Dependency Structure Grammars

Denis Béchet [†], Alexander Dikovsky [‡], and Annie Foret [§]

Abstract. In this paper, we define Dependency Structure Grammars (DSG), which are rewriting rule grammars generating sentences together with their dependency structures, are more expressive than CF-grammars and non-equivalent to mildly context-sensitive grammars.
We show that DSG are weakly equivalent to Categorial Dependency Grammars (CDG) recently introduced in [6, 3]. In particular, these dependency grammars naturally express long distance dependencies and enjoy good mathematical properties.

1 Introduction

Dependency grammars (DGs) are formal grammars, which define syntactic relations between words in the sentences. Following to the tradition going back to L.Tesnière, the DGs are lexicalized and define the surface syntactic structure in terms of syntactic valences of individual words and of constraints imposed on valency saturation, in particular, on licensed feature values and on word order. There are numerous and rather different definitions of DGs (cf. [1, 2]). Most of them are not generative string or graph-substitution rule based grammars. This can be simply explained by the absence of substructure markers (nonterminals) in the dependency structures. Meanwhile, the formalization of dependency syntax in the form of generative style grammars is an important issue for various reasons. Firstly, such grammars allow for a straightforward interface relying compositional dependency structure with other compositional structures, for instance, with constituent structure or with semantic structure of some kind. Secondly, sometimes they allow for improvement of parsing performance, in particular, for disambiguiation using meta-rules or other means of compact encoding of unifiable substructures. Thirdly but not lastly, the rule-based formal grammars have remarkable mathematical properties, which are the source of well founded and efficient methods of analysis, translation, optimization and semantical interpretation of grammars.

Some definitions of generative dependency grammars can be found in the literature (cf. [7, 4, 5, 2]). In this paper, we develop the idea put forward in [4, 5]

[†] LINA, Université de Nantes, 2, rue de la Houssinière BP 92208 F 44322 Nantes cedex 3 France. Email: denis.bechet@univ-nantes.fr
http://www.sciences.univ-nantes.fr/info/perso/permanents/bechet/

[‡] LINA, Université de Nantes, 2, rue de la Houssinière BP 92208 F 44322 Nantes cedex 3 France. Email: alexandre.dikovsky@univ-nantes.fr
http://www.sciences.univ-nantes.fr/info/perso/permanents/dikovsky/

[§] IRISA - Université de Rennes 1, Campus Universitaire de Beaulieu, Avenue du Général Leclerc. 35042 Rennes Cedex France. Email: annie.foret@irisa.fr

to distinguish between local and long distance dependencies and to treat them differently: the former by the composition of right-hand-side dependency trees of the rules, and the latter by the unique global rule of pairing a long distance dependency valency with the *first available* (i.e. the closest not used) dual valency: **FA**-rule. Recently, this idea was implemented in the form of calculus of syntactic types: *Categorial Dependency Grammars (CDG)* generalizing classical categorial grammars [6, 3]. In this paper, we dramatically simplify the rather technical definition of *polarized dependency grammars* of [4, 5] by renouncing the tree constraints and considering general graph dependency structures. The resulting *generalized Dependency Structure Grammars (gDSG)* prove to be weakly equivalent to *generalized Categorial Dependency Grammars (gCDG)* resulting from CDG by a similar dependency structure generalization. This equivalence of two completely different simple and elegant formal models shows the invariant nature of the rule **FA**. At the same time, this equivalence proves that the languages in this family have an efficient polynomial parsing algorithm due to their gCDG definition, and that they enjoy good mathematical properties due to their gDSG definition.

The paper is organized as follows. In section 2, we introduce the generalized Dependency Structure Grammars and some their important particular cases. In the next section, we summarize the main definitions and notation of the Categorial Dependency Grammars and define the generalized Categorial Dependency Grammars. In section 3, we prove the equivalence of the two definitions. Finally, in section 4, we establish the results characterizing the expressive power and main properties of this class of dependency grammars.

2 Dependency Structure Grammars

2.1 Dependency valency

We follow the proposals in [5, 6] and specify long distance (in particular, non-projective discontinuous) dependencies by polarized dependency types, which we call *valences*. A *positive valency* specifies the name and the direction of an outgoing long distance dependency. The corresponding negative valency with the same name has the opposite direction and specifies the end of this incoming dependency (we say that the two valences are *dual*). Long distance dependencies are specified by correctly paired dual valences. In this pairing, positive valences needing the corresponding negative valency on the right and negative valences needing the corresponding positive valency on the right are considered as left brackets. Symmetrically, the valences needing the corresponding dual valences on the left are considered as right brackets.

For instance, the first member of the french discontinuous negation *ne .. pas* must have the left positive valency ($\nearrow n\!-\!compound$), whereas the second member must have the dual right negative valency ($\searrow n\!-\!compound$). Together they define the long distance dependency $n\!-\!compound$.

Formally, we consider a finite set $\mathbf{C}$ of elementary dependency types and introduce four *polarities*: left and right positive: $\nearrow, \nwarrow$ (*outgoing from* left (re-

spectively, right) *to* right (respectively, left)) and left and right negative: $\swarrow, \searrow$ (*incoming from* right (respectively, left) *to* left (respectively, right)). For each polarity v, there is the unique "dual" polarity $\breve{v}$: $\breve{\nearrow} = \searrow$, $\breve{\nwarrow} = \swarrow$, $\breve{\swarrow} = \nwarrow$, $\breve{\searrow} = \nearrow$. A *polarized valency* is an expression (vC), in which v is one of the four polarities and $C \in \mathbf{C}$. For instance, in the phrase *upon what dependency theory we rely*, the *right positive* valency $(\nwarrow pre{-}UPON{-}obj)$ of the transitive verb *rely* requires the beginning of the long distance dependency *pre–UPON–obj* relating this verb with the subordinate object *dependency theory* dislocated from right to left and headed by the preposition *'UPON'*. The end of this dependency will be required by the type of the preposition $UPON$ through the dual left negative valency $(\swarrow pre{-}UPON{-}obj)$ [1].

$\nearrow\mathbf{C}, \nwarrow\mathbf{C}, \swarrow\mathbf{C}$ and $\searrow\mathbf{C}$ denote the corresponding sets of polarized valences. For instance, $\nearrow \mathbf{C} = \{(\nearrow C) \mid C \in \mathbf{C}\}$ is the set of *left positive* valences. $V^+(\mathbf{C}) = \nearrow\mathbf{C} \cup \nwarrow\mathbf{C}$ is the set of positive valences, $V^-(\mathbf{C}) = \searrow\mathbf{C} \cup \swarrow\mathbf{C}$ is the set of those negative.

2.2 Generalized dependency structures

Definition 1 Potentials. *A* potential *is a string* $\Gamma \in \mathcal{P} =_{df} (V^+(\mathbf{C}) \cup V^-(\mathbf{C}))^*$.

Let $\Gamma = \Gamma_1(vC)\Gamma_2(\breve{v}C)\Gamma_3$ *and* $\Gamma' = \Gamma_1\Gamma_2\Gamma_3$ *be two potentials such that* $(vC) = (\nearrow A)$, $(\breve{v}C) = (\searrow A)$ *or* $(vC) = (\swarrow A), (\breve{v}C) = (\nwarrow A)$. *We say that* (vC) *is* **first available (FA)** *for* $(\breve{v}C)$ *in* Γ *and both are* neutralized *in* Γ' *(denoted* $\Gamma \twoheadrightarrow_{FA} \Gamma'$*) if* Γ_2 *has no occurrences of* (vC) *and* $(\breve{v}C)$. *This reduction of potentials* $\twoheadrightarrow_{FA}$ *is terminal and confluent. So each potential* Γ *has a unique FA-normal form* [2] *denoted* $[\Gamma]_{FA}$. *Therefore, we can define the* product $\odot$ *of potentials as follows:* $\Gamma_1 \odot \Gamma_2 =_{df} [\Gamma_1\Gamma_2]_{FA}$.

Clearly, this product is associative. So we obtain the *monoid of potentials* $\mathbf{P} = (\mathcal{P}, \odot)$ under the product $\odot$ with the unit ε.

Definition 2 Generalized dependency structures. *Let* W *and* N *be two disjoint sets of* terminals *and* nonterminals. *A* generalized dependency structure (gDS) *over* $W \cup N$ *is a graph* δ *with linearly ordered nodes in which :*

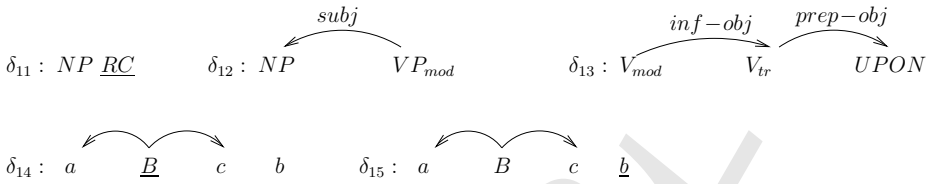
- the nodes are labeled by symbols in $W \cup N$,

- one maximal connected component D_0 *and one node* $n_0 \in D_0$ *are selected, called respectively* head component *and* head *of* δ [3]. *The decomposition of* δ *into maximal connected components (called below just* components*) will be denoted by* $\delta = \{\underline{D_0}, D_1, \ldots, D_k\}$.

Due to the linear order, δ *determines the string of node labels* $w(\delta) \in (W \cup N)^+$ *called* framework *of* δ. *We will also say that* δ *is a gDS of* $w(\delta)$. *In particular, each component* D_i *is a gDS of the corresponding string* $w(D_i)$.

[1] See [6] and [3] for more details.
[2] Irreducible potential.
[3] We visualize D_0 underlining its head n_0 if δ has at least two components.

Example 1 *For instance, the following graphs are dependency structures:*

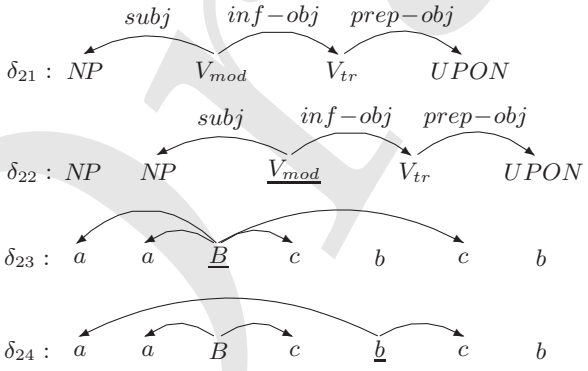


gDS δ_{11} has two components, the second is head. gDS δ_{12}, δ_{13} are dependency trees. The head of δ_{14} is B and the head of δ_{15} is b.

Definition 3 Composition of gDS. *Let $\delta_1 = \{\underline{D_0}, D_1, \ldots, D_k\}$ be a gDS. Let a nonterminal A have an occurrence in δ_1: $w(\delta_1) = xAy$ and δ_2 be a gDS 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 gDS 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 $w(\delta) = xw(\delta_2)y$.
4. The head of δ is the head of the component resulting from D_0.

$\delta = \delta_0[A_1, \ldots, A_n\backslash\delta_1, \ldots, \delta_n]$ will denote the result of simultaneous composition of DS $\delta_1, \ldots, \delta_n$ into $A_1, \ldots, A_n$ in δ_0.

Example 2 *The following gDS are compositions of the gDS in example 1:*



Namely, $\delta_{21} = \delta_{12}[VP_{mod}\backslash\delta_{13}]$, $\delta_{22} = \delta_{11}[CR\backslash\delta_{21}]$, $\delta_{23} = \delta_{14}[B\backslash\delta_{14}]$, $\delta_{24} = \delta_{14}[B\backslash\delta_{15}]$.

2.3 Grammar definition

Definition 4 *A* **generalized Dependency Structure Grammar (gDSG)** *is a system $G = (W, N, \mathbf{C}, S, R)$, where W, N and $\mathbf{C}$ are finite sets of terminals (words), nonterminals and elementary types, $S \in N$ is the axiom and R is a finite*

set of rules. Each rule r consists of a substitution $s(r)$ *of the form* $A \to \delta$, *where* $A \in N$ *and* δ *is a gDS, and of* potential assignments *of the form* $\omega(r,a)[\Gamma]$, *where* $\omega(r,a)$ *is an occurrence of a terminal* a *in* δ *and* Γ *is a (unique) potential in normal form* [4] *assigned to this occurrence.*

For each substitution $A \to \delta$, $A \to w(\delta)$ *is the corresponding* framework rule. *The* framework cf-grammar $f(G)$ *consists of all framework rules of* G.

Definition 5 Derivations. *In definition 4,* s *is a many-to-one relation between the rules of* G *and* $f(G)$. *It is naturally extended to trees. A* terminal *derivation tree* [5] T_0 *of* $f(G)$ *corresponds through* s *to a* composition tree T *of* G *if* T *results from* T_0 *by assigning to each non-terminal node* n *a rule* $r(n) \in R$ *such that* $s(r)$ *is applied to* n *in* T_0.

For each node n *of a composition tree* T, *we define its* potential $\pi(T,n)$ *and gDS* $gDS(T,n)$ induced *by* n *in* T *as follows:*
1. Let $n = a_i \in W$ *be a terminal node of* T, n' *be its parent node,* $\omega(r,a_i)$ *be the occurrence of* a_i *in the right-hand side of rule* $r = r(n')$ *and* $\omega(r,a_i)[\Gamma]$ *be its potential assignment. Then* $gDS(T,n) = a_i$ *and* $\pi(T,n) = \Gamma$. *We suppose that each valency* $v \in \Gamma$ *keeps the position* i *of* a_i *in the generated string* w *(denoted* v^i*). The positions are needed only for gDS construction and can be neglected if the gDS are not pertinent.*
2. Let $n = A \in N$ *be a node in* T *with assigned rule* $r(n) = (A \to \delta)$, *whose framework rule is* $A \to \alpha_1 \dots \alpha_k$. *This means that* n *has in* T k *sons:* $n_1, \dots, n_k$ *corresponding to* $\alpha_1, \dots, \alpha_k$ *(in this order). Let the potentials and the gDS of the sons be defined as:* $\pi(T,n_i) = \Gamma_i$ *and* $gDS(T,n_i) = \delta_i$, $1 \le i \le k$. *Then*

$$\pi(T,n) =_{df} \Gamma_1 \odot \dots \odot \Gamma_k$$
$$gDS(T,n) =_{df} \delta[\alpha_1 \dots \alpha_k \backslash gDS(T,n_1) \dots gDS(T,n_k)] \ \cup \Delta_n,$$

where Δ_n *is the set of all long distance dependencies* $(a_i \xleftarrow{C} a_j)$ *or* $(a_i \xrightarrow{C} a_j)$ *between terminals* a_i, a_j, *induced by neutralization of dual valences* $(\swarrow C)^i$, $(\nwarrow C)^j$ *(respectively* $(\nearrow C)^i, (\searrow C)^j$*).*

The maximal length of potentials $\pi(T,n)$ *in* T *is called* valency deficit *of* T *(denoted* $\sigma(T)$*).*

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

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

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

Intuitively, the derivation trees are induced by the framework grammar derivation trees, which correspond through s to composition trees. Only those composition trees derive gDS, in which all valencies are neutralized. Each derivation step can neutralize some dual dependency valences, and in this way, establish long distance dependencies between the words to which these valences are assigned.

[4] For instance, the rule $r = (A \to \underline{a}[\searrow D_1 \nearrow D_2]\ B)$ has the substitution $s(r) = (A \to \underline{a}\ B)$ and the assignment $\omega(r,a)[\searrow D_1 \nearrow D_2]$. We omit assignmens $\omega(r,a)[\varepsilon]$.
[5] I.e., in which all leaves are terminal.

Example 3 *For instance, the gDSG*

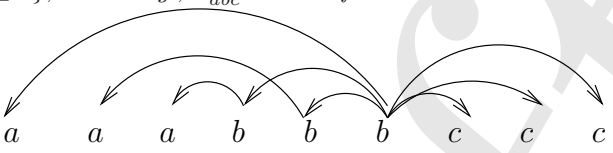
G_1 :

$$S \rightarrow a[\swarrow a] \quad \underline{S} \quad | \quad \overset{\frown}{A \quad c}$$

$$A \rightarrow b[\nwarrow a] \quad \overset{\frown}{A \quad c} \quad | \quad b[\nwarrow a]$$

generates the language $L(G_1) = \{a^n b^n c^n \mid n \geq 1\}$ *and the gDS-language* $gDS(G_1) = \{d^{(3)}_{abc} \mid n \geq 1\}$, *where e.g.,* $d^{(3)}_{abc}$ *has the form:*

$d^{(3)}_{abc}$:



The gDSG can generate dependency structures, which are arbitrary ordered graphs and not dependency trees. Even in the case, where the gDS in the rules have only dependency tree components, the generated structures may have cycles as it is the case of the following trivial gDSG:

$$S \rightarrow a[(\swarrow A)(\nearrow B)] \;\; b[(\searrow B)(\nwarrow A)] \;\; \underline{c}.$$

If we want that the grammars generate only dependency trees, then some additional constraints must be imposed.

2.4 Dependency Structure Grammars

We show the constraints, which guarantee only that the gDSG have the most important property of dependencies: *the uniqueness of the governor*. In particular, these constraints do not guarantee connectedness and cycle-freeness. The resulting Dependency Structure Grammars represent a reasonable compromise between acceptable divergence from classical dependency trees on the one hand, and simplicity of grammar rules and elimination of excess technical details on the other hand.

We split the set of nonterminals N in two parts: $N = N^+ \cup N^-$, $N^+ \cap N^- = \emptyset$. N^- corresponds to dependency structures with negative potential, and N^+ embodies the inherited through derivation impossibility of negative valences.

Definition 6 Dependency structures. *Let us call an oriented graph* P unique governor *if each node in* P *is entered by at most one arrow.*

A gDS $\delta = \{\underline{D_0}, D_1, \ldots, D_m\}$ *is a* dependency structure (DS) *if it is a unique governor graph and if each nonterminal* B *labelling a dependent node* [6] *is positive:* $B \in N^+$.

Clearly, the composition preserves such dependency structures.

Proposition 1 *For any DS* $\delta, \delta_1, \ldots, \delta_k$, $\delta[A_1, \ldots, A_n \backslash \delta_1, \ldots, \delta_k]$ *is a DS.*

Definition 7 *We call a potential* Γ non-negative *if* $\Gamma \in (V^+(\mathbf{C}))^*$, neutral *if* $\Gamma = \varepsilon$ *and* definitely negative *if* $\Gamma \in (V^+(\mathbf{C}))^* V^-(\mathbf{C})(V^+(\mathbf{C}))^*$.

[6] I.e. a node, into which a dependency enters.

Definition 8 *A* Dependency Structure Grammar (DSG) *is a gDSG* $G = (W, N, \mathbf{C}, S, R)$, *in which* $N = N^+ \cup N^-$, $N^+ \cap N^- = \emptyset$, $S \in N^+$ *and:*
(c_1) *in potential assignments* $\omega(r, a)[\Gamma]$, Γ *is either neutral, or non-negative, or definitely negative;*
(c_2) *in substitutions* $r = (A \rightarrow \{\underline{D_0}, D_1, \ldots, D_m\})$, *if a terminal* $a \in W$ *labels a non-head node of a component of* D_i *or it labels the head* n_0 *of* D_0 *and* $A \in N^+$, *then only a non-negative potential* Γ *can be assigned to a through an assignment rule* $\omega(r, a)[\Gamma]$*;*
(c_3) *if in a substitution* $A \rightarrow \delta$ $A \in N^+$ *and the head* n_0 *of* δ *is labeled with a nonterminal* B, *then* $B \in N^+$.

We denote the structure language of a DSG G by $DS(G)$. This notation is justified by the following proposition.

Proposition 2 *If* G *is a DSG, then* $gDS(G)$ *contains only terminal DS.*

Proof. Proposition 2 is immediately implied by the following lemma.

Lemma 1 *Let* T *be a derivation tree of a gDS* δ_T *with the head node* a_h. *Then:*
1. If T *is a derivation tree from a positive nonterminal* $A \in N^+$, *then* a_h *has no negative valences.*
2. For all nodes n *in* T *and for each terminal node* a *of* $gDS(T, n)$, *if among the valences assigned to* a *there is one not neutralized negative valency* v, *then* a *is not dependent in* $gDS(T, n)$.

Can be proven by induction on the structure of the derivation tree T. □

3 Categorial Dependency Grammars

In this section, we summarize the main notions related with the Categorial Dependency Grammars needed to define some their generalization.

3.1 Dependency types

Categorial dependency grammars are simply related with classical categorial grammars. They use "curried" variants of first order types: $[l_1 \backslash \ldots \backslash m / \ldots / r_1]$. In these types, all subtypes: left argument (l_i), right argument (r_i) and main (m) can be elementary or polarized. The elementary subtypes define local dependencies and the polarized subtypes define long distance dependencies. In particular, elementary left argument type l corresponds to the beginning of the local dependency l outgoing to the left, whereas main subtype l corresponds to the end of incoming local dependency l. As in DSG, the polarized subtypes represent long distance dependency valences. They have the same meaning. There is however a fundamental difference between the two formal models. In DSG, the linear order is directly defined by the right-hand-side gDS of rules. Categorial dependency grammars are completely lexicalized. To define a linear order on long distance

dependencies, they use so called "anchored" valences. For instance, in the sentence *It was yesterday that they had this meeting* the discontinuous dependency $it{-}cleft$ starting from the conjunction *that* must enter the expletive pronoun *It* in the position immediately preceding the main verb. To express this requirement, two adjacency markers: # and ♭ are applied to dependency valences. Assigning to *It* the type $\#(\swarrow it{-}cleft)$ one requires that the long distance dependency $it{-}cleft$ must enter *It* from the right and that the position of *It* must be *anchored* to some host word. To make *was* the host word for *It*, the type $[\flat(\swarrow it{-}cleft)\backslash S/subj/circ]$ is assigned to *was*. This type requires that the end of the long distance dependency $it{-}cleft$ must immediately precede *was* (i.e. be *anchored* on its left), that two local dependencies *subj* and *circ* must start from *was* to its right and that *was* becomes the root of the dependency tree if the three requirements are met. Below we summarize the definitions of dependency types and type calculus and address the reader to [6, 3] for more details.

We call syntactic types *categories*. Let $\mathbf{C}$ be a nonempty set of *elementary categories*. Elementary categories, e.g. *subj*, *inf-subj*, *dobj*, *det*, *modif*, etc. are dependency names. For instance, *subj* is the dependency, whose subordinate is a noun or a pronoun in the syntactic role of the subject and whose governor is a verb. Elementary categories may be *iterated*. For $a \in \mathbf{C}$, a^* denotes the corresponding *iterative* category. For instance, $modif^*$ is the type of iterated category $modif$. For a set $X \subseteq \mathbf{C}$, $X^* = \{C^* \mid C \in X\}$. The elementary and iterated categories are *local*.

The negative valences in $V^-(\mathbf{C})$ do not constrain the position of the end of the required long distance dependency. So they are called *loose*.

To specify the positions of the ends of long distance dependencies, we use two markers: # (*anchor*) and ♭ (*host*). For each negative valency $vC \in V^-(\mathbf{C})$, the expressions $\#(vC)$ and $\flat(vC)$ are the corresponding *anchor* and *host valences*. We distinguish *left-argument* and *right-argument* host valences and the corresponding *left* and *right positioned* anchor valences:

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

The sets $Host^l(\mathbf{C}), Host^r(\mathbf{C}), Anc^l(\mathbf{C})$ and $Anc^l(\mathbf{C})$ are supposed to be disjoint.

Definition 9 *The set* $Cat(\mathbf{C})$ *of* categories *is the least set such that:*
1. $\mathbf{C} \cup V^-(\mathbf{C}) \cup Anc(\mathbf{C}) \subset Cat(\mathbf{C})$.
2. For $C \in Cat(\mathbf{C})$, $A_1 \in (\mathbf{C} \cup \mathbf{C}^* \cup Host^l(\mathbf{C}) \cup \nwarrow \mathbf{C} \cup \searrow \mathbf{C})$ *and* $A_2 \in (\mathbf{C} \cup \mathbf{C}^* \cup Host^r(\mathbf{C}) \cup \nearrow \mathbf{C} \cup \swarrow \mathbf{C})$, *the categories* $[A_1\backslash C]$ *and* $[C/A_2]$ *also belong to* $Cat(\mathbf{C})$.

Categories, which cannot have left arguments in $\searrow \mathbf{C}$ *and right arguments in* $\swarrow \mathbf{C}$ *are called* dependency categories *(denoted* $DCat(\mathbf{C})$*); those which do not have subcategories in* $V^-(\mathbf{C}) \cup V^+(\mathbf{C})$, *are called* continuous dependency categories *(denoted* $CCat(\mathbf{C})$*).*

We suppose that the constructors $\backslash$, $/$ are associative. So every complex category α can be presented in the form $\alpha = [L_k\backslash \dots L_1\backslash C/R_1 \dots /R_m]$.

For instance, $[\flat^l(\swarrow clit{-}dobj)\backslash subj\backslash S/aux]$ is one of possible categories of an auxiliary verb in French, which defines it as the host word for a cliticized direct object, requires a local subordinate subject on its left and a local subordinate through dependency *aux* on its right.

3.2 Definition of Categorial Dependency Grammars

Definition 10 *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-set-valued function on* W *such that* $\delta(a) \subset Cat(\mathbf{C})$ *for each word* $a \in W$. G *is a* Categorial Dependency Grammar (CDG) *if* $\delta(W) \subseteq DCat(\mathbf{C})$.

We index categories by their positions in a string of categories related by G with a given sentence $w = a_1 \dots a_n$: α^i is a (positioned) category of a dependency structure with the root position a_i. As in gDSG, these indices serve only to define dependency structures.

Definition 11 *A* D-sentential form *of a sentence* $w = a_1 \dots a_n \in W^+$ *is a pair* (Δ, Γ), *where* Δ *is an oriented labelled graph with the set of nodes* $V = \{a_1, \dots, a_n\}$ *and a set of arcs labeled by elementary categories, and* Γ *is a nonempty string of positioned categories.*

An initial D-sentential form *of* $w = a_1 \dots a_n$ *is an expression* $((V, \emptyset), C_1^1 \dots C_n^n)$, *in which* $C_i \in \delta(a_i)$ *for all* $1 \le i \le n$. *D-sentential forms* (Δ, S^j) *are* terminal.

gCDG derivations are proofs in the following dependency calculus.

Definition 12 *Sub-commutative dependency calculus (only left constructor rules* R^l *are presented; the corresponding right constructor rules* R^r *are similar).*
Local dependency rule*:*
$\mathbf{L}^l$. $((V,E), \Gamma_1 C^i[C\backslash\beta]^j\Gamma_2) \vdash ((V, E \cup \{a_i \xleftarrow{C} a_j\}), \Gamma_1\beta^j\Gamma_2)$ *for* $C \in \mathbf{C}$.
Iterative dependency rules*:*
$\mathbf{I^l}$. $((V,E), \Gamma_1 C^i[C^*\backslash\alpha]^j\Gamma_2) \vdash ((V, E \cup \{a_i \xleftarrow{C} a_j\}), \Gamma_1[C^*\backslash\alpha]^j\Gamma_2)$ *for* $C \in \mathbf{C}$.
$\mathbf{\Omega^l}$. $((V,E), \Gamma_1[C^*\backslash\alpha]^i\Gamma_2) \vdash ((V,E), \Gamma_1\alpha^i\Gamma_2)$ *for* $C \in \mathbf{C}$.
Argument valency rule:
$\mathbf{V^l}$. $((V,E), \Gamma_1[\beta\backslash\alpha]^i\Gamma_2) \vdash ((V,E), \Gamma_1\beta^i\alpha^i\Gamma_2)$, *where* β *is a host or polarized valency.*
Anchored dependency rule:
$\mathbf{A^l}$. $((V,E), \Gamma_1 \#^l(\alpha)^i \flat^l(\alpha)^j \Gamma_2) \vdash ((V,E), \Gamma_1\alpha^i\Gamma_2)$ *for* $\#^l(\alpha) \in Anc^l(\mathbf{C})$ *and* $\flat^l(\alpha) \in Host^l(\mathbf{C})$.
Sub-commutativity rule:
$\mathbf{C^l}$. $((V,E), \Gamma_1\ C^i\alpha^j\ \Gamma_2) \vdash ((V,E), \Gamma_1\ \alpha^j C^i\ \Gamma_2)$ *if* $\alpha \in (V^-(\mathbf{C}) \cup V^+(\mathbf{C})$ *and*
(i) $C \in Host(\mathbf{C})$ *or*
(ii) $C \in Cat(\mathbf{C})$ *and* C *has no subexpressions* $\alpha, \#(\alpha), \flat(\alpha)$, *and* $\breve{\alpha}$.
Long distance dependency rule:
$\mathbf{D^l}$. $((V,E), \Gamma_1(\swarrow C)^i(\nwarrow C)^j\Gamma_2) \vdash ((V, E \cup \{a_i \xleftarrow{C} a_j\}), \Gamma_1\Gamma_2)$ *for* $(\swarrow C) \in {\swarrow}\mathbf{C}$ *and* $(\nwarrow C) \in {\nwarrow}\mathbf{C}$.

The one-step provability relation in this calculus is denoted by $\vdash^R$, *where* R *is one of the rules above, or just by* $\vdash$, *if* R *is irrelevant. The transitive closure of this relation is denoted by* $\vdash^*$.

Besides this sub-commutative calculus, we consider its restriction to the continuous categories in $CCat(\mathbf{C})$ *with the additional equivalence* $\#^\alpha(t) \equiv \flat^\alpha(t)$ *and to the first three rules* $\mathbf{L}, \mathbf{I}$ *and* $\mathbf{\Omega}$. *We call this restricted calculus* projective.

The one-step provability relation in the projective calculus is denoted by $\vdash_p^R$ *(or just* $\vdash_p$*). Its transitive closure is denoted by* $\vdash_p^*$.

We see that rule **L** is a direct analogue of the classical elimination rule. Rules **I** and **Ω** extend **L** to the iterative categories. In projective calculus, anchor and host types are not distinguished, e.g. $[\alpha/\flat^r(d)]\#^r(d) \vdash_p \alpha$. Particular are the polarized valences' rules. Rule **V** extracts non-local valences from complex categories. Rule **C** moves the valences in the indicated directions towards the **first available** valency, to which one can apply rules **A** or **D**. Rule **D** adds a long distance dependency C, when two loose dual valences with the same name C become adjacent. The crucial difference between gCDG and CDG is that due to negative argument subtypes available in gCDG, the rule **D** can violate the uniqueness of the governor, which is impossible in CDG, where non-local argument subtypes are positive or host. Rule **A** verifies that an anchored valency $\#(\alpha)$ has become adjacent to the corresponding host valency $\flat(\alpha)$, consumes $\flat(\alpha)$ and looses $\flat(\alpha)$. Intuitively, this means that α is well-placed with respect to the category with the corresponding host argument. If this test succeeds, α becomes available to the long distance dependency rule **D**. We address the reader to [6, 3] for linguistic examples.

Definition 13 *Let* $G = (W, \mathbf{C}, S, \delta)$ *be a gCDG. A gDS* D *is assigned by* G *to a sentence* w *(denoted* $G(D, w)$*) if* $(\Delta_0, \Gamma_0) \vdash^* (D, S^j)$ *for some initial sentential form* (Δ_0, Γ_0) *of* w *and some* $1 \leq j \leq n$.
The D-language *generated by* G *is the set of gDS* $gDS(G) =_{df} \{D \mid \exists w\ G(D, w)\}$.
The language *generated by* G *is the set of sentences* $L(G) =_{df} \{w \mid \exists D\ G(D, w)\}$.

Proposition 3 *1. For each CDG* G, $gDS(G)$ *contains only DS.*
2. If gCDG is projective, it is a CDG and $DS(G)$ *contains only projective DS.*

We denote by $\mathcal{L}(gCDG), \mathcal{L}(CDG)$ and $\mathcal{L}(pCDG)$ the families of languages generated by gCDG, CDG and projective CDG. If G is a CDG, then we use notation $DS(G)$ in the place of $gDS(G)$.

gCDG have the following fundamental property established in [3].

Definition 14 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 a string* $\gamma \in Cat(\mathbf{C})^*$ *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* $a \in V^+(\mathbf{C})$.

Definition 15 *For a category* $C = [\alpha D^*\backslash\beta]$, *the categories* $[\alpha\beta]$, $[\alpha D\backslash\beta]$, $[\alpha D\backslash D\backslash\beta]$, $[\alpha D\backslash D\backslash D\backslash\beta]$, *etc. are* realizations *of* C *(similar for right iterative categories). To obtain a realization of a string of categories* $\gamma \in Cat(\mathbf{C})^+$, *each of its elements having iterative subcategories should be replaced by one of its realizations. Let* $R(\gamma)$ *denote the set of all realizations of* γ.

Theorem 1 *Let* $G = (W, \mathbf{C}, S, \delta)$ *be a gCDG.* $x \in L(G)$ *iff there is a string of categories* $\alpha \in \delta(x)$ *such that for some its realization* $\gamma \in R(\alpha)$*:*
1. $\|\gamma\|_l \vdash_p^* S$, *2.* $[\|\gamma\|_v]_{FA} = \varepsilon$.

In fact, this property is proven for CDG but the proof holds for gCDG too.

Corollary 1 *[3] There is a polynomial time parsing algorithm for gCDG.*

4 Expressive power of gDSG

Definition 16 *A gDSG* $G = (W, N, \mathbf{C}, S, R)$ *is in generalized Greibach normal form (GNF) iff for each rule* $A \to \delta \in R$, $w(\delta) \in WN^*$.

Remark 1 *The condition* $w(\delta) \in WN^*$ *is the conjunction of three conditions: (i) all* $w(\delta)$ *are not empty, (ii) the first symbol of* $w(\delta)$ *must be a terminal, (iii) all other symbols in* $w(\delta)$ *must be non-terminals.*

The first condition is always true for gDSG and the third one is not difficult to obtain because it is always possible to introduce, for each terminal, a new non-terminal that replaces it in the right members of the rules, where the condition is not true. Thus, only the second condition is not trivial.

Proposition 4 *For any gDSG* G, *a weakly equivalent gDSG* G' *in generalized GNF can be constructed.*

Proof. Let $G = (W, N, \mathbf{C}, S, R)$ be a gDSG. As we are interested only in weak equivalence, we can chose arbitrary heads and dependencies to transform the frame rules to the form $N \to W(W \cup N)^*$. We follow the Greibach's construction and proceed by induction on the number of *critical* non-terminals, i.e. the non-terminals occurring in the first position of right-hand-sides of framework rules:

$$n = \#(\{A \in N \mid \exists (B \to \delta) \in R\ (w(\delta) \in A(W \cup N)^*\})).$$

– In the case of $n = 0$, we already have a gDSG in generalized GNF.
– If $n > 0$, let A be one of these n non-terminals. Let A' be a new non-terminal and $N' =_{df} N \cup \{A'\}$. Let us classify the rules of R corresponding to the following framework rules:

$$\begin{array}{lll} A \to A & & (1) \\ A \to B_1 \cdots B_k & k \geq 1, B_1 \cdots B_k \in (W \cup N)^+, B_1 \neq A & (2) \\ A \to AB_1 \cdots B_k & k \geq 1, B_1 \cdots B_k \in (W \cup N)^+ & (3) \\ C \to AB_1 \cdots B_k & k \geq 0, B_1 \cdots B_k \in (W \cup N)^*, C \in N, C \neq A & (4) \end{array}$$

For $1 \leq i \leq 4$, we denote $R_{(i)} \subset R$ the rules in the class (i). The rules in $R_{(3)}$ and $R_{(4)}$ need to be modified. We define successively:

$$\begin{array}{ll} R_A & = R_{(2)} \cup \{A \to \delta A' \parallel A \to \delta \in R_{(2)}\} \\ R_{A'} & = \{A' \to \delta[A\backslash\epsilon] \parallel A \to \delta \in R_{(3)}\} \cup \{A' \to \delta[A\backslash\epsilon]A' \parallel A \to \delta \in R_{(3)}\} \\ R_C & = \{A' \to \delta[A\backslash\delta'] \parallel C \to \delta \in R_{(4)} \wedge A \to \delta' \in R_A\} \\ R' & = (R - R_{(1)} - R_{(2)} - R_{(3)} - R_{(4)}) \cup R_A \cup R_{A'} \cup R_C \\ G' & = (W, N', \mathbf{C}, S, R') \end{array}$$

The framework languages of G and G' are the same. Let T be a derivation tree of a string w in G. There exists a derivation tree T' of w in the framework grammar of G'. In T, each leaf is associated to a potential. Let us keep in T' the same potentials assignment as in T and extend the frame rules to the corresponding dependency structure rewriting rules. Then T' will become a composition tree in G'. Given that the product $\odot$ is associative, in the transformed tree T' exactly the same potential is calculated. So T' is a derivation tree for w in G', which proves that $w \in L(G')$ and $L(G) \subseteq L(G')$. The reverse inclusion is similar, so G and G' are weakly equivalent. Now, the induction hypothesis can be applied because the critical non-terminals of G' are fewer than those of G. □

Theorem 2 *gDSG and gCDG are weakly equivalent.*

Proof. ($\Rightarrow$) To prove that $\mathcal{L}(gDSG) \subseteq \mathcal{L}(gCDG)$, we use a gDSG in generalized GNF. Let $G = (W, N', \mathbf{C}, S, R')$ be such a gDSG. We will simulate G by the gCDG $G' = (W, \mathbf{C}, S, \lambda)$, where the lexicon λ is computed from G as follows.

Let $r = (A \to \delta) \in R$ be a rule of G, whose framework rule has the form $A \to aB_1 \cdots B_i, a \in W$, and let $\omega(r, a)[\Gamma]$ be a potential assignment. Keeping in mind the associativity of potential product and the sub-commutativity rule $\mathbf{C}$, we can group together similar valences and represent Γ in the form:
$\Gamma \equiv (\nwarrow C_1) \cdots (\nwarrow C_j)(\swarrow D_1) \cdots (\swarrow D_k)(\searrow E_1) \cdots (\searrow E_l)(\nearrow F_n) \cdots (\nearrow F_n)$.
To these rules we associate in $\lambda(a)$ the category:

$$\begin{array}{l} (\nwarrow C_1)\backslash \cdots \backslash (\nwarrow C_j)\backslash (\swarrow D_1)\backslash \cdots \backslash (\swarrow D_k)\backslash A/B_1/ \cdots \\ \qquad \cdots /B_i/(\searrow E_1)/ \cdots /(\searrow E_l)/(\nearrow F_n)/ \cdots /(\nearrow F_n). \end{array}$$

The equivalence $L(G) = L(G')$ is relatively evident [7]. The first part $L(G) \subseteq L(G')$ holds because a derivation tree of any string $w \in L(G)$ uniquely determines

[7] This construction cannot serve to prove the strong equivalence, because in the case, when the head valency is negative, the resulting type has a negative argument subtype, which is impossible in CDG.

a sequence of reduction steps of categories assigned to w by G'. Indeed, the potential of a leaf of the derivation tree constitutes the part of the category determining the same long distance dependencies of a as those defined by the rule r. The rest of the category is uniquely determined by the rule r. One should first eliminate all long distance dependency valences (which is always possible), and then apply the category to its argument subtypes. This application is also possible because it directly simulates the application of the framework rule $w(r)$. This means that, using this tactics, the sequence of categories assigned by λ to the string w following the structure of the derivation tree of w in G will be reduced to S.

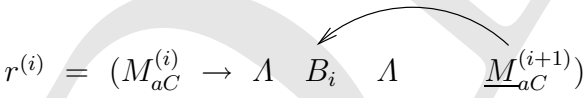
The converse inclusion $L(G') \subseteq L(G)$ is similar and follows from the fact that in a reduction to S of categories assigned by the lexicon λ, we can always start with reductions of long distance dependencies and continue with reductions of local dependencies.

($\Leftarrow$) The converse relation between the two families is stronger: for each gCDG $G_1 = (W, \mathbf{C}, S, \lambda)$, we can construct a gDSG $G_2 = (W, N, \mathbf{C}, S, R)$ such that $\Delta(G_2) = \Delta(G_1)$. This strong simulation is implied by theorem 1. Namely, the grammar G_2 is defined as the union $\bigcup_{a \in W, C \in \lambda(a)} \mathcal{M}(a, C)$, where each module $\mathcal{M}(a, C)$ is defined as follows.

Let us suppose for simplicity that in $\|C\|_l = [\alpha \backslash B / \beta]$ $\alpha \neq \varepsilon$ and $\beta = \varepsilon$. The three other cases are similar. Then $\alpha = B_n \backslash \cdots \backslash B_1$ for some $n > 0$. In this case,

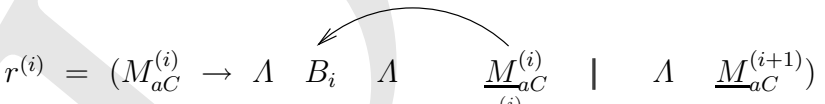
$$\mathcal{M}(a, C) =_{df} \{r^{(0)}, r^{(1)}, \ldots, r^{(n)}, r^{(n+1)}\},$$

where $r^{(0)} = (M_C \rightarrow \Lambda \ \underline{M}^{(1)}_{aC} \ \Lambda)$, $M_C = B$, if $B \neq \varepsilon$ and $M_C = E$ otherwise, $r^{(n+1)} = (M^{(n+1)}_{aC} \rightarrow a[\|C\|_v])$, $\Lambda \in \{E, \varepsilon\}$, and the resting rules $r^{(i)}$ are as follows:



$$r^{(i)} = (M^{(i)}_{aC} \rightarrow \Lambda \quad B_i \quad \Lambda \qquad \underline{M}^{(i+1)}_{aC})$$

if B_i is not iterative and



$$r^{(i)} = (M^{(i)}_{aC} \rightarrow \Lambda \quad B_i \quad \Lambda \qquad \underline{M}^{(i)}_{aC} \quad | \quad \Lambda \quad \underline{M}^{(i+1)}_{aC})$$

if it is iterative. In this construction, E and $M^{(i)}_{aC}$ are new pairwise different non-terminals different from all types. The equality $\Delta(G_2) = \Delta(G_1)$ immediately follows from theorem 1. □

Without constraint that gDS must be trees, the main result of [5] can be easily carried over to gDSG.

Theorem 3 *If in a gDSG G the valency deficit $\sigma(T)$ of correct terminal derivation trees is uniformly bounded by a constant c then G generates a CF-language.*

Proof. We can simply consider nonterminals $A[\Gamma]$, where Γ is a potential of the size not exceeding c, and define the rules so that for each node n of a complete derivation tree T its label should be $A[\pi(T,n)]$, A being the original nonterminal label of n. Clearly, $S[\varepsilon]$ becomes the axiom. □

Using the classical constructions, one can easily prove that $\mathcal{L}(gDSG)$ is an abstract family of languages.

Proposition 5 *$\mathcal{L}(gDSG)$ is closed under ε-free homomorphism, inverse homomorphism, intersection with regular sets, union, concatenation and $+$.*

Seemingly, $\mathcal{L}(gDSG)$ is not closed under intersection and complementation.

Conjecture *The copy language $L_{copy} = \{wcw \mid w \in \{a,b\}^*\}$ cannot be generated by a gDSG.*

Meanwhile, the complement of L_{copy} is linear and so belongs to $\mathcal{L}(gDSG)$. It is also well-known that L_{copy} is generated by a basic TAG. On the other hand, in [6, 3] it is proven that each language $L^{(m)} = \{d_0 a_0^n d_1 a_1^n \dots d_m a_m^n d_{m+1} | n \geq 0\}$ is generated by a CDG. So they can be generated by gCDG. Meanwhile, starting from $m = 5$, the languages $L^{(m)}$ cannot be generated by basic TAG. The languages $L^{(m)}$ are mildly context-sensitive [9]. This leads to the question of comparison of mildly CS languages and gDSG-languages. Seemingly, the two families are incomparable. Indeed, there is another strong conjecture that the mildly CS grammars cannot generate the language MIX of Emmon Bach consisting of all permutations of words $a^n b^n c^n, n > 0$:

$$MIX = \{w \in \{a,b,c\}^+ \mid |w|_a = |w|_b = |w|_c\}.$$

At the same time, we show that MIX is generated by a CDG.

Theorem 4 *There is a CDG generating MIX.*

Proof. We can construct a CDG G_{mix} with only loose valences and with only anchored valences. We show the former, because it is simpler:

TABLE OF	CATEGORY	ASSIGNMENTS
left	right	middle
$a \mapsto [\nwarrow B \setminus \nwarrow C \setminus S]$	$a \mapsto [S \,/\, \nearrow C \,/\, \nearrow B]$	$a \mapsto [\nwarrow B \setminus S \,/\, \nearrow C], [\nwarrow C \setminus S \,/\, \nearrow B]$
$a \mapsto [\nwarrow B \setminus \nwarrow C \setminus S \setminus S]$	$a \mapsto [S \setminus S \,/\, \nearrow C \,/\, \nearrow B]$	$a \mapsto [\nwarrow B \setminus S \setminus S \,/\, \nearrow C], [\nwarrow C \setminus S \setminus S \,/\, \nearrow B]$
$b \mapsto \swarrow B$	$b \mapsto \searrow B$	
$c \mapsto \swarrow C$	$c \mapsto \searrow C$	

Inclusion ($\subseteq$). $L(G_{mix}) \subseteq MIX$.
Let us consider the following *commutative* group interpretation of non-iterative categories (where $k_{l,x}$, $k_{r,x}$ are new symbols for each elementary x):
$< p >= p$ for elementary p,
$< [x \setminus y] >=< x >^{-1} < y >$, $< [y \,/\, x] >=< y >< x >^{-1}$,
$< \swarrow x >= k_{l,x}^{-1}$, $< \searrow x >= k_{r,x}^{-1}$, $< \nwarrow x >= k_{l,x}$, $< \nearrow x >= k_{r,x}$.

Fact. $\Gamma \vdash S$ implies $< \Gamma >= S$. (By evident induction on the derivation length.)

Being applied to the categories of G_{mix}, this interpretation shows that the number of a, b and c is the same in all $w \in L(G_{mix})$.

Inclusion ($\supseteq$). $MIX \subseteq L(G_{mix})$.
Let us consider a word $w_0 \in MIX$. We construct a canonical assignment of categories to the occurrences of a, b, c in w_0 as follows.
Canonical assignment algorithm CCA
$w := w_0$;
WHILE $w \neq \varepsilon$
DO
Phase I. Basic triangulation
FIND in w the *leftmost* occurrences α, β such that:
$w = u_1\alpha u_2\beta u_3$, where $u_2 \in c^*, \alpha \neq \beta, \alpha, \beta \in \{a, b\}$;
FIND in w the occurrence γ of c *closest* to α, if $\alpha = a$, else closest to β;
IF the selected $a \in \{\alpha, \beta\}$ is leftmost in w_0
THEN $X := S$;
ELSE $X := S\backslash S$
END;
CASE
$w = v_1\gamma v_2\alpha v_3\beta v_4 \ \wedge \ \alpha = a \rightarrow \alpha := [\nwarrow C\backslash X/\nearrow B]; \gamma := \swarrow C; \beta := \searrow B;$
$w = v_1\gamma v_2\alpha v_3\beta v_4 \ \wedge \ \alpha = b \rightarrow \beta := [\nwarrow B\backslash\nwarrow C\backslash X]; \gamma := \swarrow C; \alpha := \swarrow B;$
$w = v_1\alpha v_2\gamma v_3\beta v_4 \ \wedge \ \alpha = a \rightarrow \alpha := [X/\nearrow B/\nearrow C]; \gamma := \searrow C; \beta := \searrow B;$
$w = v_1\alpha v_2\gamma v_3\beta v_4 \ \wedge \ \alpha = b \rightarrow \beta := [\nwarrow C\backslash\nwarrow B\backslash X]; \gamma := \swarrow C; \alpha := \swarrow B;$
$w = v_1\alpha v_2\beta v_3\gamma v_4 \ \wedge \ \alpha = a \rightarrow \alpha := [X/\nearrow C/\nearrow B]; \gamma := \searrow C; \beta := \searrow B;$
$w = v_1\alpha v_2\beta v_3\gamma v_4 \ \wedge \ \alpha = b \rightarrow \beta := [\nwarrow B\backslash X/\nearrow C]; \gamma := \searrow C; \alpha := \swarrow B$
END;
Phase II. Elimination
$w := v_1v_2v_3v_4$
END

It is easy to see that **CCA** exits successfully the loop on the condition $w = \varepsilon$ for each $w_0 \in MIX$. Being applied to w_0, **CCA** defines the canonical assignment of categories $\mathbf{CCA}(w_0)$. The inclusion $MIX \subseteq L(G_{mix})$ is implied by the following fact.

Fact. $\mathbf{CCA}(w_0) \vdash S$ holds for all $w_0 \in MIX$.
(By evident induction on the number of a.) $\square$

5 Conclusions

We can resume the relations between structure languages and languages generated by the dependency grammars considered in this paper as follows:

$$\mathcal{D}(CDG^{proj}) \subsetneq \mathcal{D}(CDG) \subsetneq \mathcal{D}(gCDG) \subseteq \mathcal{D}(gDSG) \text{ and}$$
$$CFL = \mathcal{L}(CDG^{proj}) = \mathcal{L}(gDSG^{\sigma<\omega}) \subsetneq \mathcal{L}(CDG) \subseteq \mathcal{L}(gCDG) = \mathcal{L}(gDSG),$$

where CDG^{proj} is the class of projective CDG and $gDSG^{\sigma<\omega}$ is the class of gDSG with bounded valency deficit.

The dependency structure and categorial dependency grammars can be easily adopted to practical large scale definitions of surface dependency syntax of natural languages. For this, one should relate dependency names with bounded length feature value products admitting feature unification and value propagation through dependencies. Besides this, the explicit use of anchored categories in DSG and CDG make possible to express a variety of word order constraints. In fact, the potential assignments are closely related to Debusmann and Duchier's formulation of dependency grammar [8]. However, the FA-constraint excludes crossing of similarly labeled long distance dependencies.

CDG and DSG have an efficient parsing algorithm ($\mathbf{O}(n^5)$ in the worst case) [3]. In practice, the valency deficit is bounded by a small constant (2 or 3). In this situation, this parsing algorithm has complexity $\mathbf{O}(n^3)$ even if there are discontinuous long distance dependencies. So the dependency grammars studied in this paper represent an interesting class of grammars competitive with respect to mild context-sensitive grammars.

References

1. Les grammaires de dépendance. In Sylvain Kahane, editor, *Traitement automatique des langues*, volume 41, Paris, 2000. Hermes.
2. Proc. of the workshop "Recent Advances in Dependency Grammars". in conjunction with coling 2004. In Geert-Jan M. Kruijff and Denys Duchier, editors, *"Recent Advances in Dependency Grammars". COLING'04 Workshop, August 28 2004*, Geneva, Switzerland, August 2004.
3. Michael Dekhtyar and Alexander Dikovsky. Categorial dependency grammars. In *Proc. of Int. Conf. on Categorial Grammars*, pages 76–91, Montpellier, France, 2004.
4. Alexander Dikovsky. Grammars for local and long dependencies. In *Proc. of the 39th Intern. Conf. ACL'2001*, pages 156–163. ACL & Morgan Kaufman, 2001.
5. Alexander Dikovsky. Polarized non-projective dependency grammars. In Ph. de Groote, G. Morill, and Ch. Retoré, editors, *Proc. of the Fourth Intern. Conf. on Logical Aspects of Computational Linguistics*, Lecture Notes in Artificial Intelligence. vol. 2099, pages 139–157. Springer, 2001.
6. Alexander Dikovsky. Dependencies as categories. In G-J.M. Kruijff and D. Duchier, editors, *Proc. of Workshop "Recent Advances in Dependency Grammars". In conjunction with COLING 2004*, pages 90–97, Geneva, Switzerland, August, 28th 2004.
7. Alexander Dikovsky and Larissa Modina. Dependencies on the other side of the Curtain. *Traitement Automatique des Langues (TAL)*, 41(1):79–111, 2000.
8. Denis Duchier and Ralph Debusmann. Topological dependency trees: A constraint based account of linear precedence. In *Proc. of the 39th Intern. Conf. ACL'2001*, pages 180–187. ACL & Morgan Kaufman, 2001.
9. Aravind K. Joshi, Vijay K. Shanker, and David J. Weir. The convergence of mildly context-sensitive grammar formalisms. In P. Sells, S. Shieber, and T. Wasow, editors, *Foundational issues in natural language processing*, pages 31–81, Cambridge, MA, 1991. MIT Press.