Categorial Dependency Grammars

Michael Dekhtyar [a,1] Alexander Dikovsky [b,1]

[a] *Dept. of CS, Tver St. Univ., Tver, Russia, 170000*
Michael.Dekhtyar@tversu.ru

[b] *LINA, Université de Nantes, 2, rue de la Houssinière, BP 92208 44322 Nantes Cedex 03 France*
Alexandre.Dikovsky@lina.univ-nantes.fr

Abstract

Subcommutative Categorial Dependency Grammars (CDGs) introduced in this paper express projective and discontinuous dependencies in classical categorial grammar terms extended by oriented polarized valencies. They are defined in terms of a simple calculus of dependency types with a bounded commutativity rule. CDGs generate non-CF languages. At that, they are parsed in polynomial time.

Key words: Dependency grammar, discontinuous dependency, polynomial parser

1 Introduction

Dependency grammars (DGs) are formal grammars assigning *dependency trees* (DTs) to well-formed sentences. A DT of a sentence is a labelled arrows tree whose nodes are the words of the sentence. A rather formal description of DGs and DG syntax was given by L. Tesnière [Tes59]. The first exact definitions are due to D. Hays [Hay60] and H. Gaifman [Gai61].

The basic syntactic principle behind the DGs is quite different from that of syntagmatic grammars. They are designed for and more adapted to definitions of binary relations between wordforms (*syntactic dependencies*) than to definitions of sentence constituents. Meanwhile, historically the first DGs define in fact both. The Hays-Gaifman's DGs have much in common with the *categorial*

[1] This work was sponsored by the Russian Fundamental Studies Foundation (Grant 04-01-00565).

grammars (CGs) in [BHGS60]. Both are completely lexicalized, use syntactic types in the place of rewriting rules, naturally fit functional semantic structures and are equivalent to CF-grammars as far as only the weak expressive power is concerned and the core syntax is considered. The grammatical categories of DGs *position the subordinates with respect to their governors*. In this manner, they define not only the binary relations "governor → subordinate", whose union forms a tree, but also (due to the order given) the projections of words on the sentence. These projections form a system of constituents with the projected words serving as the constituents' *heads*. From the 70ies, it is known ([Gla66,Rob70]) [2] that this link between the two structures is reversible: a selection of one immediate head per constituent induces a unique DT by the following induction: $C \subsetneq C' \Rightarrow (ImmHead(C') = ImmHead(C) \vee root(ImmHead(C')) \rightarrow^* root(ImmHead(C)))$. This "structural equivalence" produced an illusion that DTs are *byproduct of head selection in constituent structures*. It may then seem that the syntagmatic grammars with explicit head selection (such as LFG [RJ82] or HPSG [PS94]) or those with implicit head selection (e.g., TAGs [JLT75] and CGs [BHGS60,Lam58]) can also be considered as DGs. However, this resemblance is very superficial and is immediately lost as far as the strong expressive power is concerned. For instance, the DTs assigned to constituent structures with selected heads are always *projective*: the projections of words fill continuous segments. Meanwhile, discontinuous non-projective dependencies are inevitable in languages. They often mark communicative structure (e.g. topicalization) and special constructions encoding complex semantic relations (e.g. clefting, subject or object extraction in relative clauses, object clitisation etc.). Still more important is the fact that this technical resemblance *does not preserve the intended syntactic types*. The reason is that the syntactic functions of the dependencies directly assigned to words (*word driven dependencies*) differ from those of the dependencies determined by head selection (*head driven dependencies*). Basically, the difference is that an individual dependency between the governor G and the subordinate S is naturally defined in terms of these words' classes and features, whereas the heads represent their projections (constituents), therefore a dependency of one head on another is more naturally defined in terms of types of constituents and their head features. The former represent word valencies, which is closer to the original Tesnière's dependency grammar idea, the latter represent phrase valencies and so are closer to the X-bar syntactic relations [Jac77]. Intuitively, a word driven dependency $G \xrightarrow{D} S$ represents the constraints under which "G licenses S". D may constrain lexical and grammatical feature values, order relations, pronominalization possibilities, etc. of G and S (sometimes also of their neighbors). For instance, extremely simplifying, the dependency $attr-rel$, as in *filles dequelles je parle* (where $filles \xrightarrow{attr-rel} dequelles$ and $dequelles \xrightarrow{rel} parle$), applies when G is a

[2] See [DM00] for details.

noun, S is a relative pronoun on which depends through rel a transitive verb V, whose second argument is co-referential with G and also G and S agree in number and gender.

Another simplified example: the dependencies $TO-obj$ and $pre-TO-obj$ encode the relations between words x, y with the following feature structures (in HPSG-like notation):

$$\left[\begin{array}{ll} dep: & TO-obj \\ gov: & \left[\begin{array}{ll} node: & \Box^1\ x:V \\ lex: & l_x \notin \{mod, copul\} \\ gram: & trans \\ act_2: & [head: \Box^2 Pp: TO] \end{array}\right] \\ sub: & \Box^2\ y \\ WO: & \Box^1 < \Box^2 \end{array}\right] \quad \left[\begin{array}{ll} dep: & pre-TO-obj \\ gov: & \left[\begin{array}{ll} node: & \Box^1\ x:V \\ lex: & l_x \notin \{mod, copul\} \\ gram: & trans \\ act_2: & [head: \Box^2 Pp: TO] \end{array}\right] \\ sub: & \Box^2\ y \\ WO: & \Box^1 > \Box^2 \end{array}\right]$$

(cf.$(refer \overset{TO-obj}{\longrightarrow} to)$ in *You refer to Smith* and $(to \overset{pre-TO-obj}{\longleftarrow} refer)$ in *The person to whom you must refer is Smith*).

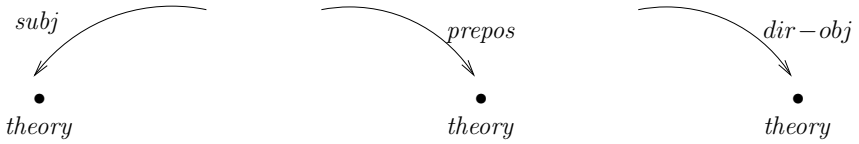
More detailed definitions of dependencies may be very complex (see [MP87]). But even these simplified examples explain why there are so many types of dependencies between verbs and their actants. They also explain the monomorphic nature of dependency types, in particular, inadequacy of dependency type raising. Essential distinctions are also in treating verb and noun modifiers, which in dependency surface syntax are *subordinate* and, in principle, *iterated*. This particularity is reflected by more recent DGs (cf. [ST93,LL96]). In contrast, the canonical CGs' elimination rules induce 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 $[n/n]$, would govern the modified nouns and not vice versa as in DGs. This distinction led to many propositions, both in terms of order constraints (e.g., [Mar90,Br98,DD01]) and in structure sharing terms like lifting (e.g., [LL98,KNR98]). In type logical grammars, a proper account of these distinctions is taken by multi-modal extensions of CGs (cf. [MM,Mor94,Kru01]), which explicitly distinguish the *semantic functionality* and the *syntactic subordinacy* (opposite, for instance, for verb and noun modifiers).

The overwhelming majority of DGs are head driven in the above sense. At the same time, some linguistic theories, e.g. "Meaning-Text Theory" [Mel97], "Word Grammar" [Hud84] are word driven. Formal definitions of word driven DGs are few in number. Historically the first was [GM71]. It proposed a concept of tree generating DG and started a mathematical research into subfam-

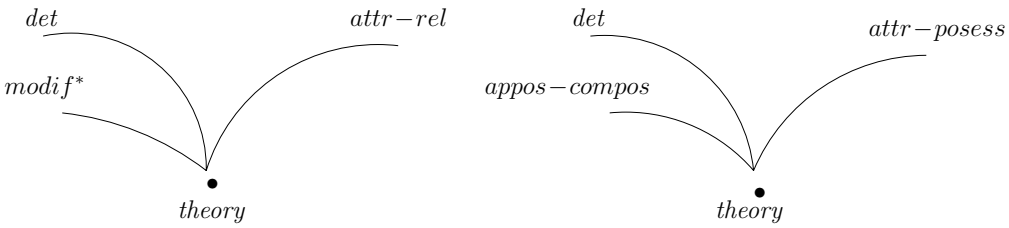
ilies of this very general class of grammars (see a review in [DM00]). More recent proposals are Link Grammars [ST93] expressing only projective DTs and Polarized DGs [Dik01a]. The existing word driven DG definitions are operational and make no link with logic. In particular, there is no definition based on word driven dependency types. In this paper, we propose such a definition in classical CGs terms.

2 Syntactic types

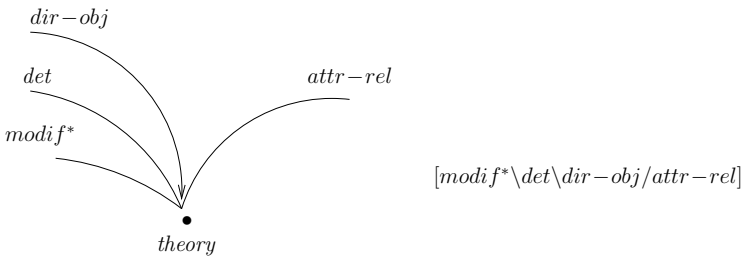
Suppose that we have to describe syntactic type of the word *theory* in terms of word driven dependencies. Then we are to look at this word from two different points of view. For *theory* as *dependent*, we must find all possible incoming dependencies:



For *theory* as *governor*, we must find all possible dependencies outgoing from it:



In so doing, we must of course take into account the precedence of beginnings (ends) of the arrows with respect to the word. This can be done using the classical CG type constructors. The fundamental difference compared with CG is that now a primitive type D corresponds to the *incoming* dependency D and in complex types, e.g. $D\backslash\alpha$ assigned to a word w, D corresponds to *the beginning* of dependency D. Finally, we must find all mutually compatible combinations of outgoing arrows and of an incoming arrow compatible with the features of the word *theory*. Each such combination describes a dependency type:



In this example, the type $[modif^*\backslash det\backslash dir{-}obj/attr{-}rel]$ assigned to *theory* ad-

mits several (eventually none) left dependents of *theory* through dependency $modif$ ($modif^*$ denotes iteration), requires a left dependent through *det*, a right dependent through $attr-rel$ and the incoming dependency $dir-obj$. As for the classical CG, the position of the ends of the outgoing dependencies (e.g. of $attr-rel$) must be determined through a proof. As we will see, such types are sufficient do describe projective dependencies (we call these dependencies *local*).

In this paper, we follow the proposal in [Dik01a,Dik04] and specify long distance discontinuous dependencies by polarized dependency types, which we call *valencies*. A *positive valency* specifies the name and the direction of an outgoing discontinuous 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 valencies are *dual*). So the long distance discontinuous dependencies are specified by pairs of dual valencies. For instance, the first member of the french discontinuous negation *ne .. pas* has the type $[neg/(\nearrow n-compound)]$, in which the left positive valency $(\nearrow n-compound)$ requires that the dependency $n-compound$ of *ne* should be outgoing from left to right. Respectively, the second member has type $(\searrow n-compound)$. Its right negative valency $(\searrow n-compound)$ requires that the dependency $n-compound$ of *pas* should be incoming from left to right. Together they define the long distance dependency $n-compound$.

It is not difficult to see that such valencies do not suffice to express the requirements that the end of a long distance dependency must *precede* or be *adjacent* to a word with a certain type. 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, we will apply to these valencies the adjacency modalities: $\#$ and $\flat$. Assigning to *It* the type $\#(\swarrow it-cleft)$ we require that the 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*, we assign to *was* the type $[\flat(\swarrow it-cleft)\backslash S/subj/circ]$. This type requires that the end of the long distance dependency $it-cleft$ must immediately precede *was* (i.e. must be *anchored* to it on the 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. Now we set to definition of types.

We address to syntactic types as *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, whereas *inf-subj* is that, in which the subordinate is a verb in infinitive. 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 $X \subseteq \mathbf{C}$, $X^* = \{C^* \mid C \in X\}$ and $X^\omega = X \cup X^*$. The elementary and iterated categories are *local*.

In order to define polarized valencies, we introduce four *polarities*: left and right positive: $\nearrow, \nwarrow$ (*outgoing from* left (respectively, 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, the *right positive* valency $(\nwarrow pre{-}UPON{-}obj)$ requires the beginning of the distant dependency $pre{-}UPON{-}obj$ of a transitive verb governing an oblique object dislocated from right to left and headed by the preposition 'UPON'. The end of this dependency will be required by the category of $UPON$ through the dual negative valency $(\swarrow pre{-}UPON{-}obj)$ (cf. *upon what dependency theory we rely*).

$\nearrow\mathbf{C}, \nwarrow\mathbf{C}, \swarrow\mathbf{C}$ and $\searrow\mathbf{C}$ denote the corresponding sets of polarized valencies. For instance, $\nearrow\mathbf{C} = \{(\nearrow C) \mid C \in \mathbf{C}\}$ is the set of *left positive* valencies. $V^+(\mathbf{C}) = \nearrow\mathbf{C} \cup \nwarrow\mathbf{C}$ is the set of positive valencies, $V^-(\mathbf{C}) = \searrow\mathbf{C} \cup \swarrow\mathbf{C}$ is the set of those negative. Intuitively, the positive valencies require the beginnings of long distance dependencies, whereas the negative valencies require their ends. The negative valencies in $V^-(\mathbf{C})$ do not constrain the position of the end of the required long distance dependency. So they are called *loose*.

In order to specify the positions of the ends of long distance dependencies, we use two modalities: # (*anchor*) and $\flat$ (*host*). Respectively, for each negative valency $vC \in V^-(\mathbf{C})$, the expressions $\#(vC)$ and $\flat(vC)$ are the corresponding *anchor* and *host valencies*. We will distinguish *left-argument* and *right-argument* host valencies and the corresponding *left* and *right positioned* anchor valencies:

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

and suppose that the sets $Host^l(\mathbf{C}), Host^r(\mathbf{C}), Anc^l(\mathbf{C})$ and $Anc^l(\mathbf{C})$ are disjoint.

Definition 1 *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 $C \in Cat(\mathbf{C})$, $A_1 \in (\mathbf{C} \cup \mathbf{C}^* \cup Host^l(\mathbf{C}) \cup \nwarrow\mathbf{C})$ *and* $A_2 \in (\mathbf{C} \cup \mathbf{C}^* \cup Host^r(\mathbf{C}) \cup \nearrow\mathbf{C})$, *the categories* $[A_1\backslash C]$ *and* $[C/A_2]$ *also belong to* $Cat(\mathbf{C})$.

$CCat(\mathbf{C}) \subset Cat(\mathbf{C})$ *denotes the set of all categories, which do not have subcategories in* $V^-(\mathbf{C}) \cup V^+(\mathbf{C})$. *We call them* continuous.

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 Dependency Grammar and Dependency Calculus

Definition 2 *A* categorial dependency grammar (CDG) *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* $a \in W$.

Below, we will index categories by their positions in a string of categories related by G with a given sentence $w = a_1 \dots a_n$. The indices serve to define dependency structures: α^i will be a category of a dependency structure with the root position a_i (a *positioned* category).

Definition 3 *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.

CDG derivations are proofs in the following dependency calculus.

Definition 4 *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* $\beta \in Host^l(\mathbf{C}) \cup \nwarrow \mathbf{C}$ *is a 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:
$\mathbf{C^l}$. $((V,E),\Gamma_1\ C^i\alpha^j\ \Gamma_2) \vdash ((V,E),\Gamma_1\ \alpha^jC^i\ \Gamma_2)$ *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 $CCat(\mathbf{C})$ *and to the first three rules* $\mathbf{L}, \mathbf{I}$ *and* $\mathbf{\Omega}$. *We call this restricted calculus* projective. *For purely technical reasons, we will admit in the projective calculus the anchor and host subcategories and add the following elimination rule* [3] *:*

$\mathbf{E^l}$. $((V,E),\Gamma_1\#^l(\alpha)^i[\flat^l(\alpha)\backslash\beta]^j\Gamma_2) \vdash ((V,E),\Gamma_1\beta^j\Gamma_2)$ *for* $\alpha \in V^-(\mathbf{C})$.

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 $\mathbf{L}$ is a direct analogue of the classical elimination rule. Rules $\mathbf{I}$ and $\mathbf{\Omega}$ extend $\mathbf{L}$ to the iterated categories. Particular are the rules of polarized valencies' control. Rule $\mathbf{V}$ extracts positive and host valencies from complex categories. Rule $\mathbf{C}$ moves the valencies in the indicated directions towards the *first available* valency, to which one can apply rules $\mathbf{A}$ or $\mathbf{D}$. Rule $\mathbf{D}$ adds a dependency C, when two loose dual valencies with the same name C become adjacent. Dependencies introduced by $\mathbf{D}$ will be called *long distance*. Rule $\mathbf{A}$ 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 $\mathbf{D}$.

Definition 5 *CDG with the sub-commutative (projective) provability relation will be called* sub-commutative *(respectively* projective*). A dependency struc-*

[3] In fact, this rule is needed only to prove the correctness of the parsing algorithm below.

*ture (*DS*) D is assigned by a CDG $G = (W, \mathbf{C}, S, \delta)$ 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 dependency structures $D(G) =_{df} \{D \mid \exists w\ G(D, w)\}$. The* language *generated by G is the set of sentences $L(G) =_{df} \{w \mid \exists D\ G(D, w)\}$. $\mathcal{D}(CDG^{sc})$ and $\mathcal{L}(CDG^{sc})$ will denote the families of D-languages and languages generated by grammars in CDG^{sc}.*

4 Expressive power

The following example cited from [Dik04] shows that sub-commutative CDGs are more expressive than CFGs.

Example 1 *Let $G_0 = (\{a, b, c, d_1, d_2, d_3\}, \mathbf{C}_0, S, \delta_0)$, where δ_0 is defined by:*
$a \mapsto [\natural^l(\swarrow A)\backslash\#^l(\swarrow B)],\ [\natural^l(\swarrow B)\backslash\#^l(\swarrow B)]$, $\quad d_1 \mapsto \#^l(\swarrow A)$,
$b \mapsto [\nwarrow B\backslash D/C]$, $\quad d_2 \mapsto [\natural^l(\swarrow B)\backslash \nwarrow A\backslash S/D]$,
$c \mapsto [D\backslash C]$, $\quad d_3 \mapsto D$,
Fig. 1 shows a proof of $G_0(D^{(3)}, d_1a^3d_2b^3d_3c^3)$, which defines the DS $D^{(3)}$ on $d_1a^3d_2b^3d_3c^3$. In this proof, $\alpha =_{df} \#^l(\swarrow B)$, $\alpha_1 = \#^l(\swarrow A)$, $\beta = \natural^l(\swarrow B)$, $\beta_1 = \natural^l(\swarrow A)$, $\gamma = (\nwarrow B)$, and $\gamma_1 = (\nwarrow A)$.

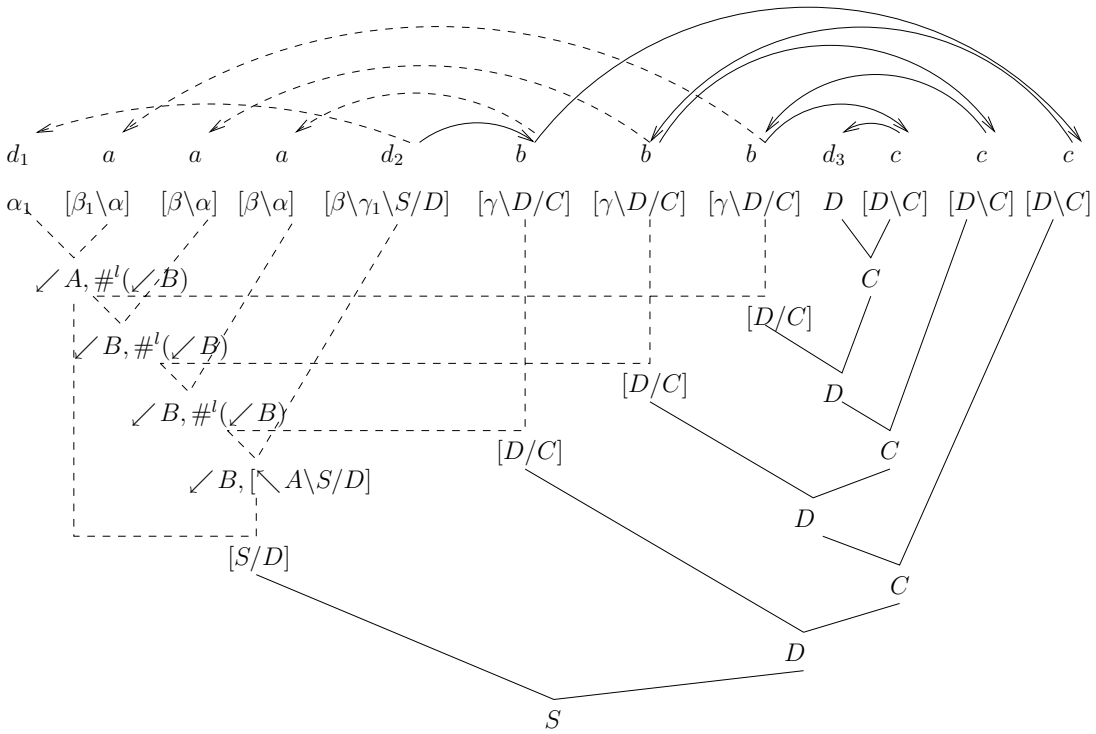


Fig. 1. A proof of $G_0(D^{(3)}, d_1a^3d_2b^3d_3c^3)$.

In Fig. 1, two meeting solid slanting lines correspond to one application of the local or iterated dependency rule, two meeting dashed slanting lines correspond to one application of the anchored dependency rule, and right-angled dashed lines connect categories to which the rules $\mathbf{V^l}, \mathbf{C^l}, \mathbf{C^r}, \mathbf{D^l}$ are applied.

Proposition 1 $L(G_0) = \{d_1 a^n d_2 b^n d_3 c^n \mid n > 0\}$.

Proof. If we delete all polarized subcategories, we obtain a rigid [4] categorial grammar defining the language $\{d_2 b^n d_3 c^n \mid n \geq 1\}$. In particular, this shows that in the original grammar G_0, b cannot have occurrences on the right of d_3. Returning to G_0, we remark that only one occurrence of d_2 is possible (its principal subcategory is S). From this follows that there is a single occurrence of d_1. Due to the orientation of valencies $\#^l(\swarrow A), \nwarrow A$ in $\delta(d_1)$ and $\delta(d_2)$, d_1 must precede d_2. If some a precedes d_1 or follows d_2, then its category evidently cannot be eliminated. Only the first a can have category $[\flat^l(\swarrow A)\backslash\#^l(\swarrow B)]$, otherwise the category or subcategory $\flat^l(\swarrow B)$ will separate $\#^l(\swarrow A)$ and $\flat^l(\swarrow A)$ and $\swarrow A$ will never be eliminated. Similarly, there cannot be an occurrence of b preceding a. Otherwise, the subcategory $\#^l(\swarrow B)$ of the category $[\flat^l(\swarrow B)\backslash\#^l(\swarrow B)] \in \delta(b)$ will never be eliminated. Thus, all a precede all b and so there must be as many a as b. □

Example 2
Fig. 2 shows a case of topicalization in German using the categories:
$C_{das} =_{df} det$,
$C_{Buch} =_{df} [det\backslash\#^l(\swarrow pre{-}dobj)]$,
$C_{hat} =_{df} [\flat^l(\swarrow pre{-}dobj)\backslash S/aux/subj/\flat^r(\swarrow pre{-}iobj)]$,
$C_{mir} =_{df} \#^r(\swarrow pre{-}iobj)$,
$C_{Peter} =_{df} subj$,
$C_{versprochen} =_{df} [\nwarrow pre{-}iobj\backslash aux/inf{-}dobj]$,
$C_{zu} =_{df} [inf{-}dobj/zu{-}inf]$,
$C_{lesen} =_{df} [\nwarrow pre{-}dobj\backslash zu{-}inf]$.

In the proof in Fig. 2, rule $\mathbf{L^l}$ applied to C_{das}, C_{Buch} gives $\#^l(\swarrow pre{-}dobj)$. $\swarrow pre{-}dobj, [S/aux/subj/\flat^r(\swarrow pre{-}iobj)]$ results from this category and C_{hat} by applying rules $\mathbf{V^l}$ and $\mathbf{A^l}$. Next, $\swarrow pre{-}dobj$ is moved to the right by several applications of $\mathbf{C^r}$ till it reaches C_{lesen}. Then $\nwarrow pre{-}dobj$ is eliminated in C_{lesen} by $\mathbf{V^l}$ and $\mathbf{D^l}$, and the corresponding discontinuous dependency ($Buch \overset{pre-dobj}{\longleftarrow} lesen$) is introduced. Rule $\mathbf{L^r}$ applied to C_{zu} and $zu{-}inf$ gives $inf{-}dobj$. Similarly, rules $\mathbf{V^r}$ and $\mathbf{A^r}$ give $[S/aux/subj], \swarrow pre{-}iobj$ being applied to $[S/aux/subj/\flat^r(\swarrow pre{-}iobj)]$ and C_{mir}. Next, $\swarrow pre{-}iobj$ is moved to the right by several applications of $\mathbf{C^r}$ till it reaches $C_{versprochen}$. Then by $\mathbf{V^l}$ and $\mathbf{D^l}$, $\nwarrow pre{-}iobj$ is eliminated in $C_{versprochen}$ and

[4] I.e. assigning a single type per word.

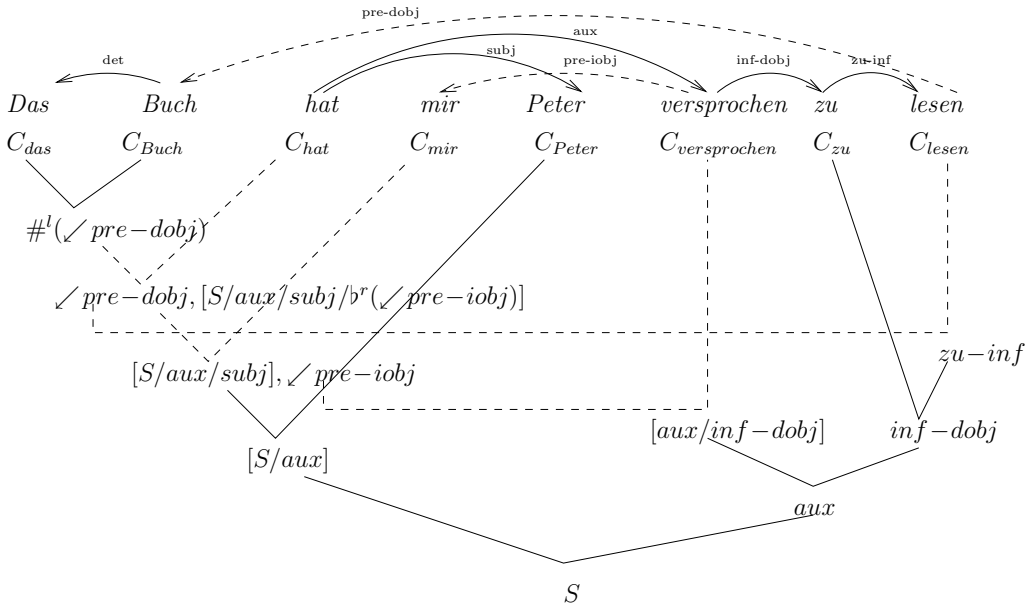


Fig. 2. A case of topicalization in German.

the corresponding discontinuous dependency ($mir \overset{pre-iobj}{\longleftarrow} versprochen$) is introduced. It remains to apply three times rule $\mathbf{L^r}$ to obtain S.

Definition 6 *Let D be a DS of a sentence $w = a_1 \dots a_n$. For a space i between the words a_i and a_{i+1}, $1 \leq i < n$, we define the* long distance dependencies thickness *in i (denoted $dth(D, i)$) as the number of long distance dependencies $(a_k \overset{d}{\longleftarrow} a_l)$, $(a_k \overset{d}{\longrightarrow} a_l)$ in D covering i (i.e. such that $k < i < l$ for some k, l and d).*

$$dth(D) =_{df} max\{dth(D, i) \mid 1 \leq i < |D|\},$$
$$dth(G) =_{df} max\{0, dth(D) \mid D \in D(G)\} \text{ and}$$
$$dth_G(n) =_{df} max\{0, dth(D) \mid G(D, w),\ |w| \leq n\}.$$

For instance, $dth(G_0) = \infty$. For natural languages, this measure is seemingly bounded by a small constant (2 or 3). In example 2, $dth(D) = 2$.

Theorem 1 *If for a sub-commutative CDG G, the measure $dth(G)$ is bounded by a constant, then $L(G)$ is context-free.*

Proof. To prove this theorem, we use the polarized dependency tree grammars (*PDTGs*) of [Dik01b]. For dependency trees generated by PDTGs, a discontinuity complexity measure is defined called *defect* and denoted $\sigma(G)$. The main result of [Dik01b] is that PDTGs with the defect bounded by a constant generate cf-languages. As it can be seen from [Dik01b], for each sub-commutative CDG G, a strongly equivalent PDTG G_1 can be trivially constructed such that $\sigma(G_1) \leq dth(G)$. □

It is an interesting theoretical problem to compare the weak generative capacity of sub-commutative CDGs and that of mildly context-sensitive grammars [JSW91]. We conjecture that the copy language $\{wcw \mid w \in W^*\}$ cannot be generated by sub-commutative CDGs . On the other hand, the following proposition shows that if it is true, then sub-commutative CDG -languages are incomparable with basic TAG languages.

Proposition 2 *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 sub-commutative CDG .*

Proof. An argument similar to that in Proposition 1 shows that $L^{(m)}$ is generated by the following sub-commutative CDG :
$d_0 \mapsto [S/D_0]$ and $d_{m+1} \mapsto D_m$,
$a_0 \mapsto [D_0/D_0/(\nearrow A_m)/\dots/(\nearrow A_1)]$,
for $0 < i \leq m$, $d_i \mapsto [D_{i-1}/\flat^r(\searrow A_i)]$,
and $a_i \mapsto [\#^r(\searrow A_i)/\flat^r(\searrow A_i)], [\#^r(\searrow A_i)/D_i]$. $\square$

As it is well known, the languages $L^{(m)}, m > 4$, are not generated by basic TAGs.

Remark 1 *In contrast with the CDGs of [Dik04], which generate only dependency trees, the dependency structures generated by sub-commutative CDG may have cycles, as shows the following example:*
$[(\swarrow A)/(\nearrow B)]^1[(\nwarrow A)\backslash(\searrow B)]^2 S^3 \vdash (\swarrow A)^1(\nearrow B)^1[(\nwarrow A)\backslash(\searrow B)]^2 S^3 \vdash$
$(\swarrow A)^1(\nearrow B)^1(\nwarrow A)^2(\searrow B)^2 S^3 \vdash (\swarrow A)^1(\nwarrow A)^2(\nearrow B)^1(\searrow B)^2 S^3 \vdash^* S^3$.
In our opinion, it is not very important. Nevertheless, a simple sufficient condition of acyclicity of sub-commutative CDGs may be formulated in terms of a well-founded order on dependency types.

5 Complexity

If there is no uniform bound on the number of elementary categories, then parsing of sub-commutative CDGs is a hard problem.

Theorem 2 *The problem* $G(D, w)$ *is NP-complete.*

Proof. The NP-hardness can be proven by the following polynomial reduction of $3-CNF$. Let $\Phi = C_1 \wedge \dots \wedge C_m$ be a CNF with clauses C_j including three literals l_1^j, l_2^j, l_3^j and $l_k^j \in \{x_1, \neg x_1, \dots, x_n, \neg x_n\}$.
We define from Φ the sub-commutative CDG $G(\Phi) = (W, \mathbf{C}, S, \delta)$, in which $W = \{\Phi, C_1, \dots, C_m, x_1, \dots, x_n, y_1, \dots y_n\}$, $\mathbf{C} = \{S, A, 1_0, 1_1, 2_0, 2_1, \dots, n_0, n_1\}$ 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 $w(\Phi) = x_1 x_2 \dots x_n \Phi C_1 C_2 \dots C_m y_1 y_2 \dots y_n$.

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. □

Fortunately, this anomaly is impossible in practice, because the inventory of elementary categories is finite and universal. All the more so, it is finite for dependency grammars of particular languages. We show that under this natural condition sub-commutative CDG parsing is polynomial time.

It turns out that to parse sub-commutative CDGs , it suffice to perform two *independent* tests: the first in terms of the projective provability $\vdash_p$ and the second in terms of neutralizability of long distance dependency valencies. To formulate this fact, we need two different projections of categories. The first, called *local*, preserves only elementary and host argument sub-categories. Intuitively, it preserves only projective dependencies of words and also their neighborhood with anchored words. The second projection, called *valency projection*, preserves only polarized valencies and their respective order.

Definition 7 Local projection $\|\gamma\|_l$ *of a string* $\gamma \in Cat(\mathbf{C})^*$ *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* $a \in V^+(\mathbf{C})$.

Example 3 *According to these definitions,*
$\|[\mathfrak{b}^l(\searrow c)\backslash(\nwarrow a)\backslash b\backslash\#^r(\swarrow d)]\|_l\ =\ [\mathfrak{b}^l(\searrow c)\backslash b\backslash\#^r(\swarrow d)]$,
$\|[\mathfrak{b}^l(\searrow c)\backslash(\nwarrow a)\backslash b\backslash(\swarrow d)/e]\|_l\ = [\mathfrak{b}^l(\searrow c)\backslash b\backslash\varepsilon/e]$,
$\|[\mathfrak{b}^l(\searrow c)\backslash(\nwarrow a)\backslash b\backslash d]\|_v\ = \nwarrow a$, *and*
$\|[\mathfrak{b}^l(\searrow c)\backslash(\nwarrow a)\backslash b\backslash\#^l(\swarrow d)/e]\|_v\ = \nwarrow a \swarrow d$.

Besides these projections, we need a "well-bracketing" criterion of polarized valencies. In this bracketing, $\swarrow d$ and $\nearrow d$ play the role of left brackets and $\nwarrow d$ and $\searrow d$ serve as the corresponding right brackets. Obviously, for a left valency α, the corresponding right valency is $\breve{\alpha}$. Pairs $(\alpha, \breve{\alpha})$ are called *correct*.

Definition 8 *Let $G = (W, \mathbf{C}, S, \delta)$ be a sub-commutative CDG , $(\alpha, \breve{\alpha})$ be a correct pair and $\gamma \in Cat(\mathbf{C})^+$ be a string of categories. For both dependency valencies β in $(\alpha, \breve{\alpha})$, $|\gamma|_\beta$ will denote the number of occurrences of β in the valency projection $\|\gamma\|_v$. The values*

$$\Delta^L_\alpha(\gamma) = max\{|\gamma'|_{\breve{\alpha}} - |\gamma'|_\alpha \mid \gamma' \textit{ is a } \mathsf{prefix} \textit{ of } \gamma\},$$
$$\Delta^R_\alpha(\gamma) = max\{|\gamma'|_\alpha - |\gamma'|_{\breve{\alpha}} \mid \gamma' \textit{ is a } \mathsf{suffix} \textit{ of } \gamma\}$$

express respectively the number of right and left non-neutralized dependency valencies α (i.e. the maximal deficit of left and right $\alpha-$parentheses) in γ [5] *.*

Let $\gamma, \gamma_1, \gamma_2 \in Cat(\mathbf{C})^+$ be some strings of categories and $(\alpha, \breve{\alpha})$ be a correct valency pair.
1. If $|\gamma|_\alpha = |\gamma|_{\breve{\alpha}} = 0$, then the pair $(\alpha, \breve{\alpha})$ is neutralized *in γ.*
2. If $(\alpha, \breve{\alpha})$ is neutralized *in γ, γ_1, and γ_2, then it is also neutralized in $\gamma_1\alpha\gamma\breve{\alpha}\gamma_2$.*

Definition 9 *A string γ of polarized valencies is* well balanced *if each projection of γ onto $\{\alpha, \breve{\alpha}\}$, $\alpha \in \nearrow \mathbf{C} \cup \swarrow \mathbf{C}$, is well-bracketed.*

Lemma 1 *1. A string of polarized valencies is well balanced iff each correct pair is neutralized in it.*
2. A string of polarized valencies is well balanced iff ε can be derived from it by a number of applications of rules $\mathbf{C^l}$, $\mathbf{C^r}$, $\mathbf{D^l}$ and $\mathbf{D^r}$.

Proof. Point 1 follows immediately from definitions 8,9.
Point 2 is proved by a straightforward induction (the necessity: on the string length, the sufficiency: on the number of rule applications). □

Finally, the use of iterative types leads to the following notion of realization.

Definition 10 *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.*

It is easy to notice that for any sub-commutative CDG G and any string x the following equivalence holds: $x \in L(G)$ iff there is a string of categories $\alpha \in \delta(x)$ and some its realization $\gamma(x) \in R(\alpha)$ such that $((x, \emptyset), \gamma(x)) \vdash^*$

[5] Having in mind that there is $\gamma' = \varepsilon$, the values $\Delta^L_\alpha(\gamma)$ and $\Delta^R_\alpha(\gamma)$ are non-negative.

(D, S^j) for some dependency structure D and $1 \leq j \leq |x|$. Here is the two-test membership criterion.

Theorem 3 *Let $G = (W, \mathbf{C}, S, \delta)$ be a sub-commutative CDG . $x \in L(G)$ 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. $\|\gamma\|_v$ is well balanced.

Proof. In this proof, we ignore the dependency structures and consider only the inference of categories.

$\Rightarrow$ Let $x \in L(G)$, $\alpha \in \delta(x)$ and $\gamma \in R(\alpha)$ be some its realization such that there exists a proof $((x, \emptyset), \gamma) \vdash \ldots \vdash (D^k, \gamma^k) \vdash (D^n, \gamma^n) = (D, S^t)$ for some $n \geq 0$, dependency structure D and $1 \leq t \leq |x|$. We will prove by induction on k that for each $0 \leq k \leq n$ the following two assertions hold:
(i) $\|\gamma\|_l \vdash_p^* \|\gamma^k\|_l$,
(ii) each correct pair $(\alpha, \breve{\alpha})$ is neutralized in $\|\gamma^k\|_v$ iff it is neutralized 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 $(D^k, \gamma^k) \vdash^R (D^{k+1}, \gamma^{k+1})$ (immediately derived by rule R).

If $R = \mathbf{L^1}$, then $\gamma^k = \Gamma_1 C^i [C \backslash \beta]^j \Gamma_2$ and $\gamma^{k+1} = \Gamma_1 \beta^j \Gamma_2$ for some $\Gamma_1, \Gamma_2, \beta, i$ and j. Passing to local projection, we obtain: $\|\gamma^k\|_l = \|\Gamma_1\|_l C^i [C \backslash \|\beta\|_l]^j \|\Gamma_2\|_l \vdash_p \|\Gamma_1\|_l \|\beta\|_l^j \|\Gamma_2\|_l = \|\gamma^{k+1}\|_l$ and $\|\gamma^k\|_v = \|\gamma^{k+1}\|_v$. Then $\|\gamma\|_l \vdash_p^* \|\gamma^k\|_l \vdash^{\mathbf{L^1}} \|\gamma^{k+1}\|_l$ and both conditions (i) and (ii) are satisfied for $k + 1$.

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

If $R = \mathbf{C^l}$, then $\gamma^k = \Gamma_1 \; C^j \alpha^i \; \Gamma_2$ and $\gamma^{k+1} = \Gamma_1 \; \alpha^i C^j \; \Gamma_2$ 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 neutralized in $\|\gamma^{k+1}\|_v$ iff it is neutralized in $\|\gamma^k\|_v$. 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 $\gamma^k = \Gamma_1 (\swarrow C)^i (\nwarrow C)^j \Gamma_2$ 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 $\|\gamma^k\|_v = \|\Gamma_1\|_v (\swarrow C)^i (\nwarrow C)^j \|\Gamma_2\|_v$ by cancelling the cor-

rect valency pair $(\swarrow C)^i(\nwarrow C)^j$. Therefore, this pair is neutralized in $\|\gamma^{k+1}\|_v$ iff it is neutralized 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 proven similarly. Iterative dependency rules $\mathbf{I^r}$ and $\mathbf{\Omega^l}$ and their "right" versions are never used in the proof because the initial string of categories γ does not include iterative categories. The rules $\mathbf{V^l}, \mathbf{V^r}$ 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.

$\Leftarrow$ Let us suppose that there is a string of categories $\alpha \in \delta(x)$ and some its realization $\gamma \in R(\alpha)$ such that:
1. $\|\gamma\|_l \vdash_p^* S$,
2. each correct pair $(\alpha, \breve{\alpha})$ is neutralized in $\|\gamma\|_v$.

We must prove that $x \in L(G)$. For that we show that $((x, \emptyset), \gamma) \vdash^* ((x, D), S)$.

We presume that $\|\gamma\|_l \vdash_p^n S$ and show the existence of a proof $((x, \emptyset), \gamma) \vdash^* ((x, D), S)$ by induction on n. As we ignore the dependency structures, somewhat abusing notation we will expose only the category component of this proof: $\gamma \vdash^* S$.

If $n = 0$, then $\gamma = \Gamma_1 S \Gamma_2$ for some strings Γ_1 and Γ_2 of polarized valencies, where $\Gamma_1\Gamma_2$ is well balanced. 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.

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

If $R = \mathbf{L^l}$, then $\|\gamma\|_l = \Gamma_1 C[C\backslash\beta]\Gamma_2$ and $\gamma' = \Gamma_1\beta\Gamma_2$ 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 string of polarized valencies (in $V^+(\mathbf{C}) \cup V^-(\mathbf{C})$). Now we can construct the proof $\gamma = \tilde{\Gamma}_1 C \Delta [C\backslash\tilde{\beta}]\tilde{\Gamma}_2 \vdash^* \tilde{\Gamma}_1 \Delta C [C\backslash\tilde{\beta}]\tilde{\Gamma}_2 \vdash^{\mathbf{L^l}} \tilde{\Gamma}_1\Delta\tilde{\beta}\tilde{\Gamma}_2 = \gamma'$, where in the first part of the proof only commutativity rules are used in order to move C on the right of Δ. Regarding the projections of γ', we can see that $\|\gamma'\|_l = \gamma'_l$ and $\|\gamma'\|_v = \|\gamma\|_v$. Therefore, $\|\gamma'\|_v$ is well 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 well balanced, $\|\tilde{\Gamma}_1\|_l = \Gamma_1$, $\|\tilde{\Gamma}_2\|_l = \Gamma_2$, $\|\tilde{\beta}\|_l = \beta$, and Δ is a string in $(V^+(\mathbf{C}) \cup V^-(\mathbf{C}))^*$.

Let us consider the proof:

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

in which in the part $\vdash^*$ only the commutativity rules are used in order to move $\flat^l(\alpha)$ on the left of 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 well balanced. Therefore, by induction the proof (1) can be extended by a proof $\tilde{\Gamma}_1\alpha\Delta\tilde{\beta}\tilde{\Gamma}_2 \vdash^* S$. □

This theorem enables efficient parsing algorithms for sub-commutative CDGs .

The algorithm **CdgAnalyst** we describe below is a typical dynamic programming style recognizer. When applied to a string $x = w_1...w_n$, **CdgAnalyst** incrementally fills a triangular matrix M of size $n \times n$, whose element $M[i,j], i \leq j$, is a finite set of so called "*items*" (see below). Given a category $C \in Cat$, let **-contraction*(C) denote the smallest set of categories such that: (i) $C \in$ **-contraction*(C) and (ii) if $[A^*\backslash\alpha] \in$ **-contraction*(C) or $[\alpha/A^*] \in$ **-contraction*(C), then $[\alpha] \in$ **-contraction*(C).

For a sub-commutative CDG $G = (W, \mathbf{C}, S, \delta)$, let $L = (\alpha_1, \ldots, \alpha_p)$ be the list of all positive and negative left dependency valencies in $\nearrow\mathbf{C} \cup \swarrow\mathbf{C}$ and $R = (\breve{\alpha}_1, \ldots, \breve{\alpha}_p)$ be the list of all corresponding negative and positive right dependency valencies in $\searrow\mathbf{C} \cup \nwarrow\mathbf{C}$. *Items* of **CdgAnalyst** have the form (I, lc, rc), where $lc = (\Delta^L_{\alpha_1}, \ldots, \Delta^L_{\alpha_p})$ and $rc = (\Delta^R_{\alpha_1}, \ldots, \Delta^R_{\alpha_p})$ are integer vectors, whose components i are respectively the left and right dependency valency deficits (see Definition 8) of the corresponding valencies $\alpha_i \in L$, and $\breve{\alpha}_i \in R$. $I = C^{(k)}$, where $C \in Cat \cup \{\varepsilon\}$, $1 \leq k \leq n$. Intuitively, $C^{(k)}$ corresponds to a DS of type C with the root position k.

CdgAnalyst uses procedures **PROPOSE**, **SUBORDINATE_L** and **SUBORDINATE_R**. The last two are similar, so we show one of them. (α_m) and $(\breve{\alpha}_m)$ in **PROPOSE**, are the corresponding members of the lists L and R.

For $1 \leq i \leq n$:
PROPOSE(i)
FORALL $\gamma \in \delta(w_i)$
DO
  FORALL $\alpha_j \in L$
  DO
    $(lc)_j = \Delta^L_{\alpha_j}(\gamma)$; $(rc)_j = \Delta^R_{\alpha_j}(\gamma)$
  END_FORALL;
  FORALL $C \in$ **-contraction*$(\|\gamma\|_l)$
  DO
    add $(C^{(i)}, lc, rc)$ **to** $M[i,i]$
  END_FORALL
END_FORALL

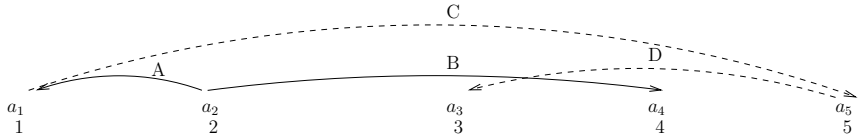
For $1 \leq i \leq j < k \leq n$:
SUBORDINATE_L(i, j, k)
FORALL $(C_1^{(u)}, lc_1, rc_1) \in M[i, j]$, **and**
$(C_2^{(v)}, lc_2, rc_2) \in M[j+1, k]$
DO
FORALL m **incrementally in** $1 \leq m \leq p$
DO
$(lc)_m = (lc_1)_m + max\{0, (lc_2)_m - (rc_1)_m\}$;
$(rc)_m = (rc_2)_m + max\{0, (rc_1)_m - (lc_2)_m\}$
END_FORALL;
IF $(C_2 = [C_1 \backslash \gamma])$ **OR** $(C_1 = \#^l(\alpha)$ **AND** $C_2 = [\flat^l(\alpha) \backslash \gamma])$
THEN
FORALL $C \in$ **-contraction*(γ)
DO
add $(C^{(v)}, lc, rc)$ **to** $M[i, k]$
END_FORALL;
ELSE_IF $C_2 = [C_1^* \backslash \gamma]$ **OR** $C_1 = \varepsilon$
THEN
FORALL $C \in$ **-contraction*(C_2)
DO
add $(C^{(v)}, lc, rc)$ **to** $M[i, k]$
END_FORALL;
END_IF
END_FORALL

Algorithm CdgAnalyst
Input: $G \in \mathcal{L}(CDG^{sc})$ and $x = w_1 ... w_n$
Output: "YES" iff $x \in L(G)$.

FORALL m **incrementally in** $1 \leq m \leq n$
DO
PROPOSE(m)
END_FORALL;
FORALL l **incrementally in** $1 \leq l \leq n-1$
DO
FORALL i **incrementally in** $1 \leq i \leq n-l$
DO
$k = i + l$;
FORALL j **incrementally in** $i \leq j < k$
DO
SUBORDINATE_L(i, j, k);
SUBORDINATE_R(i, j, k)
END_FORALL
END_FORALL

END_FORALL;
IF $(S,(0,0,\ldots,0),(0,0,\ldots,0)) \in M[1,n]$
THEN Output("YES")
ELSE Output("NO")
END_IF

Example 4 *Consider the sub-commutative CDG* $G = (W, \mathbf{C}, S, \delta)$, *where* $W = \{a_1, a_2, a_3, a_4, a_5\}, \mathbf{C} = \{A, B, C, D\}$, $\delta(a_1) = \{[A/ \nearrow C]\}$, $\delta(a_2) = \{[A^*\backslash S/B/\flat^r(\swarrow D)]\}$, $\delta(a_3) = \{\#^r(\swarrow D)\}$, $\delta(a_4) = \{B\}$, $\delta(a_5) = \{[\nwarrow D\backslash \searrow C]\}$. *Let* $w = a_1a_2a_3a_4a_5$. *Here is the matrix* $M[i,k]$ *calculated by* **CdgAnalyst**(G,w) *and the dependency tree assigned to* w *in this computation.*



1	$(A^{(1)},(0,0),(1,0))$	$([S/B/\flat^r(\swarrow D)]^{(2)},(0,0),(1,0))$ $([A^*\backslash S/B/\flat^r(\swarrow D)]^{(2)},(0,0),(1,0))$	$([S/B]^{(2)},(0,0),(1,1))$ $([A^*\backslash S/B]^{(2)},(0,0),(1,1))$	$(S^{(2)},(0,0),(1,1))$ $([A^*\backslash S]^{(2)},(0,0),(1,1))$	$(S^{(2)},(0,0),(0,0))$
2		$([A^*\backslash S/B/\flat^r(\swarrow D)]^{(2)},(0,0),(0,0))$ $([S/B/\flat^r(\swarrow D)]^{(2)},(0,0),(0,0))$	$([A^*\backslash S/B]^{(2)},(0,0),(0,1))$ $([S/B]^{(2)},(0,0),(0,1))$	$([A^*\backslash S]^{(2)},(0,0),(1,0))$ $(S^{(2)},(0,0),(1,0))$	$([A^*\backslash S]^{(2)},(1,0),(0,0))$ $(S^{(2)},(1,0),(0,0))$
3			$(\#^r(\swarrow D)^{(3)},(0,0),(0,1))$	$\emptyset$	$\emptyset$
4				$(B^{(4)},(0,0),(0,0))$	$(B^{(4)},(1,1),(0,0))$
5					$(\varepsilon^{(5)},(1,1),(0,0))$

Theorem 4 **CdgAnalyst**(G,x) *outputs "YES" iff* $x \in L(G)$.

Proof. The proof is based on Theorem 3 and the following lemma:

Lemma 2 *For all* $1 \le i \le k \le n$, *the item* $(C^{(v)}, lc, rc)$ *is added to* $M[i,k]$ *iff there exists a sequence of categories* $\gamma \in \delta(a_i \ldots a_k)$ *such that the following two conditions hold:*
(i) $\|\gamma\|_l \vdash_p^* C$,
(ii) $(lc)_m = \Delta^L_{\alpha_m}(\|\gamma\|_v)$ *and* $(rc)_m = \Delta^R_{\alpha_m}(\|\gamma\|_v)$ *for all* $m = 1, \ldots, p$.

Sketch of the proof (by induction on the length $l = k - i + 1$ of $a_i \ldots a_k$).

For $l = 1$, the assertion of the lemma follows immediately from the definition of procedure **PROPOSE**(i) for each i.

Let $l = k - i + 1 > 1$ and $(C^{(v)}, lc, rc) \in M[i,k]$. Then this item was added to $M[i,k]$ by **SUBORDINATE_L**(i,j,k) or by **SUBORDINATE_R**(i,j,k) for some $j \in [i,k]$. Let us suppose that it was added by **SUBORDINATE_L**(i, j, k) using two items: $(C_1^{(u)}, lc_1, rc_1) \in M[i,j]$, $(C_2^{(v)}, lc_2, rc_2) \in M[j+1,k]$. Since $j - i + 1 < l$ and $k - j < l$, there exist two sequences of categories $\gamma_1 \in \delta(a_i \ldots a_j)$ and $\gamma_2 \in \delta(a_{j+1} \ldots a_k)$, for which conditions (i) and (ii) hold.

Let $\gamma = \gamma_1\gamma_2$. Then $\gamma \in \delta(a_i \dots a_k)$ and by definition of local projection and from the choice of C, C_1 and C_2 it follows that $\|\gamma\|_l = \|\gamma_1\|_l\|\gamma_2\|_l \vdash_p^* C_1C_2 \vdash_p^* C$. It is easy to see that the definitions of $(lc)_m$ and $(rc)_m$ in **SUBORDINATE_L**(i, j, k) ensure condition (ii) of the lemma for $\|\gamma\|_v$ and all $m = 1, \dots, p$.

Now let us consider some sequence $\gamma \in \delta(a_i \dots a_k)$ and a category $C \in CCat(\mathbf{C})$ such that $\|\gamma\|_l \vdash_p^* C$. Suppose that the last rule in this derivation is one of the left rules $\mathbf{L^l}$, $\mathbf{I^l}$, $\mathbf{\Omega^l}$, $\mathbf{E^l}$. Then there are two categories C_1 and C_2 and a position $j \in [i, k]$ such that $\gamma = \gamma_1\gamma_2$, $\gamma_1 \in \delta(a_i \dots a_j)$, $\gamma_2 \in \delta(a_{j+1} \dots a_k)$, $\|\gamma_1\|_l \vdash_p^* C_1$, and $\|\gamma_2\|_l \vdash_p^* C_2$. By the induction assumption, an item $(C_1^{(u)}, lc_1, rc_1)$ belongs to $M[i, j]$ and an item $(C_2^{(v)}, lc_2, rc_2)$ belongs to $M[j+1, k]$ for some $u \in [i, j]$, $v \in [j+1, k]$ and some lc_1, rc_1, lc_2, rc_2 satisfying (ii). Then, whatever be the last rule, the item $(C^{(v)}, lc, rc)$ will be added to $M[i, k]$ by the call of **SUBORDINATE_L**(i, j, k). □

A straightforward calculation shows the following complexity of our algorithm.

Theorem 5 *Let p be the number of all polarized valencies. Then:*
(1) **CdgAnalyst** *has time complexity* $\mathbf{O}(n^5 dth_G(n)^{4p})$ *(clearly, $dth_G(n) < n$).*
(2) For any constant bound on p and $dth(G)$, **CdgAnalyst** *has time complexity* $\mathbf{O}(n^3)$.

Remark 2
1. As we have remarked above, constant bounds on p and $dth(G)$ exist in practice. (2) holds because firstly, due to the constant bounds the set of deficit vectors is finite and secondly, when the DSs are not constructed, there is no need to include the root positions into the items.
2. Using the standard chart techniques, the recognizer **CdgAnalyst** *can be easily transformed into a parser. In particular, this needs keeping the root positions. So in the conditions of point (2), the parser has complexity* $\mathbf{O}(n^5)$.
3. Clearly, if sub-commutative CDG G is projective, *then $dth(G) = 0$ and the deficit vectors are not needed.*

Corollary 1 *For any constant bound on p :*
(1) projective sub-commutative CDGs are parsed in time $\mathbf{O}(n^3)$*;*
(2) if $dth_G(n) = \mathbf{O}(\log n)$, then **CdgAnalyst** *has time complexity* $\mathbf{O}(n^{5+\varepsilon})$.

6 Acknowledgments

The authors are grateful to Denis Béchet and Annie Foret for helpful comments and fruitful discussions of this work.

7 Concluding remarks

The sub-commutative CDGs introduced in this paper combine the type logical style fitting well the standard methods of constructing formal semantics with the traditional valency/polarity style peculiar to all dependency grammars. The abstract theoretical version of sub-commutative CDGs in this paper can be easily adapted to practical definitions of surface dependency syntax of natural languages. For this, the elementary types should be parametrized by *non-recursive* feature structures and feature unification and propagation through dependencies must be allowed. The use of anchor/host modalities on top of order sensitive dependency valencies makes possible to express a variety of linear order constraints seemingly sufficient for the needs of surface dependency syntax formalization. At the same time, the sub-commutative CDGs are realistic because they have an efficient deterministic parsing algorithm, which will work in fact in cubic time for practical grammars taking into account discontinuous long distance dependencies.

Some of important properties of sub-commutative CDGs are already established. For example, they turn to be learnable from positive data (see [BDFM04]). At the same time, a detailed study of their expressiveness is needed, in particular, a comparison with mild context-sensitive grammars, pregroups and multi-modal Lambek calculus.

References

[BDFM04] D. Bechet, A. Dikovsky, A. Foret, and E. Moreau. On learning discontinuous dependencies from positive data. In G. Jäger, P. Monachesi, G. Penn, and S. Winter, editors, *Proc. of the 9th Intern. Conf. "Formal Grammar 2004" (FG 2004)*, pages 1–16, Nancy, France, August 2004. http://cs.haifa.ac.il/~shuly/fg04/.

[BHGS60] Y. Bar-Hillel, H. Gaifman, and E. Shamir. On categorial and phrase structure grammars. *Bull. Res. Council Israel*, 9F:1–16, 1960.

[Br98] Norbert Bröker. Separating surface order and syntactic relations in a dependency grammar. In *Proc. COLING-ACL*, pages 174–180, Montreal, 1998.

[DD01] Denis Duchier and Ralf 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.

[Dik01a] Alexander Dikovsky. Grammars for local and long dependencies. In *Proc. of the 39th Intern. Conf. ACL'2001*, pages 156–163. ACL & Morgan Kaufman, 2001.

[Dik01b] 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.

[Dik04] Alexander Dikovsky. Dependencies as categories. In G-J. Kruiff 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.

[DM00] Alexander Dikovsky and Larissa Modina. Dependencies on the other side of the curtain. *Traitement Automatique des Langues (TAL)*, 41(1):79–111, 2000.

[Gai61] Haïm Gaifman. 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.

[Gla66] Alexej V. Gladkij. *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.

[GM71] Alexej V. Gladkij and Igor A. Mel'čuk. Grammatiki derev'ev. I. Opyt formalizacii preobrazovanij sintaksičeskix struktur estestvennogo jazyka [Tree grammars. I. A formalism for syntactic transformations in natural languages]. In *Informacionnye voprosy semiotiki, lingvistiki i avtomatičeskogo perevoda [Informational Problems of Semiotics, Linguistics and Automatic Translation]*, volume 1, pages 16–41. 1971. Published In "Linguistics. An International Review", n. 150. April 15, 1975, The Hague : Mouton, pp. 47-82.

[Hay60] D.G. Hays. 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.

[Hud84] Richard Hudson. *Word Grammar.* Basil Blackwell, Oxford-New York, 1984.

[Jac77] Ray Jackendoff. *X-bar Syntax. A Study of Phrase Structure.* MIT Press, Cambridge, 1977.

[JLT75] A.K. Joshi, L.S. Levy, and M. Takahashi. Tree adjunct grammars. *Journ. of Comput. and Syst. Sci.*, 10(1):136–163, 1975.

[JSW91] 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.

[KNR98] Sylvain Kahane, Alexis Nasr, and Owen Rambow. Pseudo-projectivity : A polynomially parsable non-projective dependency grammar. In *Proc. COLING-ACL*, pages 646–652, Montreal, 1998.

[Kru01] Geert-Jan M. Kruijff. *A Categorial-Modal Logical Architecture of Informativity. Dependency Grammar Logic & Information Structure.* PhD thesis, Charles University, Prague, 2001.

[Lam58] Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, pages 154–170, 1958.

[LL96] Vincenzo Lombardo and Leonardo Lesmo. An earley-type recognizer for dependency grammar. In *Proc. 16th COLING*, pages 723–728, 1996.

[LL98] Vincenzo Lombardo and Leonardo Lesmo. Formal aspects and parsing issues of dependency theory. In *Proc. COLING-ACL*, pages 787–793, Montreal, 1998.

[Mar90] Hiroshi Maruyama. Structural disambiguation with constraint propagation. In *Proc. of 28th ACL Annual Meeting*, pages 31–38, 1990.

[Mel97] Igor Mel'čuk. Vers une linguistique ≪Sens–Texte≫ . Leçon inagurale au Collège de France. 1997.

[MM] Michael Moortgat and Glin V. Morrill. Heads and phrases. Type calculus for dependency and constituent structure. Ms OTS, Utrecht. Available at: http://www-lsi.ups.es/~morrill/.

[Mor94] Glin V. Morrill. *Type Logical Grammar. Categorial Logic of Signs.* Kluwer Academic Publishers, Dordrecht, 1994.

[MP87] I. Mel'čuk and N.V. Pertsov. *Surface Syntax of English. A Formal Model Within the Meaning-Text Framework.* John Benjamins Publishing Company, Amsterdam/Philadelphia, 1987.

[PS94] Carl Pollard and Ivan Sag. *Head-driven Phrase Structure Grammar.* CSLI, Stanford, California, 1994.

[RJ82] Kaplan R. and Bresnan J. Lexical functional grammar : A formal system for grammatical representation. In Bresnan J.W., editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, Massachusetts, 1982.

[Rob70] Jane J. Robinson. Dependency structures and transformational rules. 46(2):259–285, 1970.

[ST93] Daniel Sleator and Davy Temperly. Parsing English with a Link Grammar. In *Proc. IWPT'93*, pages 277–291, 1993.

[Tes59] L. Tesnière. *Éléments de syntaxe structurale.* Librairie C. Klincksiek, Paris, 1959.