Generalized Categorial Dependency Grammars

Michael Dekhtyar[1] and Alexander Dikovsky[2] *

[1] Dept. of Computer Science, Tver State University, Tver, Russia, 170000.
Michael.Dekhtyar@tversu.ru
[2] LINA CNRS 2729, Université de Nantes, 2, rue de la Houssinière BP 92208 F 44322 Nantes cedex 3 France. Alexandre.Dikovsky@univ-nantes.fr

To our dear Teacher Boris Trakhtenbrot

Abstract. Generalized Categorial Dependency Grammars (gCDG) studied in this paper are genuine categorial grammars expressing projective and discontinuous dependencies, stronger than CF-grammars and non-equivalent to mild context-sensitive grammars. We show that gCDG are parsed in polynomial time and enjoy good mathematical properties.

1 Introduction

Dependency grammars (DGs) are formal grammars assigning *dependency structures* to the sentences of the language they define. A dependency structure (DS) of a sentence is an oriented graph whose nodes are the words of the sentence and whose arcs are labelled with dependency names. In other words, they are structures on sentences in terms of various binary relations on words. If two words v_1 and v_2 are related by dependency d (denoted $v_1 \xrightarrow{d} v_2$), then v_1 is the *governor* and v_2 is the *subordinate*. Intuitively, the dependency d encodes constraints on lexical and grammatical features of v_1 and v_2, on their precedence, pronominalization, context, etc. which together mean that "v_1 licenses v_2" (see [24] for a detailed presentation). For instance, in the DS of the sentence *In the beginning was the Word* in Fig. 1, *was* $\xrightarrow{pred}$ *Word* stands for the predicative dependency between the copula *was* and the subject *Word*. From such

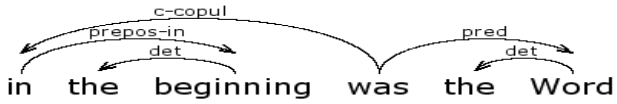
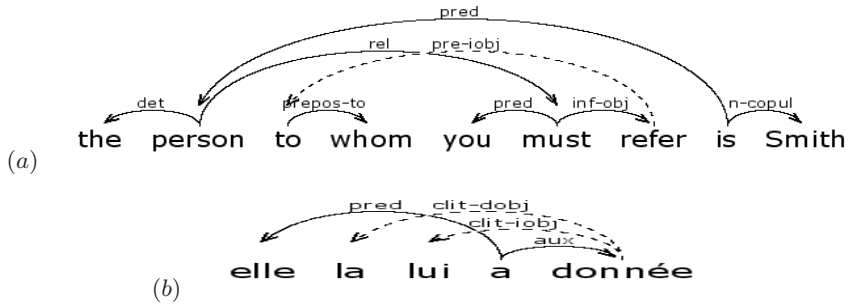


Fig. 1.

* This work was sponsored by the Russian Fundamental Studies Foundation (Grant 05-01-01006-a).

basic dependency relations some derived relations are defined, e.g. the dependency relation $v_1 \longrightarrow v_2 \equiv \exists d\ (v_1 \xrightarrow{d} v_2)$ and its reflexive-transitive closure (*dominance*) $v_1 \xrightarrow{*} v_2$. For instance, in the DS in Fig. 1, both occurrences of the article *the* are dominated by *was*: $was \xrightarrow{*} the_2$, $was \xrightarrow{*} the_5$, but only the first one is dominated by *in*: $in \xrightarrow{*} the_2$.

The idea of such explicit representation of syntactic relations in sentences is by far more ancient than that of the constituent structure and goes back at least to the early grammars of the Arabic language, which used the notions of governor and subordinate (Kitab al-Usul of Ibn al-Sarrang, (d. 928)). Modern theories of syntax use various DSs. Prevailing is the tradition, going back to L. Tesnire [29], to use only the tree-like DS: *dependency trees* (DTs). There are also approaches where general dependency structures are used (cf. [17, 28]). Sometimes (this is the case of [17]), it is due to combining in the same structure several relations of different nature, for instance, the surface syntactic relations and the co-reference relations. Another difference point is the word order (WO) included or not into the DSs. Some important properties of DSs cannot be expressed without the WO, first of all, *projectivity*. This property is defined in terms of the *projection* $D(v)$ of a word v in DS D of a sentence w: $D(v) = \{v' \in D \mid v \xrightarrow{*} v'\}$. D is *projective* if the projections of all words in w are **continuous intervals** of w. So the DS in Fig. 1 is projective. Meanwhile, non projective DSs are frequent in natural languages. E.g., both DTs in Fig. 2 are non projective. The non projectivity is



(French: *she gave it*$_{\text{FEM}}$ *to him*)

Fig. 2.

always due to *discontinuous* dependencies, i.e. the dependencies in which the governor v_g is separated from the subordinate v_s by a word not dominated by v_g (see [13] for more details). In Fig. 2, the discontinuous dependencies are represented by dotted arrows. When the dependencies are emancipated from the WO (cf. [6]), it is only done to define more exactly the WO constraints. Contrary to this, we suppose that the DSs are linearly ordered by the WO.

There is a great many definitions of DGs: from generating to constraint based (see [20, 21] for references and discussion). Our definition goes back to the early valency/precedence style definitions [16, 14] having much in common with those of the classical categorial grammars [1, 2]. Both are lexicalized, use syntactic types in the place of rewriting rules, naturally fit compositional semantic structures and are equivalent to CF-grammars if only the weak expressive power is concerned and the core syntax is considered. But as far as it concerns the strong expressive power, many fundamental differences appear between these formalisms. It is true that there is a simple translation from phrase structures with head selection to projective DTs and back (see [15, 27] or [13] for more details), which conforms with the direct simulation of core dependency grammars by the classical CGs [14]. Unfortunately, this technical resemblance does not preserve the intended syntactic types. The reason is that the syntactic functions corresponding to the dependencies are different from those of the heads in the syntagmatic structures originating from the X-bar theory [18]. Basically, the difference is that the type of a constituent head determines its syntagmatic (phrase) valencies, whereas a dependency represents a valency of the governor word in one subordinate word. It reflects its lexical and syntactic class, its position with respect to the governor, its semantic role, pronominalization, etc. (see [24] for more details). In particular, this means that the dependency types should be more numerous and specific than the syntagmatic ones and not prone to type raising. Essential distinctions are also in treating verb and noun modifiers, which in dependency surface syntax are *subordinate* and *iterated*. The canonical CGs' elimination rules imply dependencies from the functional type words to the argument type words. So, in the absence of type raising, the adjectives, whose canonical type in English is $[n/n]$, must govern the modified nouns and not vice versa as in DGs. This also explains the difference in treating the modifiers. In DGs (cf. [28, 23]) the modifiers are iterated and not recursed. Another important difference is that DTs, in contrast with phrase structures, naturally capture discontinuous surface word order. Rather expressive and complex extensions of CGs are needed to cope with the discontinuous and naturally oriented dependencies simulation (e.g. multi-modal extensions of Lambek calculus [25, 26]). Meanwhile, as it was shown in [9, 10], both can be naturally and feasibly expressed in DGs in terms of *polarized dependency valencies* controlled by a simple principle, which enables a discontinuous dependency between two closest words having the same valency with the opposite signs ("first available" (**FA**) principle).

Below, we study a class of *generalized categorial dependency grammars* established on the base of the **FA** principle. These grammars prove to be very expressive. At the same time, they are parsed in practical polynomial time and can be naturally linked with the underspecified semantics defined in [11] (this subject will be treated elsewhere).

2 Syntactic Types

Dependency type of a word (to be called *category*) represents its governor-subordinate valencies. There are two basic ideas of how to transform dependencies into categories. The first idea, proposed in [12], consists in decomposing each dependency $Gov \xrightarrow{d} Sub$ into two parts: Gov and Sub. The first becomes the *argument-type* d, whereas the second, the *value-type* d (see Fig. 3). Grouping together, for a word H, the value type f corresponding to the incoming

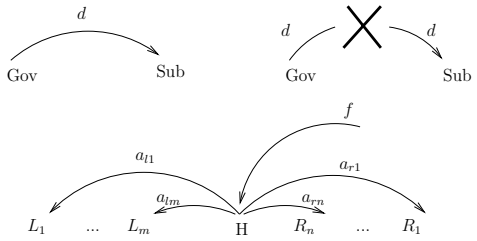


Fig. 3.

dependency f and the argument types corresponding to the outgoing left dependencies $a_{l1}, \ldots, a_{lm}$ and right dependencies $a_{r1}, \ldots, a_{rm}$ (in this order) we obtain the category $[a_{lm}\backslash ... \backslash a_{l1} \backslash f / a_{r1} / ... / a_{rn}]$ of H (denoted $H \mapsto [a_{lm}\backslash ... \backslash a_{l1} \backslash f / a_{r1} / ... / a_{rn}]$). For instance, the DT in Fig. 1 determines the types:

$$\begin{array}{ll} in \mapsto [c{-}copul/prepos{-}in], & the \mapsto [det], \\ beginning \mapsto [det\backslash prepos{-}in], & was \mapsto [c{-}copul\backslash S/pred], \\ Word \mapsto [det\backslash pred] & \end{array}$$

The second idea put forward in [9, 10], consists in interpreting discontinuous dependencies as polarized valencies using four *polarities*: left and right positive $\nwarrow, \nearrow$ and left and right negative $\swarrow, \searrow$. For each polarity v, there is the unique "dual" polarity $\breve{v}$: $\breve{\nwarrow} = \swarrow$, $\breve{\swarrow} = \nwarrow$, $\breve{\nearrow} = \searrow$, $\breve{\searrow} = \nearrow$. Intuitively, the argument type $\nwarrow d$ can be seen as the valency of a word whose subordinate through dependency d is situated *somewhere* on the left. The dual value type $\swarrow d$ can be seen as the valency of a word whose governor through the same dependency d is situated *somewhere* on the right. Together, the *paired* dual valencies $\swarrow d, \nwarrow d$ (respectively, $\nearrow d, \searrow d$) define the discontinuous dependency d. For instance, the DT in Fig. 2b determines the types:

$$\begin{array}{l} elle \mapsto [pred], \;\; la \mapsto [\swarrow clit{-}dobj], \\ lui \mapsto [\swarrow clit{-}iobj], \;\; a \mapsto [pred\backslash S/aux], \\ donne \mapsto [\nwarrow clit{-}iobj \backslash \nwarrow clit{-}dobj \backslash aux] \end{array}$$

Speaking about *generalized categories*, we will factor out from them the polarized subtypes. For instance, $[\nwarrow clit{-}iobj \backslash \nwarrow clit{-}dobj \backslash aux]$ and $[\swarrow clit{-}iobj]$

will become respectively $[aux]^{\nwarrow clit\text{–}iobj \nwarrow clit\text{–}dobj}$ and $[\varepsilon]^{\swarrow clit\text{–}iobj}$. Here is a definition of the generalized categories.

Definition 1. *Let* $\mathbf{C}$ *be a set of* elementary (dependency) categories. $S \in \mathbf{C}$ *is the selected category of sentences. For each* $d \in \mathbf{C}$, *the category* $d*$ *is* iterated.

All elementary categories and ε *are* neutral. *If a category* C *is neutral and a category* α *is elementary or iterated, then the categories* $[\alpha\backslash C]$ *and* $[C/\alpha]$ *are also neutral. There are no other neutral categories. The set of neutral categories over* $\mathbf{C}$ *is denoted* $nCat(\mathbf{C})$.

Polarized valencies are expressions $\swarrow d$, $\searrow d$, $\nwarrow d$, $\nearrow d$, *where* $d \in \mathbf{C}$. *The set of polarized valencies over* $\mathbf{C}$ *is denoted* $V(\mathbf{C})$. *Strings of valences* $P \in Pot(\mathbf{C}) =_{df} V(\mathbf{C})^*$ *are called* potentials. *A* generalized category *is either neutral or has the form* C^P, *where* P *is a potential and* C *is a neutral category. We will omit the empty potential. The set of generalized categories over* $\mathbf{C}$ *is denoted* $gCat(\mathbf{C})$.

We suppose that the constructors $\backslash$ and $/$ are associative. So every generalized category has the form $[\alpha_{lm}\backslash ... \backslash \alpha_{l1} \backslash f / \alpha_{r1} / ... / \alpha_{rn}]^P$, where $f \in \mathbf{C} \cup \{\varepsilon\}$, each α_{li} and α_{rj} is an elementary category $d \in \mathbf{C}$ or its iteration $d*$, $m, n \geq 0$ and $P \in Pot(\mathbf{C})$.

In [9] a simple and natural principle of pairing dual polarized valencies was proposed called **First Available** (**FA**)-principle: *the closest dual valences with the same name are paired.*

Definition 2. *An occurrence of dual polarized valencies* v *and* $\check{v}$ *in a potential* $P_1 v P \check{v} P_2$ *satisfies the* **FA***-principle if* P *has no occurrences of* v *and* $\check{v}$.

3 Generalized Categorial Dependency Grammar

Categorial dependency grammars are *lexicalized* in the same sense as the conventional categorial grammars: they have a few language non-specific rules constituting a dependency calculus and a language specific lexicon defining the words using dependency types.

Definition 3. *A* generalized categorial dependency grammar (gCDG) *is a system* $G = (W, \mathbf{C}, S, \delta)$, *where* W *is a finite set of words,* $\mathbf{C}$ *is a finite set of elementary categories containing the selected category* S, *and* δ *- called* lexicon *- is a finite substitution on* W *such that* $\delta(a) \subset gCat(\mathbf{C})$ *for each word* $a \in W$.

The generalized dependency calculus consists of the following rules.[3]

$\mathbf{L^1}$. $C^{P_1}[C\backslash\beta]^{P_2} \vdash [\beta]^{P_1 P_2}$
$\mathbf{I^1}$. $C^{P_1}[C^*\backslash\beta]^{P_2} \vdash [C^*\backslash\beta]^{P_1 P_2}$
$\mathbf{\Omega^1}$. $[C^*\backslash\beta]^P \vdash [\beta]^P$
$\mathbf{D^1}$. $\alpha^{P_1(\swarrow C)P(\nwarrow C)P_2} \vdash \alpha^{P_1 P P_2}$, if $(\swarrow C)P(\nwarrow C)$ satisfies **FA**

[3] We show only left argument rules. The right argument rules are symmetric.

Intuitively, the rule $\mathbf{L^1}$ corresponds to the classical elimination rule of categorial grammars. Eliminating the argument subtype C it constructs the (projective) dependency C in which the governor is the word with the functional type and the subordinate is the word with the argument type. At the same time, it concatenates the potentials of these types (if any). The rules $\mathbf{I^1}, \mathbf{\Omega^1}$ derive the iterated (projective) dependencies. $\mathbf{I^1}$, analogous to the rule $\mathbf{L^1}$, may derive $k > 0$ dependencies C and $\mathbf{\Omega^1}$ corresponds to the case $k = 0$. It is the rule $\mathbf{D^1}$ which derives discontinuous dependencies. It pairs and eliminates dual valencies $\swarrow C, \nwarrow C$ (or $\nearrow C, \searrow C$) and creates the discontinuous dependency C between the words whose types have these polarized valencies. This calculus naturally induces the immediate provability relation $\vdash$ on the strings of generalized dependency types $\Gamma_1 \vdash \Gamma_2$ underlying the following definition of languages.

Definition 4. *For a gCDG $G = (W, \mathbf{C}, S, \delta)$, let $G(D, w)$ denote the relation: D is the DS of a sentence w constructed in the course of a proof $\Gamma \vdash S$ for some $\Gamma \in \delta(w)$. In particular, we will use notation $w = w(D)$ for the DS D of w. The* DS-language *generated by G is the set of dependency structures*

$$\Delta(G) =_{df} \{D \mid \exists w\ G(D, w)\}$$

and the language *generated by G is the set of sentences*

$$L(G) =_{df} \{w \mid \exists D\ G(D, w)\}.$$

$\mathcal{D}(gCDG)$ and $\mathcal{L}(gCDG)$ will denote the families of DS-languages and languages generated by these grammars.

Example 1. For instance, in the gCDG G_{abc} :

$$a \mapsto A^{\swarrow A}, [A\backslash A]^{\swarrow A},\ b \mapsto [B/C]^{\nwarrow A}, [A\backslash S/C]^{\nwarrow A},\ c \mapsto C, [B\backslash C],$$

$G_{abc}(D^{(3)}, a^3b^3c^3)$ holds for the DS $D^{(3)}$ in Fig. 4 and the string $a^3b^3c^3$ due to the types assignment

$$a^3b^3c^3 \mapsto A^{\swarrow A}[A\backslash A]^{\swarrow A}[A\backslash A]^{\swarrow A}[A\backslash S/C]^{\nwarrow A}[B/C]^{\nwarrow A}[B/C]^{\nwarrow A}C[B\backslash C][B\backslash C]$$

and the proof in Fig. 5.

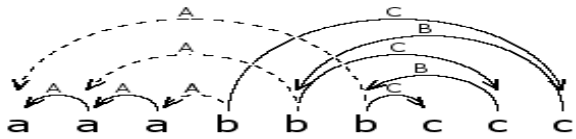


Fig. 4.

It is not difficult to prove:

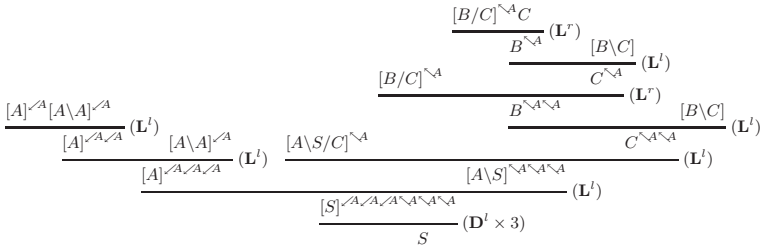


Fig. 5.

Proposition 1. $L(G_{abc}) = \{a^n b^n c^n \mid n > 0\}$.

Remark 1. It should be noted that a type assignment to a string may have multiple correctness proofs. As a consequence, even a *rigid gCDG*, i.e. a gCDG assigning one type per word, may generate various DSs for the same string, as it is shown in the next example.

Example 2. The rigid gCDG $G_r : \; x \mapsto [S/S], \; y \mapsto S, \; z \mapsto [S\backslash S]$ generating the regular language x^*yz^*, has two different proofs for $[S/S]S[S\backslash S] \vdash S$. As a result, the string xyz has the two DTs shown in Fig. 6.

Fig. 6.

An important particularity of gCDG is the property of independence of neutral and polarized valencies in the proofs, expressed using two *projections* of generalized categories.

Definition 5. Local *and* valency *projections* $\|\gamma\|_l$, $\|\gamma\|_v$ *are defined as follows:*
1. $\|\varepsilon\|_l = \|\varepsilon\|_v = \varepsilon$; $\|\alpha\gamma\|_l = \|\alpha\|_l\|\gamma\|_l$ *and* $\|\alpha\gamma\|_v = \|\alpha\|_v\|\gamma\|_v$ *for* $\alpha \in gCat(\mathbf{C})$ *and* $\gamma \in gCat(\mathbf{C})^*$.
2. $\|C^P\|_l = C$ *and* $\|C^P\|_v = P$ *for* $C^P \in gCAT(\mathbf{C})$.

To speak about "well-bracketing" of potentials, we interpret $\swarrow d$ and $\nearrow d$ as *left brackets* and $\nwarrow d$ and $\searrow d$ as *right brackets*. The sets of all left and right bracket valencies are denoted $V^l(\mathbf{C})$ and $V^r(\mathbf{C})$. $V(\mathbf{C}) =_{df} V^l(\mathbf{C}) \cup V^r(\mathbf{C})$.

Definition 6. *Pairs* $(\alpha, \breve{\alpha})$ *are called* correct. *For a dependency* d *and a potential* P, *let* $P \restriction d$ *be the result of deleting the occurrences of all valencies but* $\swarrow d$,

$\nearrow d$, $\nwarrow d$ *and* $\searrow d$. *Then* P *is* balanced *if* $P \restriction d$ *is well bracketed in the usual sense for every* d.

This property can be incrementally checked using the following values.

Definition 7. *For a (neutral or polarized) valency* v *and a category projection* γ, $|\gamma|_v$ *will denote the number of occurrences of* v *in* γ. *For a potential* P, *a left-bracket valency* $v \in V^l(\mathbf{C})$, *and the dual right-bracket valency* $\breve{v} \in V^r(\mathbf{C})$,

$$\Delta_v(P) = max\{|P'|_v - |P'|_{\breve{v}} \mid P' \textit{ is a } \mathsf{suffix} \textit{ of } P\}$$
$$\Delta_{\breve{v}}(P) = max\{|P'|_{\breve{v}} - |P'|_v \mid P' \textit{ is a } \mathsf{prefix} \textit{ of } P\},$$

express respectively the deficit *of right and left* $v-$*brackets in* P *(i.e. the maximal number of right and left bracket* v*-valencies which need to be added to* P *on the right (left) so that it became balanced.*[4]

The following facts are easy to prove:

Lemma 1. *1. A potential* P *is* balanced *iff* $\sum_{v \in V(\mathbf{C})} \Delta_v(P) = 0$.

2. For all potentials P_1, P_2, *and every* $v \in V^l(\mathbf{C}), \breve{v} \in V^r(\mathbf{C})$,

$$\Delta_v(P_1P_2) = \Delta_v(P_2) + max\{\Delta_v(P_1) - \Delta_{\breve{v}}(P_2), 0\},$$
$$\Delta_{\breve{v}}(P_1P_2) = \Delta_{\breve{v}}(P_1) + max\{\Delta_{\breve{v}}(P_2) - \Delta_v(P_1), 0\}$$

3. A potential P *is balanced iff for every category* α^P *there is a proof* $\alpha^P \vdash \alpha$ *using only the rules* $\mathbf{D}^l$ *and* $\mathbf{D}^r$.

Finally, we will denote by $\mathbf{c}$ the projective core of the generalized dependency calculus, consisting of the rules $\mathbf{L}$, $\mathbf{I}$ and $\mathbf{\Omega}$. $\vdash_{\mathbf{c}}$ will denote the provability relation in this sub-calculus. Now we can state the *property of projections independence.*

Theorem 1. *Let* $G = (W, \mathbf{C}, S, \delta)$ *be a gCDG.* $x \in L(G)$ *iff there is a string of categories* $\gamma \in \delta(x)$ *such that:*
1. $\|\gamma\|_l \vdash_{\mathbf{c}} S$,
2. $\|\gamma\|_v$ *is balanced.*

Proof. The theorem is proved by induction on the proof length. We will prove ($\Rightarrow$), the inverse being similar. Let $x \in L(G)$ due to an assignment $\delta : x \mapsto \gamma \in gCat(\mathbf{C})^*$ and a proof $\gamma \vdash S$. Let n be the length of this proof.
I. $n = 0$. Then $\gamma = \|\gamma\|_l = S$, $\|\gamma\|_v = \varepsilon$ and the statement is trivially true.
II. $n > 0$. Then the proof has the form $\gamma \vdash^R \gamma' \vdash S$, where R is the first rule applied in the proof.
Case 1. $R = \mathbf{D}$. In this case, R does not affect the local projection. So $\|\gamma\|_l = \|\gamma'\|_l$ and, therefore, $\|\gamma\|_l \vdash_{\mathbf{c}} S$ by induction hypothesis. On the other hand, $\|\gamma'\|_v$ results from $\|\gamma\|_v$ by elimination of a correct pair of polarized valencies satisfying the **FA**-principle. This means that $\|\gamma'\|_v$ is balanced iff $\|\gamma\|_v$ is so.
Case 2. $R \neq \mathbf{D}$. In this case, R does not affect the valency projection. So $\|\gamma\|_v = \|\gamma'\|_v$ and, therefore, $\|\gamma\|_v$ is balanced. On the other hand, $\gamma' \vdash S$ implies $\|\gamma'\|_l \vdash_{\mathbf{c}} S$ by induction hypothesis. So $\|\gamma\|_l \vdash^R \|\gamma'\|_l \vdash_{\mathbf{c}} S$. □

[4] Having in mind that there is $P' = \varepsilon$, the values $\Delta_{\breve{v}}(P)$ and $\Delta_v(P)$ are non-negative.

4 Expressive Power of GCDG

gCDG are very expressive. The Example 1 shows that they can generate non-CF languages. In fact, they have the same weak expressive power as the *Dependency Structure Grammars* (DSG), a class of generating rule based dependency grammars introduced in [9, 10] and simplified and studied in [3]. Below we cite the key definitions from [3].

The DSG use generalized DS over a mixed vocabulary of *terminals* W and *nonterminals* N. In these DS, one connected component [5] is selected as *head component* and some node in this component is selected as *DS head*. We will call *headed* the DS with such selection (hDS). In the two-component hDS in Fig. 7, the underlined node is head. The following composition $D[v\backslash D_1]$ (and

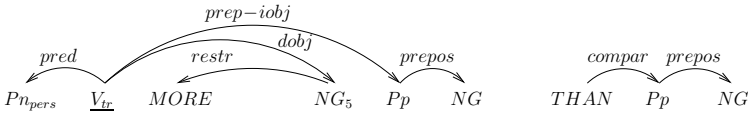


Fig. 7.

simultaneous composition $D[v_1, \ldots, v_n\backslash D_1, \ldots, D_n]$) is defined on hDS.

Definition 8. *Let $\delta_1 = \{\underline{D_0}, D_1, \ldots, D_k\}$ be a hDS. Let a nonterminal A have an occurrence in δ_1: $w(\delta_1) = xAy$ and δ_2 be a hDS with the head n_0. Then the* composition *of δ_2 into δ_1 in the selected occurrence of A, denoted $\delta_1[A\backslash\delta_2]$, is the hDS δ resulting from the union of δ_1 and δ_2 by unifying A and n_0 and by defining the order and labeling by the string substitution of $w(\delta_2)$ in the place of A in $w(\delta_1)$. Formally:*
1. $nodes(\delta) =_{df} (nodes(\delta_1) - \{A\}) \cup nodes(\delta_2)$.
2. $arcs(\delta) =_{df} arcs(\delta_2) \cup (\ arcs(\delta_1) - \{d \in arcs(\delta_1) \| \exists n(d = (A, n) \vee d = (n, A))\}\ ) \cup \{(n_0, n) \| \exists n((A, n) \in arcs(\delta_1))\} \cup \{(n, n_0) \| \exists n((n, A) \in arcs(\delta_1))\}$.
3. The order of $nodes(\delta)$ is uniquely defined by equation $w(\delta) = xw(\delta_2)y$.

In Fig. 8 is shown an example of such composition.[6] The FA-principle is used in DSG in the form of valency neutralization:

Definition 9. *For potentials $\Gamma = \Gamma_1 v \Gamma_2 \breve{v} \Gamma_3$ and $\Gamma' = \Gamma_1 \Gamma_2 \Gamma_3$ such that $v = (\nearrow A)$, $\breve{v} = (\searrow A)$ or $v = (\swarrow A), \breve{v} = (\nwarrow A)$, v is* neutralized *by $\breve{v}$ in Γ (denoted $\Gamma \twoheadrightarrow_{FA} \Gamma'$) if Γ_2 has no occurrences of v and $\breve{v}$. This reduction of potentials $\twoheadrightarrow_{FA}$ is terminal and confluent. So each potential Γ has a unique FA-normal form denoted $[\Gamma]_{FA}$. The* product $\odot$ *of potentials defined by: $\Gamma_1 \odot \Gamma_2 =_{df} [\Gamma_1 \Gamma_2]_{FA}$ is clearly associative. So we obtain the* monoid of potentials $\mathbf{P} = (Pot(\mathbf{C}), \odot)$ *with the unit ε.*

[5] Slightly abusing the standard graph-theoretic terminology, we call *connected component* of a DS D any its maximal subgraph corresponding to connected components of the *non-oriented* graph resulting from D after cancellation of its arcs' orientation.
[6] We use nonterminals $label(v)$ in the place of v when no conflicts.

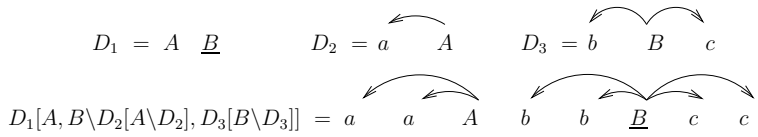


Fig. 8.

Definition 10. *A* **Dependency Structure Grammar (DSG)** *G has the* rules $r = (A \to D)$ *with* $A \in N$ *and hDS D with assigned potentials:* $[\Gamma_X^L]X[\Gamma_X^R]$ *(the left and right potentials* Γ_X^L *and* Γ_X^R *may be assigned to each nonterminal X in D* [7]*).*

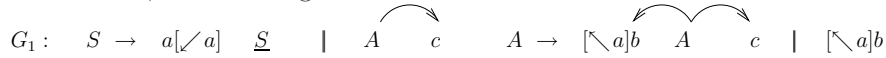
Derivation trees of G result from the derivation trees T of the cf-grammar $\{A \to w(D) \parallel A \to D \in G\}$ *by defining potentials* $\pi(T, n)$ *of nodes n :*
1. $\pi(T, n) = \varepsilon$ *for every terminal node n;*
2. $\pi(T, n) = \Gamma_1 \odot \ldots \odot \Gamma_k$*, for every node n with sons* $n_1, ..., n_k$ *derived by rule* $r = (A \to D)$*, in which* $w(D) = X_1...X_k$ *and* $\Gamma_i = \Gamma_i^L \odot \pi(T, n_i) \odot \Gamma_i^R$*, where* $[\Gamma_i^L]X_i[\Gamma_i^R]$ *are the rule potential assignments. A hDS is generated in the node n by the composition:* $hDS(T, n) = D[X_1 \ldots X_k \backslash hDS(T, n_1), \ldots, hDS(T, n_k)]$*. Every pair of dual valencies neutralized at this step corresponds to a discontinuous dependency added to this hDS.*

A derivation tree T is complete *if the potential of its root S is neutral:* $\pi(T, S) = \varepsilon$*. We set* $G(D, w)$ *if there is a complete derivation tree T of G from the axiom S such that* $D = hDS(T, S)$ *and* $w = w(D)$*.*

$\Delta(G) = \{D \mid \exists w \in W^+ \ G(D, w)\}$ *is the* DS-language *generated by G.*

$L(G) = \{w \in W^+ \mid \exists D \ G(D, w)\}$ *is the* language *generated by G.*

For instance, the following four-rule DSG:



generates the language $L(G_1) = \{a^n b^n c^n \| n > 0\}$. Its complete derivation tree of the string $a^3b^3c^3$ is shown in Fig. 9.

Clearly, $\mathcal{L}(CF) \subseteq \mathcal{L}(DSG)$. So this example shows that $\mathcal{L}(CF) \subsetneq \mathcal{L}(DSG)$. In [3] it is shown that DSG have Greibach normal form. Using this fact, it is shown that $\mathcal{L}(DSG) \subseteq \mathcal{L}(gCDG)$.[8] On the other hand, it is also proved that $\mathcal{D}(gCDG) \subseteq \mathcal{D}(DSG)$. In particular, this means that the gCDG and the DSG have the same weak generative power:

Theorem 2. *[3]* $\mathcal{L}(CF) \subsetneq \mathcal{L}(gCDG) = \mathcal{L}(DSG)$.

In [9] a measure of discontinuity σ was defined which will be called *valency deficit*. Intuitively, its value is the maximal potential size in a derivation. For instance, for the gCDG it is defined as follows.

[7] For instance, $A \to [\searrow d_1]\underline{B}[\nearrow d_2] \ C$ denotes the rule $A \to \underline{B} \ C$ with assignment $[\searrow d_1]B[\nearrow d_2]$. We omit empty potentials.

[8] In [3] is used an equivalent notational variant of gCDGs.

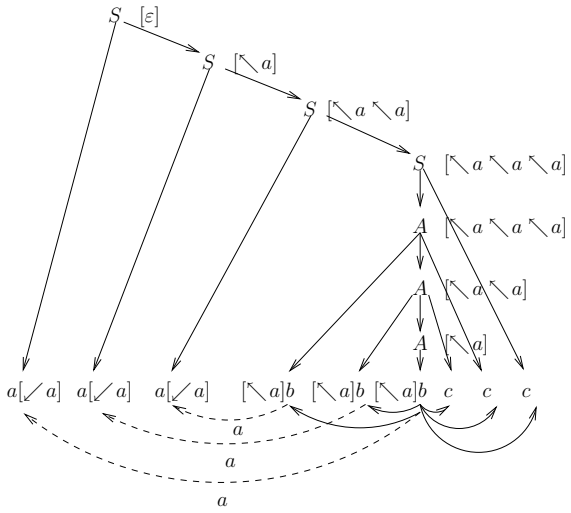


Fig. 9.

Definition 11. *Let $G = (W, \mathbf{C}, S, \delta)$ be a gCDG. For a proof $p = (\Gamma \vdash S)$, where $\Gamma \in \delta(w)$ and $w \in W^+$, its* valency deficit *$\sigma(\Gamma, p)$ is the maximal size of a potential used in p. $\sigma_G(w)$ is the minimal value of $\sigma(\Gamma, p)$ among all $\Gamma \in \delta(w)$. Finally, $\sigma_G(n) = max\{\sigma_G(w) \parallel |w| \leq n\}$.*

The examples of gCDG G_{abc} and DSG G_1 show that the valency deficit of these grammars cannot be bounded by a constant. As it is shown in [9], the dependency grammars with bounded valency deficit generate CF-languages. This theorem can be easily extended to gCDG and DSG. Let $\mathcal{L}^{\sigma<const}(gCDG)$ and $\mathcal{L}^{\sigma<const}(DSG)$ denote the classes of languages generated respectively by gCDG and DSG with bounded valency deficit.

Theorem 3. $\mathcal{L}^{\sigma<const}(gCDG) = \mathcal{L}^{\sigma<const}(DSG) = \mathcal{L}(CF)$.

Let us consider some more examples.

Example 3. Let $W = \{a_1, \ldots, a_m\}$ and $L^{(m)} = \{a_1^n a_2^n ... a_m^n \parallel n \geq 1\}$. Let us consider the gCDG

$$G^{(m)} : \begin{cases} a_1 \ \mapsto [S/A_1]^{\nearrow A_2}, [A_1/A_1]^{\nearrow A_2}, [A_1/A_2]^{\nearrow A_2}, \\ \ldots \ \ \ldots \ \ \ldots \\ a_i \ \mapsto [A_i/A_i]^{\nwarrow A_i \nearrow A_{i+1}}, [A_i/A_{i+1}]^{\nwarrow A_i \nearrow A_{i+1}}, 2 \leq i < m, \\ \ldots \ \ \ldots \ \ \ldots \\ a_m \mapsto [A_m/A_m]^{\nwarrow A_m}, [A_m]^{\nwarrow A_m} \end{cases}$$

It is not difficult to prove the proposition

Proposition 2. *$L(G^{(m)}) = L^{(m)}$ for all $m \geq 2$.*

Meanwhile, as it is well known, the languages $L^{(m)}$ are mild context sensitive and cannot be generated by basic TAGs starting from $m > 4$ (see [19]).

Example 4. Let us consider the language MIX consisting of all permutations of the strings $a^n b^n c^n, n > 0 : MIX = \{w \in \{a,b,c\}^+ \parallel |w|_a = |w|_b = |w|_c\}$. Emmon Bach conjectures that MIX is not a mild CS language. At the same time, this language is generated by the following gCDG:

gCDG G_{MIX}		
left	right	middle
$a \mapsto [S]^{\backslash B \backslash C}$ $a \mapsto [S \setminus S]^{\nwarrow B \backslash C}$	$a \mapsto [S]^{/C /B}$ $a \mapsto [S \setminus S]^{\nearrow C /B}$	$a \mapsto [S]^{\backslash B / C}, [S]^{\backslash C / B}$ $a \mapsto [S \setminus S]^{\nwarrow B / C}, [S \setminus S]^{\nwarrow C \nearrow B}$
$b \mapsto [\varepsilon]^{\swarrow B}$	$b \mapsto [\varepsilon]^{\backslash B}$	
$c \mapsto [\varepsilon]^{\swarrow C}$	$c \mapsto [\varepsilon]^{\backslash C}$	

Proposition 3. *[3]. $L(G_{MIX}) = MIX$.*

As it is well known, the copy language $L_{copy} = \{ww \parallel w \in \{a,b\}^+\}$ is generated by a basic TAG. On the other hand, it is conjectured in [12, 3] that L_{copy} cannot be generated by gCDG and DSG. As we will see below, the gCDG-language are parsed in polynomial time. This means that this family of grammars represents an interesting alternative for the mild CS grammars (see the diagram in Fig. 10 presenting a comparison of the two families in weak generative power). The

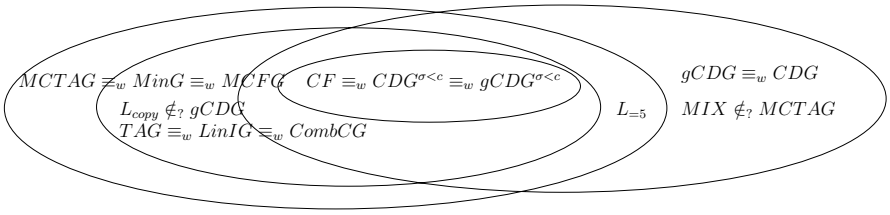


Fig. 10.

gCDG-languages have good operation closure properties. In particular, they form an AFL. To show this fact, we need some preliminary propositions.

Lemma 2. *For each gCDG G there is a weakly equivalent gCDG G' in which the axiom type S is not an argument subtype of a category.*

Proof. Otherwise, just add a new axiom S' and double the categories $[\alpha\backslash S/\beta]$ with new categories $[\alpha\backslash S'/\beta]$. □

Lemma 3. *For each gCDG G there is a weakly equivalent gCDG G' in which there are no categories with empty value type: $[\alpha\backslash\varepsilon/\beta]^P$.*

Proof. If there is one: $t = [\alpha\backslash\varepsilon/\beta]^P$, then add a new elementary type d_t, replace t in all type assignments with the new category $t' = [\alpha\backslash d_t/\beta]^P$ and then, in

the resulting grammar, substitute the new category $[d_t*\backslash A_l\backslash\ldots A_1\backslash d_t*\backslash V/d_t*/B_k\ldots/B_1/d_t*]^{P_1}$ for each category $[A_l\backslash\ldots A_1\backslash V/B_k\ldots/B_1]^{P_1}$. Regardless of the fact that the resulting gCDG is greater than G, it is weakly equivalent to G and has one empty value type less. □

Theorem 4. *The family $\mathcal{L}(gCDG)$ is an AFL.*

Proof. We suppose that gCDGs satisfy the conditions of Lemmas 2,3.

1. $\mathcal{L}(gCDG)$ is closed under ε-free homomorphisms. Let $G=(W,\mathbf{C},S,\delta)$ be a gCDG with $W=\{a_1,\ldots,a_n\}$ and $h:W\to X^+$ be a homomorphism such that $h(a_i)=x_{i0}...x_{im_i}, m_i\geq 0, x_{ij}\in X$, for all $0\leq j\leq m_i, 1\leq i\leq n$. The new gCDG G_h keeps all elementary types of G including S which is also its axiom. Besides them, it has a new elementary type d_{ij} for all $0\leq j\leq m_i, 1\leq i\leq n$. Its lexicon δ_h is defined as follows: if $h(a_i)=x_{i0}...x_{im_i}$, then for every category $\alpha\in\delta(a_i)$, it has the assignment $\delta_h: x_{i0}\mapsto[\alpha/d_{im_i}/.../d_{i1}]$. In particular, $\delta_h: x_{i0}\mapsto\alpha$, if $m_i=0$. Besides that, there are also the assignments $\delta_h: x_{ij}\mapsto d_{ij}$ for all $1\leq j\leq m_i, 1\leq i\leq n$. Clearly, $L(G_h)=h(L(G))$. □

2. $\mathcal{L}(gCDG)$ is closed under the inverses of homomorphisms. First of all, let us remark that to prove this proposition it suffices to prove it for the homomorphisms $h:X\to W^*$, $X\cap W=\emptyset$, differing from a bijection $h:X\leftrightarrow W$ by no more than one assignment which is either of the form $h(x)=ab$, $a,b\in W$, or of the form $h(x)=\varepsilon$. Let $G=(W,\mathbf{C},S,\delta)$ be the original gCDG and gCDG $G_{h^{-1}}=(X,\mathbf{C}_1,S,\delta_{h^{-1}})$ the gCDG to construct.

2.1. Let $h(x)=ab$, $a,b\in W$, $C_1\in\delta(a)$ and $C_2\in\delta(b)$. Then $\delta_{h^{-1}}=\delta\cup\delta_x$, where δ_x is defined below depending on the form of categories C_1,C_2. The following five cases are possible for some $u,v\in\mathbf{C}$:

2.1.(i). $C_1=[\alpha/u]^{P_1}$, $C_2=[v/\beta]^{P_2}$.
2.1.(ii). $C_1=[\alpha\backslash u]^{P_1}$, $C_2=[v\backslash\beta]^{P_2}$. Symmetric.
2.1.(iii). $C_1=[\alpha/u*]^{P_1}$, $C_2=[v/\beta]^{P_2}$.
2.1.(iv). $C_1=[\alpha\backslash u]^{P_1}$, $C_2=[v*\backslash\beta]^{P_2}$. Symmetric.
2.1.(v). $C_1=[\alpha\backslash u]^{P_1}$, $C_2=[v/\beta]^{P_2}$.

Construction of δ_x:

2.1.(i). In this case, are added to $\mathbf{C}_1$ the new elementary types d_v for all elementary types $d\in\mathbf{C}$ and is added to δ_x the assignment:
$x\mapsto[\alpha/\beta]^{P_1P_2}$ if $u=v$, and whatever are u,v ($u=v$ included), are also added the following assignments:
$x\mapsto[\alpha/u_v/\beta]^{P_1P_2}$,
$y\mapsto[\alpha'\backslash d_v/\beta']^P$ for each assignment $y\mapsto[\alpha'\backslash v\backslash d/\beta']^P\in\delta$,
$y\mapsto[\alpha'\backslash e_v\backslash d_v/\beta']^P$ for each assignment $y\mapsto[\alpha'\backslash e\backslash d/\beta']^P\in\delta$.

2.1.(iii). The only difference with the preceding case is in the form of the second assignment:
$x\mapsto[\alpha/u_v*/\beta]^{P_1P_2}$.

2.1.(v). In this case, three subsets are added to δ : $\delta_{x,fork}$, $\delta_{x,left}$ and $\delta_{x,right}$. We will construct $\delta_{x,fork}$ and $\delta_{x,right}$ ($\delta_{x,left}$ is symmetric to $\delta_{x,right}$).

Construction of $\delta_{x,fork}$:

Are added to $\mathbf{C}_1$ new elementary types: f_{lr}, $d_{a(b)}$ and $d_{(a)b}$ for all elementary

types $l, r, d \in \mathbf{C}$, and are added the following assignments:
$x \mapsto [\alpha\backslash l_{a(b)}\backslash f_{lr}/r_{(a)b}/\beta]^{P_1P_2}$ for all $l, r \in \mathbf{C}$,
$y \mapsto [\alpha'\backslash d_{a(b)}/\beta']^P$ for each assignment $y \mapsto [\alpha'\backslash d/u/\beta']^P \in \delta$,
$y \mapsto [\alpha'\backslash d_{(a)b}/\beta']^P$ for each assignment $y \mapsto [\alpha'\backslash v\backslash d/\beta']^P \in \delta$,
$y \mapsto [\alpha'\backslash d_{a(b)}/e_{a(b)}/\beta']^P$ for each assignment $y \mapsto [\alpha'\backslash d/e/\beta']^P \in \delta$,
$y \mapsto [\alpha'\backslash e_{(a)b}\backslash d_{(a)b}/\beta']^P$ for each assignment $y \mapsto [\alpha'\backslash e\backslash d/\beta']^P \in \delta$,
$y \mapsto [\alpha'/f_{lr}/\beta']^P$ for each assignment $y \mapsto [\alpha'/r/l/\beta']^P \in \delta$, where $\alpha' \neq \varepsilon$,
$y \mapsto [\alpha'\backslash f_{lr}\backslash\beta']^P$ for each assignment $y \mapsto [\alpha'\backslash r\backslash l\backslash\beta']^P \in \delta$, where $\beta' \neq \varepsilon$,
$y \mapsto [\alpha'/f_{dd}*/\beta']^P$ for each assignment $y \mapsto [\alpha'/d*/\beta']^P \in \delta$, where $\alpha' \neq \varepsilon$ and $d = l = r$,
$y \mapsto [\alpha'\backslash f_{dd}*\backslash\beta']^P$ for each assignment $y \mapsto [\alpha'\backslash d*\backslash\beta']^P \in \delta$, where $\beta' \neq \varepsilon$ and $d = l = r$.
Construction of $\delta_{x,right}$:
For all $d \in \mathbf{C}$, are added to $\mathbf{C}_1$ new elementary types: $d_{(\varepsilon)ab}$ and $d_{(\varepsilon)a|b}$ (symmetric types $d_{ab(\varepsilon)}$ and $d_{a|b(\varepsilon)}$ for $\delta_{x,left}$), and are added the following assignments:
$x \mapsto [\alpha\backslash d_{(\varepsilon)ab}/\beta]^{P_1P_2}$ for all $d \in \mathbf{C}$ ($x \mapsto [\alpha\backslash d_{ab(\varepsilon)}/\beta]^{P_1P_2}$ for $\delta_{x,left}$),
$y \mapsto [\alpha'\backslash d_{(\varepsilon)ab}/\beta']^P$ for each assignment $y \mapsto [\alpha'\backslash v\backslash u\backslash d/\beta']^P \in \delta$,
$y \mapsto [\alpha'\backslash e_{(\varepsilon)ab}\backslash d_{(\varepsilon)ab}/\beta']^P$ for each assignment $y \mapsto [\alpha'\backslash e\backslash d/\beta']^P \in \delta$,
$y \mapsto [\alpha'\backslash d_{(\varepsilon)a|b}/\beta']^P$ for each assignment $y \mapsto [\alpha'\backslash v\backslash d/\beta']^P \in \delta$,
$y \mapsto [\alpha'\backslash e_{(\varepsilon)a|b}\backslash d_{(\varepsilon)a|b}/\beta']^P$ for each assignment $y \mapsto [\alpha'\backslash e\backslash d/\beta']^P \in \delta$,
$y \mapsto [\alpha'\backslash e_{(\varepsilon)a|b}\backslash d_{(\varepsilon)ab}/\beta']^P$ for each assignment $y \mapsto [\alpha'\backslash u\backslash d/\beta']^P \in \delta$,
$y \mapsto [\alpha'\backslash d_{(\varepsilon)ab}\backslash\beta']^P$ for each assignment $y \mapsto [\alpha'\backslash d\backslash\beta']^P \in \delta$,
$y \mapsto [\alpha'/d_{(\varepsilon)ab}/\beta']^P$ for each assignment $y \mapsto [\alpha'/d/\beta']^P \in \delta$,
$y \mapsto [\alpha'\backslash d_{(\varepsilon)ab}*\backslash\beta']^P$ for each assignment $y \mapsto [\alpha'\backslash d*\backslash\beta']^P \in \delta$,
$y \mapsto [\alpha'/d_{(\varepsilon)ab}*/\beta']^P$ for each assignment $y \mapsto [\alpha'/d*/\beta']^P \in \delta$.
2.2. Let $h(x) = \varepsilon$. Then $\mathbf{C}_1 = \mathbf{C} \cup \{d_x\}$, $\delta_{h^{-1}}(x) = d_x$ and each assignment $\delta : c \mapsto [l_m\backslash\ldots\backslash l_1\backslash v/r_1/\ldots/r_n]$, $c \in W$, is replaced by the assignment $\delta_{h^{-1}} : c \mapsto [d_x * \backslash l_m\backslash\ldots\backslash l_1\backslash d_x * \backslash v/d_x */r_1/\ldots/r_n/d_x*]$. □

3. $\mathcal{L}(gCDG)$ is closed under intersection with regular languages. Let $G = (W, \mathbf{C}, S, \delta)$ be a gCDG and A be a FA with states $Q = \{q_{in}, q_1, ..., q_k, q_{fin}\}$, where q_{in} is initial, $q_{fin} \neq q_{in}$ is final and in every transition $a\ q \to q'$, $a \in W, q \neq q_{fin}$ and $q' \neq q_{in}$. Then in the new gCDG $G_A = (W, \mathbf{C}_A, S, \delta_A)$ $\mathbf{C}_A = \mathbf{C} \cup Q$ and for every assignment $\delta : a \mapsto C^P$ and every transition $a\ q \to q'$, the new assignments δ_A are defined as $\delta_A : a \mapsto C^{PP_1P_2}$, where:

$$P_1 = \begin{cases} (\nearrow q_{in})(\nearrow q'), & if\ q = q_{in} \\ (\searrow q)(\searrow q_{fin}), & if\ q = q_{fin} \\ (\searrow q)(\nearrow q'), & otherwise \end{cases} \quad P_2 = \begin{cases} (\searrow q_{in})(\nearrow q_{fin}), & if\ C = [\alpha\backslash S/\beta] \\ \varepsilon, & otherwise \end{cases}$$ □

4-6. The proofs of closure under union, concatenation and Kleene + are standard and obvious. □

5 Categorial Dependency Grammars

Generalized CDG are useful for formal study of the grammars and languages. However, they are not flexible enough for designing real application grammars.

Their main drawback is that in order to fix the exact position of a distant subordinate, one needs to violate the tree-likeness of the DS (cf. the DS in Fig. 4). In order to eliminate this defect, we will use two more valency types: that of the *anchored* distant subordinate # and that of the host word $\flat$. When a distant right subordinate s through a dependency d should be positioned immediately on the left of its host word h, the latter must have in its type the argument $\flat^l(\diagdown d) : h \mapsto [\flat^l(\diagdown d)\backslash\beta]$ whereas the former must have the value type $\#^l(\diagdown d) : s \mapsto [\beta_1\backslash\#^l(\diagdown d)/\beta_2]$. After the argument valencies β_1 and β_2 of s will be saturated, the value type $\#^l(\diagdown d)$ becomes adjacent to the category $[\flat^l(\diagdown d)\backslash\beta]$, the host argument $\flat^l(\diagdown d)$ of this category is eliminated and the polarized valency $\diagdown d$ loses its anchor marker and falls under the **FA**-principle. These new types need a change in the dependency calculus. Below we present an extended calculus we call *sub-commutative*. The new DGs using this calculus will be called *Categorial Dependency Grammars* (CDG). We constrain the dependency types in order that the CDG generate only the DSs *with the single governor per word*.

Type constraints: In the categories $[L_1 \backslash \cdots \backslash L_i \backslash C / R_j / \cdots / R_1]$:
(i) the value type C can be neutral, or negative $(\swarrow C, \diagdown C)$ or anchored $(\#^l(\swarrow C), \#^r(\swarrow C), \#^l(\diagdown C), \#^r(\diagdown C))$,
(ii) the argument types L_i, (R_j) can be neutral or positive $(\nwarrow C, \nearrow C)$ or host $(\flat^l(\swarrow C), \flat^l(\diagdown C)$, respectively $\flat^r(\swarrow C), \flat^r(\diagdown C))$.

Definition 12. Sub-commutative dependency calculus.[9] *[8]*

$\mathbf{L^1}$. $C[C\backslash\beta] \vdash [\beta]$
$\mathbf{I^1}$. $C[C^*\backslash\beta] \vdash [C^*\backslash\beta]$
$\mathbf{\Omega^1}$. $[C^*\backslash\beta] \vdash [\beta]$
$\mathbf{V^1}$. $[\alpha\backslash\beta] \vdash \alpha[\beta]$, $\alpha \in \{(\nwarrow C), \flat^l(\swarrow C), \flat^l(\diagdown C)\}$
$\mathbf{A^1}$. $\#^l(\alpha)\flat^l(\alpha) \vdash \alpha$, $\alpha \in \{(\swarrow C), (\diagdown C)\}$
$\mathbf{C^1}$. $\alpha\beta \vdash \beta\alpha$, $\alpha \in \{(\swarrow C), (\nwarrow C), (\nearrow C), (\diagdown C)\}$, *where* $\beta = \flat(v)$ *or* β *has no occurrences of* $\alpha, \breve{\alpha}, \#(\alpha), \flat(\alpha)$
$\mathbf{D^1}$. $(\swarrow C)(\nwarrow C) \vdash \varepsilon$

In this calculus, the new rule $\mathbf{D^1}$ creates a discontinuous dependency for *adjacent* dual dependencies. At the same time, the rule $\mathbf{C^1}$ permutes the polarized valencies with other types when the permutation does not violate the **FA**-principle. The rule $\mathbf{V^1}$ decomposes complex types and the rule $\mathbf{A^1}$ eliminates the anchor markers if the corresponding anchor and host types are adjacent.

Remark 2. In contrast with the CDGs of [12], which generate only DTs, the DSs generated by the sub-commutative CDGs may have cycles, as shows the following example:
$[(\swarrow A)/(\nearrow B)]_1[(\nwarrow A)\backslash(\diagdown B)]_2 S_3 \vdash (\swarrow A)_1(\nearrow B)_1[(\nwarrow A)\backslash(\diagdown B)]_2 S_3 \vdash$
$(\swarrow A)_1(\nearrow B)_1(\nwarrow A)_2(\diagdown B)_2 S_3 \vdash (\swarrow A)_1(\nwarrow A)_2(\nearrow B)_1(\diagdown B)_2 S_3 \vdash S_3$.
A simple sufficient condition of acyclicity of CDGs can be formulated in terms of a well-founded order on dependency types. But more important is that these CDGs can naturally express adjacency of distant subordinates without violation of the single governor condition.

[9] We show only left rules. The right rules are symmetric.

Example 5.

$$G_2 = \begin{cases} a \mapsto \#^l(\swarrow A),\ [\flat^l(\swarrow A)\backslash\#^l(\swarrow A)] \\ b \mapsto [(\nwarrow A)\backslash B \,/\, C],\ [\flat^l(\swarrow A)\backslash(\nwarrow A)\backslash S/C] \\ c \mapsto C,\ [B\backslash C] \end{cases}$$

It is not difficult to prove that $L(G_2) = \{a^n b^n c^n \mid n > 0\}$. For instance, $a^3b^3c^3 \in L(G_2)$ due to the types assignment:
$a^3b^3c^3 \mapsto \#^l(\swarrow A)[\flat^l(\swarrow A)\backslash\#^l(\swarrow A)][\flat^l(\swarrow A)\backslash\#^l(\swarrow A)][\flat^l(\swarrow A)\backslash(\nwarrow A)\backslash S/C][(\nwarrow A)\backslash B/C][(\nwarrow A)\backslash B/C]C[B\backslash C][B\backslash C]$
and the proof shown in Fig. 11. This proof determines the DT shown in Fig. 12.

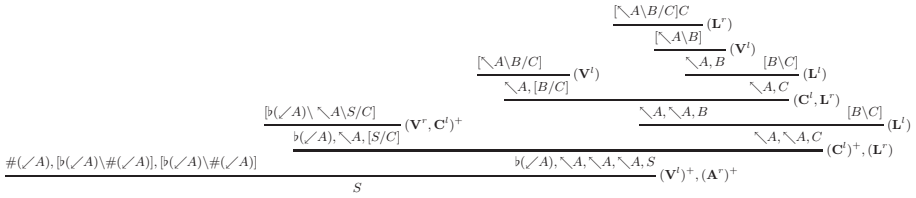


Fig. 11.

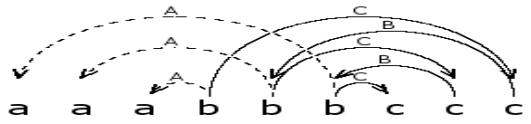


Fig. 12.

CDGs also enjoy the property of projections independence. The *local* projection of CDGs, preserves elementary and host argument subtypes. Intuitively, it reflects not only projective dependencies of words, but also their adjacency through anchor-host relations. The *valency projection* of CDGs, similar to that of gCDGs, preserves only polarized valency argument and value subtypes. Intuitively, it reflects only discontinuous dependencies.
Let $V^-(\mathbf{C})$ and $V^+(\mathbf{C})$ denote the sets of all negative polarized types: $\swarrow d$, $\searrow d$ (respectively, all positive polarized types: $\nwarrow d$, $\nearrow d$), where $d \in \mathbf{C}$. We set:

$Host^l(\mathbf{C}) =_{df} \{\flat^l(\alpha) \mid \alpha \in V^-(\mathbf{C})\}$, $\quad Anc^l(\mathbf{C}) =_{df} \{\#^l(\alpha) \mid \alpha \in V^-(\mathbf{C})\}$,
$Host^r(\mathbf{C}) =_{df} \{\flat^r(\alpha) \mid \alpha \in V^-(\mathbf{C})\}$, $\quad Anc^r(\mathbf{C}) =_{df} \{\#^r(\alpha) \mid \alpha \in V^-(\mathbf{C})\}$,
$Host(\mathbf{C}) =_{df} Host^l(\mathbf{C}) \cup Host^r(\mathbf{C})$, $\quad Anc(\mathbf{C}) =_{df} Anc^l(\mathbf{C}) \cup Anc^r(\mathbf{C})$.

Definition 13. Local projection $\|\gamma\|_l$ *of* $\gamma \in Cat(\mathbf{C})^*$ *is defined as follows:*
l1. $\|\varepsilon\|_l = \varepsilon$; $\|C\gamma\|_l = \|C\|_l\|\gamma\|_l$ *for* $C \in Cat(\mathbf{C})$ *and* $\gamma \in Cat(\mathbf{C})^*$.
l2. $\|C\|_l = C$ *for* $C \in \mathbf{C} \cup \mathbf{C}^* \cup Anc(\mathbf{C})$.
l3. $\|C\|_l = \varepsilon$ *for* $C \in V^+(\mathbf{C}) \cup V^-(\mathbf{C})$.
l4. $\|[\alpha]\|_l = \|\alpha\|_l$ *for all* $\alpha \in Cat(\mathbf{C})$.
l5. $\|[a\backslash\alpha]\|_l = [a\backslash\ \|\alpha\|_l]$ *and* $\|[\alpha/a]\|_l = [\|\alpha\|_l/a]$ *for* $a \in \mathbf{C} \cup \mathbf{C}^* \cup Host(\mathbf{C})$ *and* $\alpha \in Cat(\mathbf{C})$.
l6. $\|[(\nwarrow a)\backslash\alpha]\|_l = \|[\alpha/(\nearrow a)]\|_l = \|\alpha\|_l$ *for all* $a \in \mathbf{C}$ *and* $\alpha \in Cat(\mathbf{C})$.
Valency projection $\|\gamma\|_v$ *of* $\gamma \in Cat(\mathbf{C})^*$ *is defined as follows:*
v1. $\|\varepsilon\|_v = \varepsilon$; $\|C\gamma\|_v = \|C\|_v\|\gamma\|_v$ *for* $C \in Cat(\mathbf{C})$ *and* $\gamma \in Cat(\mathbf{C})^*$.
v2. $\|C\|_v = \varepsilon$ *for* $C \in \mathbf{C} \cup \mathbf{C}^*$.
v3. $\|C\|_v = C$ *for* $C \in V^+(\mathbf{C}) \cup V^-(\mathbf{C})$.
v4. $\|\#(C)\|_v = C$ *for* $C \in V^-(\mathbf{C})$.
v5. $\|[\alpha]\|_v = \|\alpha\|_v$ *for all* $[\alpha] \in Cat(\mathbf{C})$.
v6. $\|[a\backslash\alpha]\|_v = \|[\alpha/a]\|_v = \|\alpha\|_v$ *for* $a \in \mathbf{C} \cup \mathbf{C}^* \cup Host(\mathbf{C})$.
v7. $\|[a\backslash\alpha]\|_v = a\ \|\alpha\|_v$, *if* $a \in V^+(\mathbf{C})$.
v8. $\|[\alpha/a]\|_v = \|\alpha\|_v\ a$, *if* $a \in V^+(\mathbf{C})$.

Example 6. According to these definitions,
$\|[\mathfrak{b}^l(\searrow c)\backslash(\nwarrow a)\backslash b\backslash\#^r(\swarrow d)]\|_l\ =\ [\mathfrak{b}^l(\searrow c)\backslash b\backslash\#^r(\swarrow d)]$,
$\|[\mathfrak{b}^l(\searrow c)\backslash(\nwarrow a)\backslash b\backslash(\swarrow d)/e]\|_l\ =[\mathfrak{b}^l(\searrow c)\backslash b\backslash\varepsilon/e]$,
$\|[\mathfrak{b}^l(\searrow c)\backslash(\nwarrow a)\backslash b\backslash d]\|_v\ = \nwarrow a$, $\|[\mathfrak{b}^l(\searrow c)\backslash(\nwarrow a)\backslash b\backslash\#^l(\swarrow d)/e]\|_v\ = \nwarrow a \swarrow d$.

For technical reasons, it will be convenient to extend the common projective core **c** of the generalized and sub-commutative dependency calculus by the rule
$\mathbf{E^l}$. $\#^l(\alpha)[\mathfrak{b}^l(\alpha)\backslash\beta] \vdash \beta$ for $\alpha \in V^-(\mathbf{C})$.
The resulting extension will be denoted by **p** and the corresponding provability relation will be denoted by $\vdash_{\mathbf{p}}$.

Now we can state the projections independence criterion for the CDGs.

Theorem 5. *Let* $G = (W, \mathbf{C}, S, \delta)$ *be a CDG.* $x \in L(G)$ *iff there is a string of categories* $\gamma \in \delta(x)$ *such that:*
1. $\|\gamma\|_l \vdash_{\mathbf{p}} S$,
2. $\|\gamma\|_v$ *is balanced.*

Proof. Evidently, we can ignore the dependencies.
($\Rightarrow$) Let $x \in L(G)$ and $\delta : x \mapsto \gamma$ be an assignment for which there exists a proof $\gamma \vdash \ldots \vdash \gamma^k \vdash \gamma^n = S$ for some $n \geq 0$. We will prove by induction on k that for each $0 \leq k \leq n$ the following two assertions hold:
(i) $\|\gamma\|_l \vdash_{\mathbf{p}} \|\gamma^k\|_l$,
(ii) each correct pair $(\alpha, \breve{\alpha})$ is eliminated in $\|\gamma^k\|_v$ iff it is eliminated in $\|\gamma\|_v$.

Let us suppose that the conditions (i) and (ii) are satisfied for some $k < n$ and prove that they will be satisfied for $k + 1$ as well.

Let $\gamma^k \vdash^R \gamma^{k+1}$ (immediately derived by rule R).

If $R = \mathbf{L^l}$, then $\gamma^k = \Gamma_1 C[C\backslash\beta]\Gamma_2$ and $\gamma^{k+1} = \Gamma_1\beta\Gamma_2$ for some Γ_1, Γ_2 and β. Passing to their local projections, we obtain: $\|\gamma^k\|_l = \|\Gamma_1\|_l C[C\backslash\|\beta\|_l]\|\Gamma_2\|_l \vdash_{\mathbf{p}} \|\Gamma_1\|_l\|\beta\|_l\|\Gamma_2\|_l = \|\gamma^{k+1}\|_l$ and $\|\gamma^k\|_v = \|\gamma^{k+1}\|_v$. Then $\|\gamma\|_l \vdash_{\mathbf{p}} \|\gamma^k\|_l \vdash^{\mathbf{L^l}} \|\gamma^{k+1}\|_l$ and both conditions (i) and (ii) are satisfied for $k + 1$.

If $R = \mathbf{A^l}$, then $\gamma^k = \Gamma_1 \#^l(\alpha)[\flat^l(\alpha)\backslash\beta]\Gamma_2$ and $\gamma^{k+1} = \Gamma_1(\alpha)\beta\Gamma_2$ for some Γ_1, Γ_2 and $\#^l(\alpha) \in Anc(\mathbf{C}), \flat^l(\alpha) \in Host(\mathbf{C})$. Then by definition of projections, we get: $\|\gamma^k\|_l = \|\Gamma_1\|_l \#^l(\alpha)[\flat^l(\alpha)\backslash\|\beta\|_l]\|\Gamma_2\|_l \vdash^{\mathbf{E^l}}_{\mathbf{p}} \|\Gamma_1\|_l\|\beta\|_l\|\Gamma_2\|_l = \|\gamma^{k+1}\|_l$ and $\|\gamma^k\|_v = \|\Gamma_1\|_v\ (\alpha)\|\beta\|_v\|\Gamma_2\|_v = \|\gamma^{k+1}\|_v$. So (i) and (ii) are satisfied for $k+1$.

If $R = \mathbf{C^l}$, then $\gamma^k = \Gamma_1\ C\alpha\ \Gamma_2$ and $\gamma^{k+1} = \Gamma_1\ \alpha C\ \Gamma_2$ for some $\alpha \in (\nwarrow \mathbf{C} \cup \searrow \mathbf{C})$ and $C \in Cat(\mathbf{C})$. Clearly, in this case $\|\gamma^k\|_l = \|\gamma^{k+1}\|_l$. Now, since C has no occurrences of $\alpha, \#(\alpha)$ or $\breve{\alpha}$, then the correct pair $(\alpha, \breve{\alpha})$ is eliminated in $\|\gamma^{k+1}\|_v$ by rule $\mathbf{D^l}$ iff it is eliminated in $\|\gamma^k\|_v$ by this rule. The projections of $\|\gamma^{k+1}\|_v$ and $\|\gamma^i\|_v$ on any other pair $(\beta, \breve{\beta})$ of polarized valencies are not affected by this step, so they do not change. Therefore, both conditions (i) and (ii) are satisfied for $k+1$.

If $R = \mathbf{D^l}$, then $\gamma^k = \Gamma_1(\swarrow C)(\nwarrow C)\Gamma_2$ and $\gamma^{k+1} = \Gamma_1\Gamma_2$. Clearly, $\|\gamma^k\|_l = \|\gamma^{k+1}\|_l$. As to the valency projection $\|\gamma^{k+1}\|_v = \|\Gamma_1\|_v\|\Gamma_2\|_v$, it is obtained from the projection $\|\gamma^k\|_v = \|\Gamma_1\|_v(\swarrow C)(\nwarrow C)\|\Gamma_2\|_v$ by eliminating the correct valency pair $(\swarrow C)(\nwarrow C)$. Therefore, this pair is eliminated in $\|\gamma^{k+1}\|_v$ iff it is eliminated in $\|\gamma^k\|_v$. The projections of $\|\gamma^{k+1}\|_v$ and $\|\gamma^k\|_v$ on any other pair $(\beta, \breve{\beta})$ of polarized valencies rest intact. Therefore, both conditions (i) and (ii) are satisfied for $k+1$.

The proof steps via "right" rules $\mathbf{L^r}$, $\mathbf{A^r}$, $\mathbf{P^r}$, $\mathbf{C^r}$, and $\mathbf{D^r}$ are proved similarly. Iterative dependency rule $\mathbf{I^l}$ is treated as the rule $\mathbf{L^l}$. The case of the rule $\mathbf{\Omega^l}$ is trivial and the rules $\mathbf{V^l}, \mathbf{V^r}$ just do not affect the projections.

So we prove that (i) and (ii) are satisfied for each $k = 0, \ldots, n$. Since $\|\gamma^n\|_l = S$ and $\|\gamma^n\|_v = \varepsilon$, the assertions 1 and 2 of the theorem are true.

($\Leftarrow$) Now let us suppose that

1. $\|\gamma\|_l \vdash_{\mathbf{p}} S$ and
2. each correct pair $(\alpha, \breve{\alpha})$ is eliminated in $\|\gamma\|_v$ for an assignment $\delta : x \mapsto \gamma$.

We will show that $\gamma \vdash S$, which implies that $x \in L(G)$. To do this, we will suppose that $\|\gamma\|_l \vdash^n_{\mathbf{p}} S$ for some $n \geq 0$ and show the existence of a proof $\gamma \vdash S$ by induction on n.

If $n = 0$, then $\gamma = \Gamma_1 S \Gamma_2$ for some balanced potentials Γ_1 and Γ_2. Then, the needed proof has the form $\gamma = \Gamma_1 S \Gamma_2 \vdash \Gamma_1\Gamma_2 S \vdash S$. In the first part of this proof only the commutativity rules are used. The second part exists by Lemma 1.3.

Suppose that the assertion is valid for $n \leq k$. Let us prove it for $n = k+1$. Let $\|\gamma\|_l \vdash^{k+1}_{\mathbf{p}} S = \|\gamma\|_l \vdash^R_{\mathbf{p}} \gamma'_l \vdash^k_{\mathbf{p}} S$, where R is the first applied rule.

If $R = \mathbf{L^l}$, then $\|\gamma\|_l = \Gamma_1 C[C\backslash\beta]\Gamma_2$ and $\gamma'_l = \Gamma_1\beta\Gamma_2$ for some $\Gamma_1, \Gamma_2, \beta$. Clearly, $\gamma = \tilde{\Gamma}_1 C \Delta[C\backslash\tilde{\beta}]\tilde{\Gamma}_2$, where $\|\tilde{\Gamma}_1\|_l = \Gamma_1, \|\tilde{\beta}\|_l = \beta, \|\tilde{\Gamma}_2\|_l = \Gamma_2$ and Δ is a potential. Now we can construct the proof $\gamma = \tilde{\Gamma}_1 C\Delta[C\backslash\tilde{\beta}]\tilde{\Gamma}_2 \vdash \tilde{\Gamma}_1\Delta C[C\backslash\tilde{\beta}]\tilde{\Gamma}_2$ $\vdash^{\mathbf{L^1}} \tilde{\Gamma}_1\Delta\tilde{\beta}\tilde{\Gamma}_2 = \gamma'$, where in the first part of the proof only commutativity rules are used in order to permute C and Δ. Regarding the projections of γ', we can see that $\|\gamma'\|_l = \gamma'_l$ and $\|\gamma'\|_v = \|\gamma\|_v$. Therefore, $\|\gamma'\|_v$ is balanced and the assertion follows by induction.

If $R = \mathbf{E^l}$, then $\|\gamma\|_l = \Gamma_1 \#^l(\alpha)[\flat^l(\alpha)\backslash\beta]\Gamma_2$ and $\gamma'_l = \Gamma_1\beta\Gamma_2$ for some Γ_1, Γ_2, β and $\#^l(\alpha) \in Anc(\mathbf{C})$, $\flat^l(\alpha) \in Host(\mathbf{C})$. By definition of projections, $\gamma = \tilde{\Gamma}_1 \#^l(\alpha)\Delta[\flat^l(\alpha)\backslash\tilde{\beta}]\tilde{\Gamma}_2$, $\|\gamma\|_v = \|\tilde{\Gamma}_1\|_v\alpha\|\Delta\|_v\|\tilde{\beta}\|_v\|\tilde{\Gamma}_2\|_v$ is balanced, $\|\tilde{\Gamma}_1\|_l = \Gamma_1, \|\tilde{\Gamma}_2\|_l = \Gamma_2, \|\beta\|_l = \beta$, and $\Delta \in (V^+(\mathbf{C}) \cup V^-(\mathbf{C}))^*$.

Let us consider the proof:

(1) $\gamma = \tilde{\Gamma}_1 \#^l(\alpha) \Delta [\flat^l(\alpha) \backslash \tilde{\beta}] \tilde{\Gamma}_2 \vdash^{\mathbf{V^1}} \tilde{\Gamma}_1 \#^l(\alpha) \Delta \flat^l(\alpha) \tilde{\beta} \tilde{\Gamma}_2 \vdash \tilde{\Gamma}_1 \#^l(\alpha) \flat^l(\alpha) \Delta \tilde{\beta} \tilde{\Gamma}_2$
$\vdash^{\mathbf{A^1}} \tilde{\Gamma}_1 \alpha \Delta \tilde{\beta} \tilde{\Gamma}_2$,

in which in the part $\vdash$ only the commutativity rules are used in order to permute $\flat^l(\alpha)$ and Δ. This means that $\|\tilde{\Gamma}_1 \alpha \Delta \tilde{\beta} \tilde{\Gamma}_2\|_l = \Gamma_1 \beta \Gamma_2 = \gamma'_l$ and $\|\tilde{\Gamma}_1 \alpha \Delta \tilde{\beta} \tilde{\Gamma}_2\|_v = \|\tilde{\Gamma}_1\|_v \alpha \|\Delta\|_v \|\tilde{\beta}\|_v \|\tilde{\Gamma}_2\|_v = \|\gamma\|_v$ is balanced. Therefore, by induction the proof (1) can be completed with a proof $\tilde{\Gamma}_1 \alpha \Delta \tilde{\beta} \tilde{\Gamma}_2 \vdash S$. □

Corollary 1. $\mathcal{L}(CDG) \subseteq \mathcal{L}(gCDG)$.

Proof. This corollary follows from Theorems 1,5 using the type simulation, in which the types $\#^l(\alpha)$, $\flat^l(\alpha)$ are replaced by the new *primitive* type $<\#^l(\alpha)>$, in the place of each assignment $\delta : x \mapsto [\beta_1 \backslash \#^l(\alpha) / \beta_2]^P$ there is the assignment $\delta : x \mapsto [\beta_1 \backslash < \#^l(\alpha) > / \beta_2]^{P\alpha}$ and in the place of $\delta : x \mapsto [\flat^l(\alpha) \backslash \beta]^P$ there is the assignment $\delta : x \mapsto [< \#^l(\alpha) > \backslash \beta]^{P\alpha}$ (similar for other orientations). □

6 Parsing Complexity

The general parsing problem $pars(G, s, w) \equiv$ "$w \in L(G)$ and s is a syntactic structure assigned to w by G" is not necessarily polynomial time when all problems $pars_{G_0}(s, w) \equiv pars(G_0, s, w)$ for particular G_0 are so (as this is the case of CF-grammars). If there is no uniform bound on the number of polarized valencies, then the general parsing problem for gCDGs is hard.

Theorem 6. *The general parsing problem $G(D, w)$ for gCDGs is NP-complete.*

Proof. The NP-hardness can be proved by the following polynomial reduction of $3-CNF$. Let $\Phi = C_1 \wedge \ldots \wedge C_m$ be a CNF with clauses C_j including three literals l_1^j, l_2^j, l_3^j and $l_k^j \in \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$. We define from Φ the CDG $G(\Phi) = (W, \mathbf{C}, S, \delta)$, in which $W = \{\Phi, C_1, \ldots, C_m, x_1, \ldots, x_n, y_1, \ldots y_n\}$, $\mathbf{C} = \{S, A, 1_0, 1_1, 2_0, 2_1, \ldots, n_0, n_1\}$ and $\delta(\Phi) = [(A\backslash)^n \backslash S]$, $\delta(x_i) = \{[A/(\nearrow i_0)], [A/(\nearrow i_1)]\}$, $\delta(y_i) = \{(\searrow i_0), (\searrow i_1)\}$, $\delta(C_j) = \{cat(l_1^j), cat(l_2^j), cat(l_3^j)\}$, where $cat(x_i) = [(\searrow i_1)/(\nearrow i_1)]$, $cat(\neg x_i) = [(\searrow i_0)/(\nearrow i_0)]$. Let also $w(\Phi) = x_1 x_2 \ldots x_n \Phi C_1 C_2 \ldots C_m y_1 y_2 \ldots y_n$.

Assertion. Φ is satisfiable iff $(\exists D : DT)\ G(\Phi)(D, w(\Phi))$.

This assertion follows from the fact that $G(\Phi)(D, w(\Phi))$ does not hold iff at least for one $i, 1 \leq i \leq n$, the category $[A/(\nearrow i_0)]$ is chosen in some $\delta(C_j)$ and $[A/(\nearrow i_1)]$ is chosen in some other $\delta(C_k)$. On the other hand, this conflict cannot be avoided iff Φ is not satisfiable. □

In practice, the inventory of polarized valencies is finite and fixed for DGs of particular languages. Due to the projections independence property, each particular gCDG parsing problem turns out to be polynomial time. In [8] we have described a polynomial time parsing algorithm for CDGs. It was implemented in Lisp by Darin and Hristian Todorov.[10] Below we will present a parsing algorithm for gCDG. It was implemented in $C\#$ and optimized by Ilya Zaytsev.

[10] The analyses shown in the figures are carried out by this algorithm.

Preliminaries. Let us fix for the rest a gCDG $G = (W, \mathbf{C}, S, \delta)$. We will first define two failure functions used for the algorithm optimization.

Let $w = w_1 w_2 ... w_n \in W^+$, $\alpha \in V^l(\mathbf{C})$ and $1 \leq i \leq n$. Then

$$\pi^L(\alpha, i) = max\{\Delta_\alpha(\|\Gamma\|_v) \mid \Gamma \in \delta(w_1 ... w_i)\}$$

is the *left failure function* and for $\alpha \in V^r(\mathbf{C})$,

$$\pi^R(\alpha, i) = max\{\Delta_\alpha(\|\Gamma\|_v) \mid \Gamma \in \delta(w_{n-i+1} ... w_n)\}$$

is the *right failure function*. We set $\pi^L(\alpha, 0) = \pi^R(\alpha, 0) = 0$. It is not difficult to prove the following properties of these functions.

Lemma 4. *(i) Let* $1 \leq i \leq n-1$. *Then*
$\pi^L(\alpha, i+1) =$
$max\{\Delta_\alpha(P) + max\{\pi^L(\alpha, i) - \Delta_{\breve{\alpha}}(P), 0\} \mid P = \|\gamma\|_v, \gamma \in \delta(w_{i+1})\}$,
$\pi^R(\alpha, i+1) =$
$max\{\Delta_{\breve{\alpha}}(P) + max\{\pi^R(\alpha, i) - \Delta_\alpha(P), 0\} \mid P = \|\gamma\|_v, \gamma \in \delta(w_{n-i+1})\}$.
(ii) If $\Gamma \vdash S$ *for some* $\Gamma = \gamma_1 ... \gamma_n \in \delta(w)$, *then*
$\Delta_\alpha(\|\gamma_i ... \gamma_j\|_v) \leq \pi^R(\breve{\alpha}, n-j)$, $\Delta_{\breve{\alpha}}(\|\gamma_i ... \gamma_j\|_v) \leq \pi^L(\alpha, i-1)$
for all $1 \leq i \leq j \leq n$, *all* $\alpha \in V^l(\mathbf{C})$ *and* $\breve{\alpha} \in V^r(\mathbf{C})$.

Algorithm description. **gCdgAnalyst** is a standard dynamic programming parsing algorithm. It applies to a gCDG $G = (W, \mathbf{C}, S, \delta)$ with left polarized valencies $V^l(\mathbf{C}) = \{v_1, \ldots, v_p\}$ and dual right valencies $V^r(\mathbf{C}) = \{\breve{v}_1, \ldots, \breve{v}_p\}$ and to a string $w = w_1 w_2 ... w_n \in W^+$ and fills up a $n \times n$ triangle matrix M with *items*. Each cell $M[i, j]$, $i \leq j$, corresponds to the string interval $w_i ... w_j$ and contains a finite set of items. Each item codes a generalized type C^P and has the form $\langle C, \Delta^L, \Delta^R, I^l, I^r \rangle$, where:

- C is a neutral category $C \in nCat(\mathbf{C})$,
- $\Delta^L = (\Delta_{v_1}, \ldots, \Delta_{v_p})$ and $\Delta^R = (\Delta_{\breve{v}_1}, \ldots, \Delta_{\breve{v}_p})$ are integer vectors whose component i contains the corresponding deficits of right (left) non-paired v-brackets in the potential P (see Definition 7),
- I^l, I^r are left and right angle items from which I is calculated (for I in diagonal $M[i, i]$, $I^l = I^r = \emptyset$).

Algorithm gCdgAnalyst
//*Input*: gCDG G, string $w = w_1 ... w_n$
//*Output*: $\langle$"*yes*", $DS\mathsf{D}\rangle$ iff $w \in L(G)$
{
  CalcFailFuncL();
  CalcFailFuncR();
  for $(k = 1, \ldots, n)$
  {
    Propose(k)
  }

```
  for (l = 2, ..., n)
 {
      for (i = 1, ..., n − l)
      {
          j := i + l − 1;
          for (k = i, ..., j − l)
          {
              SubordinateL(i, k, j);
              SubordinateR(i, k, j);
          }
      }
 }
 if (I = ⟨S, (0, 0, ..., 0), (0, 0, ..., 0), I^l, I^r⟩ ∈ M[1, n])
      return ⟨"yes", Expand(I)⟩;
 //procedure Expand( I ) calculates the output DS
 else
      return ⟨"no", ∅⟩;
}

CalcFailFuncL()
{
  foreach (v ∈ V^l(C))
  {
      π^L[v, 0] := 0;
      for (i = 1, ..., n)
      {
          π_max := 0;
          foreach (C^P ∈ δ(w_i))
          {
              π_max := max{π_max, Δ_v(P) + max{π^L[v, i − 1] − Δ_v̌(P), 0}};
          }
          π^L[v, i] := π_max;
      }
  }
}

CalcFailFuncR()
{
  foreach (v̌ ∈ V^r(C))
  {
      π^R[v, 0] := 0;
      for (i = 1, ..., n)
      {
          π_max := 0;
          foreach (C^P ∈ δ(w_{n−i+1}))
          {
```

```
                $\pi_{max} := max\{\pi_{max}, \Delta_{\check{v}}(P) + max\{\pi^R[v, i-1] - \Delta_v(P), 0\}\}$;
            }
            $\pi^R[v, i] := \pi_{max}$;
        }
    }
}

AddItem( $M[i,j], \langle C, \Delta^L, \Delta^R, I^l, I^r\rangle$ )
{
    $M[i,j] := M[i,j] \cup \{\langle C, \Delta^L, \Delta^R, I^l, I^r\rangle\}$;
    if ($C = [C' * \backslash\beta]$)
    {
        AddItem( $M[i,j], \langle [\beta], \Delta^L, \Delta^R, I^l, I^r\rangle$ );
    }
    if ($C = [\beta / C' *]$)
    {
        AddItem( $M[i,j], \langle [\beta], \Delta^L, \Delta^R, I^l, I^r\rangle$ );
    }
}

//For $1 \leq i \leq n$
Propose( $i$ )
{
    (loop) foreach ($C^P \in \delta(w_i)$)
    {
        foreach ($v \in V^l(\mathbf{C})$)
        {
            $\Delta^L[v] := \Delta_v(P)$;
            if ($\Delta^L[v] > \pi^R[\check{v}, n-j]$) next (loop);
            $\Delta^R[\check{v}] := \Delta_{\check{v}}(P)$;
            if ($\Delta^R[\check{v}] > \pi^L[v, i-1]$) next (loop);
        }
        AddItem( $M[i,i], \langle C, \Delta^L, \Delta^R, \emptyset, \emptyset\rangle$ );
    }
}

//For $1 \leq i \leq k \leq j \leq n$
SubordinateL( $i, k, j$ )
{
    (loop) foreach ($I_1 = \langle \alpha_1, \Delta_1^L, \Delta_1^R, I_1^l, I_1^r\rangle \in M[i,k]$,
                    $I_2 = \langle \alpha_2, \Delta_2^L, \Delta_2^R, I_2^l, I_2^r\rangle \in M[k+1,j]$)
    {
        foreach ($v \in V^l(\mathbf{C})$)
        {
            $\Delta^L[v] := \Delta_2^L(v) + max\{\Delta_1^L(v) - \Delta_2^R(v), 0\}$;
            if ($\Delta^L[v] > \pi^R[\check{v}, n-j]$) next (loop);
```

$\Delta^R[\breve{v}] := \Delta_1^R(\breve{v}) + max\{\Delta_2^R(\breve{v}) - \Delta_1^L(\breve{v}), 0\}$;
if ($\Delta^R[\breve{v}] > \pi^L[v, i-1]$) next (loop);
}
if ($\alpha_1 = C$ and $\alpha_2 = [C\backslash\beta]$)
{
AddItem($M[i,j], \langle [\beta], \Delta^L, \Delta^R, I_1, I_2\rangle$);
}
elseif (($\alpha_1 = C$ and $\alpha_2 = [C * \backslash\beta]$) or $\alpha_1 = [\varepsilon]$)
{
AddItem($M[i,j], \langle \alpha_2, \Delta^L, \Delta^R, I_1, I_2\rangle$);
}
}
}
SubordinateR(i, k, j) is similar.

Correctness. Correctness of CalcFailFuncL() and CalcFailFuncR() follows from Lemma 4.

Example 7. Let $W = \{a, b\}, \mathbf{C} = \{A, B\}$, $\delta(a) = \{[A/A]^{\swarrow B \swarrow B}, [B * \backslash A]^{\searrow A \searrow A}\}$, $\delta(b) = \{[B]^{\nearrow A \nwarrow B}, [\varepsilon]^{\nearrow A \swarrow B}\}$. Then $V^l(\mathbf{C}) = \{\nearrow A, \swarrow B\}$, $V^r(\mathbf{C}) = \{\searrow A, \nwarrow B\}$. For the string $w = abba$, the category potentials are presented in the table:

a	b	b	a
$\swarrow B \swarrow B$	$\nearrow A \nwarrow B$	$\nearrow A \nwarrow B$	$\swarrow B \swarrow B$
$\searrow A \searrow A$	$\nearrow A \swarrow B$	$\nearrow A \swarrow B$	$\searrow A \searrow A$

CalcFailFuncL() and CalcFailFuncR() will calculate the following values:

i	0	1	2	3	4
$\pi^L[\nearrow A, i]$	0	0	1	2	2
$\pi^L[\swarrow B, i]$	0	2	3	4	6

i	4	3	2	1	0
$\pi^R[\searrow A, i]$	2	0	1	2	0
$\pi^R[\nwarrow B, i]$	2	2	1	0	0

Theorem 7. *Let $G = (W, \mathbf{C}, S, \delta)$ be a gCDG and $w = w_1 w_2 ... w_n \in W^+$. Then for any $1 \le i \le k \le j \le n$, an item $I = \langle \theta, \Delta^L, \Delta^R, I^l, I^r\rangle$ falls to $M[i,j]$ iff there is $\Gamma = \Gamma_1 \gamma_i \dots \gamma_j \Gamma_2 \in \delta(w)$ such that $\gamma_i \dots \gamma_j \in \delta(w_i \dots w_j)$ and*
(i) $\gamma_i \dots \gamma_j \vdash \theta$,
(ii) $\Delta^L[\alpha] = \Delta_\alpha(\|\gamma_i \dots \gamma_j\|_v)$, $\Delta^R[\alpha] = \Delta_{\breve{\alpha}}(\|\gamma_i \dots \gamma_j\|_v)$ *for all* $\alpha \in V^l(\mathbf{C})$, $\breve{\alpha} \in V^r(\mathbf{C})$,
(iii) $\gamma_i \dots \gamma_j$ *satisfies the condition (ii) of Lemma 4.*

Proof. Let $l = j - i + 1$.
($\Rightarrow$) Let $I \in M[i,j]$. We will show that there is Γ satisfying the conditions of the theorem by induction on l.
1. If $l = 1$, then I is put to $M[i,i]$ by Propose(i). In this case, the conditions $(i) - (iii)$ are trivially satisfied.
2. Let us suppose that the theorem is true for all $l' < l$. Then $i < j$. In

this case, $I \in M[i,j]$ implies that there is $i \leq k < j$ such that I was put in $M[i,j]$ by $\mathsf{SubordinateL}(\ i,k,j\ )$ or by $\mathsf{SubordinateR}(\ i,k,j\ )$. Let it be by $\mathsf{SubordinateL}(\ i,k,j\ )$. Then there must be $I_1 = \langle \theta_1, \Delta_1^L, \Delta_1^R, I_1^l, I_1^r \rangle \in M[i,k]$ $I_2 = \langle \theta_2, \Delta_2^L, \Delta_2^R, I_2^l, I_2^r \rangle \in M[k+1,j]$ satisfying $(i)-(iii)$. Therefore, $\gamma_i \ldots \gamma_k \vdash \theta_1$, $\gamma_{k+1} \ldots \gamma_j \vdash \theta_2$ and by definition of $\mathsf{SubordinateL}(\ i,k,j\ )$, $\theta_1 = C, \theta_2 = [C\backslash\beta], \theta = \beta$ or $\theta_1 = C, \theta_2 = [C * \backslash\beta], \theta = \theta_2$, or $\theta_1 = \varepsilon, \theta = \theta_2$. In all of these cases, $\gamma_i \ldots \gamma_k \gamma_{k+1} \ldots \gamma_j \vdash \theta$. Given that $\Gamma_1' \gamma_i \ldots \gamma_k \Gamma_2' \in \delta(w)$, $\Gamma_1'' \gamma_{k+1} \ldots \gamma_j \Gamma_2'' \in \delta(w)$, $\gamma_i \ldots \gamma_k \in \delta(w_i \ldots w_k)$, $\gamma_{k+1} \ldots \gamma_j \in \delta(w_{k+1} \ldots w_j)$, we see that $\Gamma_1' \gamma_i \ldots \gamma_j \Gamma_2'' \in \delta(w)$, $\gamma_i \ldots \gamma_j \in \delta(w_i \ldots w_j)$ and $\gamma_i \ldots \gamma_j \vdash \theta$. Point (ii) directly follows from Lemma $4(i)$. Finally, $I \in M[i,j]$ means that I does not violate the necessary condition in Lemma $4(ii)$.
($\Leftarrow$) By induction on l immediately following the definition of **gCdgAnalyst**.
□

Complexity. For a gCDG $G = (W, \mathbf{C}, S, \delta)$, let $l_G = |\delta|$ be the number of category assignments in the lexicon, $a_G = max\{k \mid \exists x \in W([\alpha_k\backslash ... \backslash\alpha_1\backslash C/\beta]^P \in \delta(x) \vee [\beta\backslash C/\alpha_1/.../\alpha_k]^P \in \delta(x))\}$ be the maximal number of argument subtypes in assigned categories, $p_G = |V^l(\mathbf{C})| = |V^r(\mathbf{C})|$ be the number of polarized valencies and $\Delta_G = max\{\Delta_\alpha(P) \mid \exists x \in W(C^P \in \delta(x) \ \vee \ \alpha \in V(\mathbf{C}))\}$ be the maximal valency deficit in assigned categories. In the complexity bound below n will denote the length of the input string $n = |w|$.

Theorem 8. *Algorithm* **gCdgAnalyst** *has time complexity*
$$\mathbf{O}(\mathsf{l_G} \cdot \mathsf{a_G^2} \cdot (\Delta_\mathsf{G} \cdot \mathsf{n})^{2\mathsf{p_G}} \cdot \mathsf{n}^3).$$

Proof. A category $\gamma \in \delta(x)$ may be cancelled to no more than a_G^2 different categories. So the maximal number of matrix cell elements is $l_G \cdot a_G^2$. The valency deficits are bounded by the maximal value of the failure functions. So the maximal deficit of a polarized valency is $\Delta_G \cdot n$. Therefore, the number of different valency deficit vectors is bounded by $(\Delta_G \cdot n)^{2p_G}$. Filling one matrix cell needs visiting n cells. There are $\frac{n^2}{2}$ cells in M. This proves the time bound. □

Remark 3. 1. When G has no polarized valencies, the parsing time is evidently $\mathbf{O}(\mathsf{n}^3)$. Due to Theorem 3, every gCDG G with bounded valency deficit $\sigma < c$ can be translated into an equivalent gCDG G_c without polarized valencies (so with parsing time $\mathbf{O}(\mathsf{n}^3)$). Of course, the size of G_c is exponential: $|G_c| = \mathbf{O}(|\mathsf{G}| \cdot \mathsf{c}^{\mathsf{p_G}})$.
2. In practice, the failure functions significantly lower the time complexity.

7 Concluding Remarks

The main advantage of gCDG as compared to other formal models of surface syntax is that they allow to define the dependencies of all kinds, local and long, projective and discontinuous, in the same elegant and completely local manner. On the one hand, they are genuine categorial grammars and, as such, they are completely lexicalized and use types in the place of rules. On the other hand, they keep the traditional valency / polarity style peculiar to all dependency

grammars. The CDG which, in fact, constitute a subclass of gCDG, can be used in real applications. As we have shown, gCDG have a practical polynomial time parsing algorithm and enjoy good mathematical properties. They are learnable from positive data (see [4]) and equivalent to rule based DSG [3]. At the same time, a more detailed study of their expressiveness is needed, in particular, a comparison with the mild CS grammars [19] and the pregroup grammars [22].[11].

Very important is the question, whether the **FA**-principle is universal. There are evidences that it is adequate for many languages with, so to say, rigid WO, e.g. English, French, Spanish, Italian, German, Japan and many others. It seems adequate even for the languages with elaborated morphology and flexible WO, such as Russian, Turkish and some others. However, this principle does not apply to the constructions with serial infinitive phrase subjects (so called *cross-serial dependencies* [5]) in Dutch. The gCDGs with the **FA**-principle can be seen as uni-modal DGs. To cover these complex constructions, one should use other polynomially implementable modes and pass to *multimodal* gCDGS.

References

1. Bar-Hillel, Y.: A quasi-arithmetical notation for syntactic description. Language **29**(1) (1953) 47–58
2. Bar-Hillel, Y., Gaifman, H., Shamir, E.: On categorial and phrase structure grammars. Bull. Res. Council Israel **9F** (1960) 1–16
3. Béchet, D., Dikovsky, A., Foret, A.: Dependency structure grammars. In Blache, P., Stabler, E., eds.: Proc. of the 5th Int. Conf. "Logical Aspects of Computational Linguistics" (LACL'2005). LNAI 3492 (2005) 18–34
4. Béchet, D., Dikovsky, A., Foret, A., Moreau, E.: On learning discontinuous dependencies from positive data. In: Proc. of the 9th Intern. Conf. "Formal Grammar 2004" (FG 2004). (2004) 1–16 http://cs.haifa.ac.il/~shuly/fg04/.
5. Bresnan, J., Kaplan, R., Peters, S., Zaenen, A.: Cross-serial dependencies in dutch. Linguistic Inquiry **13**(4) (1982) 613–635
6. Bröker, N.: Separating surface order and syntactic relations in a dependency grammar. In: Proc. COLING-ACL, Montreal (1998) 174–180
7. Buszkowski, W.: Lambek grammars based on pregroups. In de Groote, P., Morill, G., Retoré, C., eds.: Logical Aspects of Computational Linguistics. Lecture Notes in Artificial Intelligence. vol. 2099, Springer (2001)
8. Dekhtyar, M., Dikovsky, A.: Categorial dependency grammars. In Moortgat, M., Prince, V., eds.: Proc. of Intern. Conf. on Categorial Grammars, Montpellier (2004) 76–91
9. Dikovsky, A.: Grammars for local and long dependencies. In: Proc. of the Intern. Conf. ACL'2001, Toulouse, France, ACL & Morgan Kaufman (2001) 156–163
10. Dikovsky, A.: Polarized non-projective dependency grammars. In de Groote, P., Morill, G., Retoré, C., eds.: Proc. of the Fourth Intern. Conf. on Logical Aspects of Computational Linguistics. Lecture Notes in Artificial Intelligence. vol. 2099, Springer (2001) 139–157

[11] The pregroup grammars are weakly equivalent to CF-grammars [7] At the same time, the types they assign to words are often close to the projective dependency types of the CDG.

11. Dikovsky, A.: Linguistic meaning from the language acquisition perspective. In Jäger, G., Monachesi, P., Penn, G., Wintner, S., eds.: Proc. of the 8th Intern. Conf. "Formal Grammar 2003" FG 2003, Vienna, Austria, Vienna Techn. Univ. (2003)
12. Dikovsky, A.: Dependencies as categories. In Duchier, D., Kruijff, G.J.M., eds.: "Recent Advances in Dependency Grammars". COLING'04 Workshop. (2004) 90–97
13. Dikovsky, A., Modina, L.: Dependencies on the other side of the curtain. Traitement Automatique des Langues (TAL) **41**(1) (2000) 79–111
14. Gaifman, H.: Dependency systems and phrase structure systems. Report p-2315, RAND Corp. Santa Monica (CA) (1961) Published in: Information and Control, 1965, v. 8, n 3, pp. 304-337.
15. Gladkij, A.V.: Lekcii po Matematičeskoj Lingvistike dlja Studentov NGU [Course of Mathematical Linguistics. Novosibirsk State University (Russ.)]. (French transl. Leçons de linguistique mathématique. facs. 1, 1970, Dunod). Novosibirsk State University (1966)
16. Hays, D.: Grouping and dependency theories. Research memorandum RM-2646, The RAND Corporation (1960) Published in Proc. of the National Symp. on Machine Translaion, Englewood Cliffs (N.Y.), 1961, pp. 258-266.
17. Hudson, R.A.: Word Grammar. Basil Blackwell, Oxford-New York (1984)
18. Jackendoff, R.: X' Syntax : A Study of Phrase Structure. MIT Press, Cambridge, MA (1977)
19. Joshi, A.K., Shanker, V.K., Weir, D.J.: The convergence of mildly context-sensitive grammar formalisms. In Sells, P., Shieber, S., Wasow, T., eds.: Foundational issues in natural language processing, Cambridge, MA, MIT Press (1991) 31–81
20. Kahane, S., ed.: Les grammaires de dépendance. In Kahane, S., ed.: Traitement automatique des langues. Volume 41., Paris, Hermes (2000) n. 1/2000.
21. Kruijff, J.J.M., Duchier, D., eds.: Recent Advances in Dependency Grammars. In Kruijff, J.J.M., Duchier, D., eds.: Proceedings of COLING Workshop, Geneva (August 2004)
22. Lambek, J.: Type grammar revisited. In et al., A.L., ed.: Logical Aspects of Computational Linguistics. Lecture Notes in Artificial Intelligence. vol. 1582, Springer (1999) 1–27
23. Lombardo, V., Lesmo, L.: An earley-type recognizer for dependency grammar. In: Proc. 16th COLING. (1996) 723–728
24. Mel'čuk, I.: Dependency Syntax. SUNY Press, Albany, NY (1988)
25. Morrill, G.V.: Type Logical Grammar. Categorial Logic of Signs. Kluwer Academic Publishers, Dordrecht (1994)
26. Moortgat, M., Morrill, G.V.: Heads and phrases. Type calculus for dependency and constituent structure. Ms OTS, Utrecht (1991)
27. Robinson, J.J.: Dependency structures and transformational rules. Language **46**(2) (1970) 259–285
28. Sleator, D., Temperly, D.: Parsing English with a Link Grammar. In: Proc. IWPT'93. (1993) 277–291
29. Tesnière, L.: Éléments de syntaxe structurale. Librairie C. Klincksieck, Paris (1959)