Calcul de Lambek et logique linéaire”, *T.A.L.*, vol. 37, n ° 2, pp. 39–70.

ROBINSON, Jane J. (1970) : “Dependency structures and transformational rules”, *Language*, vol. 46, n ° 2, pp. 259–285.

ROUNDS, W.C. (1970) : “Mappings and Grammars on Trees”, *Math.Syst.Theory.*, vol. 4, n ° 3, pp. 257–287.

SEVBO, I.P. (1981) : *Grafičeskoe predstavlenie sintaksičeskix struktur i stilističeskaja diagnostika [Graphical representation of syntactic structures and stylistic diagnostics]*, Kiev, Naukova Dumka.

TESNIÈRE, L. (1959) : *Éléments de syntaxe structurale*, Paris, Librairie C. Klincksieck.

THATCHER, J.W. (1970) : “Generalized Sequential Machine Maps”, *Comput. and Syst. Sci.*, vol. 4, n ° 4, pp. 339–367.

TSEITIN, G.S. ; ZASORINA, L.N. (1961) : “O vydelenii konfiguracij v russkom predloženii [On extraction of configurations in russian sentences]”, *in Doklady konferencii po obrabotke informacii, mašinnomu perevodu i avtomatičeskomu čteniju teksta [Proc. of the Conf. on Inform. Retriv., Automat. Translation and Automatic Interpretation of the Text], Vol.2*, Moscow.

TSEITIN, G.S. (1959) : “K voprosu o postroenii matematičeskix modelej jazyka [On the development of mathematical models of languages]”, *in Tezisy soveščanija po matematičeskoj lingvistike [Proc. of the Workshop on Mathematical Linguistics]*, Leningrad.

VARGHA, D. (1964) : “Yngve’s hypothesis and some problems of the mechanical analysis”, *in Computational Linguistics, Vol.III*, Budapest, pp. 47–74.

WELLS, R.S. (1947) : “Immediate Constituents”, *Language*, vol. 23, n ° 2, pp. 81–117.

YNGVE, V.H. (1960) : “A model and a hypothesis for language structure”, *in Proc. of Amer. Phil. Soc., Vol.104, n ° 5*, pp. 444–466.