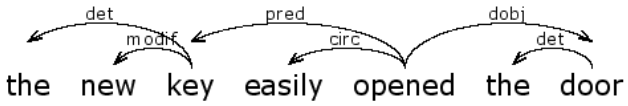
Towards Wide Coverage Categorial Dependency Grammars

Alexander Dikovsky

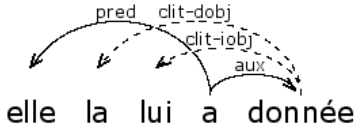
LINA CNRS UMR 6241, Université de Nantes
alexandre.dikovsky@univ-nantes.fr

1 Introduction

Categorial Dependency Grammars (CDG) [5, 4, 2] are categorial grammars generating sentences together with their *dependency structures* (DS). The DS may be trees (DT) as in Fig. 1 or not (cf. Fig . 2). Whatever is the case, they are linearly ordered and this order may be *projective*, as in Fig. 1(a), or not, as in Fig. 1(b). The non-projective order is due to *discontinuous dependencies* in which a governor g is separated from its subordinate s by a word not dominated by g (as it is the case of the auxiliary verb a in Fig. 1(b) separating the participle *donnée* from its pronominalized objects).



(a) A projective DT



(b) A non-projective DT (French: **she* it_{FEM} *to-him has given*)

Figure 1

The main advantage of CDG is that they allow to treat discontinuous non-projective dependencies strictly locally (i.e. within the lexicon entries of the governor and of the subordinate) and to parse the generated languages in a practical polynomial time.

To adapt the CDG for practical use, i.e. for elaboration of wide-coverage grammars, one should resolve the problem of grammar size explosion. Like the classical categorial grammars, CDG are lexicalized, i.e. are defined by a *lexicon*: a function λ assigning to each word a finite set of its dependency types. The primary source of lexicon size explosion is that the number of types per entry is close to the product of the numbers of possible argument subtypes. Another considerable source is a loose word order peculiar to many languages with reach

inflectional morphology (cf. Russian, German, Arabic, etc.). Finally, an excessive lexicon size may be due to massive type sharing between entries. There are various means aiming at a practical solution to this problem: regular expressions in types, supertagging, lexicon structuring. Below we show a way to reduce the lexicon size using *flexible types introduced through specific calculus rules.*

2 Generalized Categorial Dependency Grammars

We will consider the *Generalized CDG* (gCDG) ([5, 2]), which can generate non-tree dependency structures (for instance, a union of superposed trees).

A gCDG type has the form $[l_m\backslash\ldots\backslash l_1\backslash v/r_n/\ldots/r_1]^P$, where v is the (continuous) dependency on the governor, $l_m,\ldots,l_1$ and $r_n,\ldots,r_1$ are the (continuous) dependencies of left and right subordinates in the reverse order, and P is a sequence of *polarized valencies* (the *potential*) determining discontinuous dependencies. Types with empty potentials determine projective DT. E.g., the DT in Fig. 1(a) is determined by the type assignments:

$$\text{the} \mapsto det \quad \text{key} \mapsto [modif*\backslash det\backslash pred] \quad \text{new} \mapsto modif$$
$$\text{easily} \mapsto circ \; \text{opened} \mapsto [circ*\backslash pred\backslash S/dobj] \; \text{door} \mapsto [det\backslash dobj]$$

where the types $modif$ and $circ$ are *iterated* and S is the sentence type. Potentials are sequences of polarized valencies of four kinds: $\swarrow d$ (negative: that of the distant right governor through dependency d), $\searrow d$ (same on the left), $\nwarrow d$ (positive: that of a distant left subordinate through dependency d), $\nearrow d$ (same on the right). A pair of *dual* valencies $\swarrow d$ and $\nwarrow d$ ($\nearrow d$ and $\searrow d$) define a discontinuous dependency d. A negative valency can be anchored on a host word using anchor types $\#(\swarrow d)$, $\#(\searrow d)$. E.g., the lexicon below uniquely determines the DT in Fig. 1(b) ($\swarrow clit{-}dobj, \swarrow clit{-}iobj$ are anchored on the auxiliary a):

$$\text{elle} \mapsto pred \quad \text{la} \mapsto [\#(\swarrow clit{-}dobj)]^{\swarrow clit\text{–}dobj} \quad \text{lui} \mapsto [\#(\swarrow clit{-}iobj)]^{\swarrow clit\text{–}iobj}$$
$$\text{a} \mapsto [\#(\swarrow clit{-}iobj)\backslash\#(\swarrow clit{-}dobj)\backslash pred\backslash S/aux] \quad \text{donnée} \mapsto [aux]^{\nwarrow clit\text{–}dobj\nwarrow clit\text{–}iobj}$$

Such dependency types are formalized with the following calculus [1]

L$^{\mathbf{l}}$. $C^{P_1}[C\backslash\beta]^{P_2} \vdash [\beta]^{P_1P_2}$

I$^{\mathbf{l}}$. $C^{P_1}[C^*\backslash\beta]^{P_2} \vdash [C^*\backslash\beta]^{P_1P_2}$

Ω$^{\mathbf{l}}$. $[C^*\backslash\beta]^P \vdash [\beta]^P$

D$^{\mathbf{l}}$. $\alpha^{P_1(\swarrow C)P(\nwarrow C)P_2} \vdash \alpha^{P_1PP_2}$, if $(\swarrow C)P(\nwarrow C)$ satisfies the pairing rule **FA** (first available): P has no occurrences of $\swarrow C, \nwarrow C$.

L$^{\mathbf{l}}$ is the classical elimination rule. Eliminating the argument subtype $C \neq \#(\alpha)$ it constructs the (projective) dependency C and concatenates the potentials. $C = \#(\alpha)$ creates no dependency. **I**$^{\mathbf{l}}$ derives $k > 0$ instances of C. **Ω**$^{\mathbf{l}}$ serves for the case $k = 0$. **D**$^{\mathbf{l}}$ derives discontinuous dependencies. It pairs and eliminates dual valencies satisfying the rule **FA** to create the discontinuous dependency C.

For a DS D and a string x, let $G(D, x)$ denote the relation: D is constructed in a proof $\Gamma \vdash S$ for some $\Gamma \in \lambda(x)$. Then the *language* generated by G is the set $L(G) =_{df} \{w \mid \exists D\ G(D, w)\}$.

[1] We show left-oriented rules. The right-oriented are symmetric.

The gCDG are very expressive. Evidently, they generate all CF-languages. They can also generate non-CF languages.

Example 1 *The gCDG*

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

generates the language $\{a^n b^n c^n \mid n > 0\}$ *[6]. For instance,* $G_{abc}(D^{(3)}, a^3b^3c^3)$ *holds for the dependency structure (DS) in Fig. 2 and the string* $a^3b^3c^3$ *due to the proof in Fig. 3.*

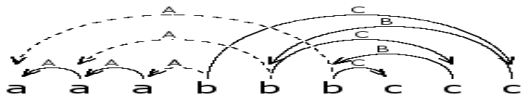


Fig. 2. Dependency structure for $a^3b^3c^3$

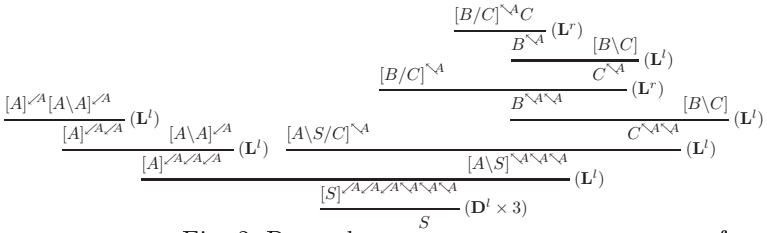


Fig. 3. Dependency structure correctness proof

Seemingly, it is incomparable with mildly CS languages ([8]) generated by multi-component TAG, linear CF rewrite systems and some other grammars. $\mathcal{L}(gCDG)$ contains non-TAG languages, e.g. $L^{(m)} = \{a_1^n a_2^n ... a_m^n \parallel n \geq 1\}$ for all $m > 0$. In particular, it contains the language $MIX = \{w \in \{a,b,c\}^+ \mid |w|_a = |w|_b = |w|_c\}$, for which E. Bach conjectures that it is not mildly CS. On the other hand, we conjecture [3] that this family does not contain the copy language $L_{copy} = \{xx \mid x \in \{a,b\}^*\}$, which is TAG [2]. gCDG have an efficient polynomial time parsing algorithm [3, 2] which resides upon a property of independence of basic types and polarized valencies expressed in terms of *projections* of types and *"well-bracketing"* criteria for potentials.

For a sequence of types γ, its *local projection* $\|\gamma\|_l$ and *de valency projection* $\|\gamma\|_v$ are defined as follows:

1. $\|\varepsilon\|_l = \|\varepsilon\|_v = \varepsilon$; $\|\alpha\gamma\|_l = \|\alpha\|_l\|\gamma\|_l$ and $\|\alpha\gamma\|_v = \|\alpha\|_v\|\gamma\|_v$ for a type α.

2. $\|C^P\|_l = C$ et $\|C^P\|_v = P$ for every type C^P.

To speak about "well-bracketing" of potentials, we interpret $\swarrow d$ and $\nearrow d$ as *left brackets* and $\nwarrow d$ and $\searrow d$ as *right brackets*. A potential is *balanced* if it is well bracketed in the usual sense.

Let $\mathbf{c}$ be the projective core of the dependency calculus, consisting of the rules $\mathbf{L}$, $\mathbf{I}$ and $\mathbf{\Omega}$ and $\vdash_{\mathbf{c}}$ denote the provability relation in this sub-calculus. Then the *projections independence* property of gCDG [3, 2] is formulated as follows.

Theorem 1 *For a gCDG G with lexicon* λ *and a string* x, $x \in L(G)$ *iff there is* $\Gamma \in \lambda(x)$ *such that* $\|\Gamma\|_l \vdash_{\mathbf{c}} S$ *and* $\|\Gamma\|_v$ *is balanced.*

[2] It doesn't prevent that gCDG express the cross-serial dependencies [5].

3 Extension of gCDG with Flexible Type Rules

1. Alternative rules.
The first extension allows to use alternative choice of a subtype in a type: $(C_1|\dots|C_k)$ meaning intuitively "one of types C_i". E.g., assigning to *are* the type $[pred\backslash S/(n-copul|c-copul|q-copul)]$ one can replace three entries: $\lambda : are \mapsto [pred\backslash S/n-copul])$, $\lambda : are \mapsto [pred\backslash S/c-copul])$, $\lambda : are \mapsto [pred\backslash S/q\text{–}copul])$ by one entry: $\lambda : are \mapsto [pred\backslash S/(n\text{–}copul|c\text{–}copul|q\text{–}copul)])$. Here are the calculus rules allowing such types:

$\mathbf{LA^l_{gov}}$. $C^{P_1}[(C|\alpha)\backslash\beta]^{P_2} \vdash [\beta]^{P_1P_2}$
$\mathbf{LA^l_{sub}}$. $(C|\alpha)^{P_1}[C\backslash\beta]^{P_2} \vdash [\beta]^{P_1P_2}$
$\mathbf{LA^l_{gs}}$. $(C|\alpha_1)^{P_1}[(C|\alpha_2)\backslash\beta]^{P_2} \vdash [\beta]^{P_1P_2}$

2. Flexible order rules.
The next extension allows to eliminate left (right) subordinates whose position with respect to other subordinates is not fixed. It is supported by two rules treating the set of such not restrained subordinates as a bag. The first rule eliminates the subordinates belonging to the bag. The second rule eliminates fixed-position subordinates irrelative to presence and position of the bag subordinates.

2.1. X-sided bag rules. We show inside and outside left-sided bag rules (X=l):

$\mathbf{LB^l_i}$. $C^{P_1}[\{C,\alpha\}\backslash\beta]^{P_2} \vdash [\{\alpha\}\backslash\beta]^{P_1P_2}$, $C \notin \beta$.
$\mathbf{LB^l_o}$. $C^{P_1}[\{\alpha\}\backslash C\backslash\beta]^{P_2} \vdash [\{\alpha\}\backslash\beta]^{P_1P_2}$, $C \notin \alpha$.

The next two rules support elimination of subordinates whose position with respect to the governor is also not fixed.

2.2. Unoriented bag rules. We show inside and outside unoriented bag rules:

$\mathbf{LB^u_i}$. (a) $C^{P_1}[\beta]^{P_2}_{\{C,\alpha\}} \vdash [\beta]^{P_1P_2}_{\{\alpha\}}$, (b) $[\beta]^{P_2}_{\{C,\alpha\}}C^{P_1} \vdash [\beta]^{P_2P_1}_{\{\alpha\}}$ $(C \notin \beta)$.
$\mathbf{LB^u_o}$. $C^{P_1}[C\backslash\beta]^{P_2}_{\{\alpha\}} \vdash [\beta]^{P_1P_2}_{\{\alpha\}}$, $C \notin \alpha$.

Example 2 *Let us consider the following fragment of a lexicon for German:*

$$\begin{cases} deshalb \mapsto [circ] & gab \mapsto [circ^*\backslash S/\{pred, dobj, iobj\}] \\ Frau \mapsto [det\text{–}df\backslash iobj] & das \mapsto [det\text{–}a] \\ Mann \mapsto [det\text{–}n\backslash pred] & der \mapsto [(det\text{–}n|det\text{–}df)] \\ Buch \mapsto [det\text{–}a\backslash dobj] \end{cases}$$

The proof in Fig. 4 shows correctness of this assignment for the sentence deshalb gab der Mann der Frau das Buch *and for its DS. The proof in Fig. 5 shows correctness of the same assignment for the sentence* deshalb gab das Buch der Mann der Frau *with a different word order and a different DS.*

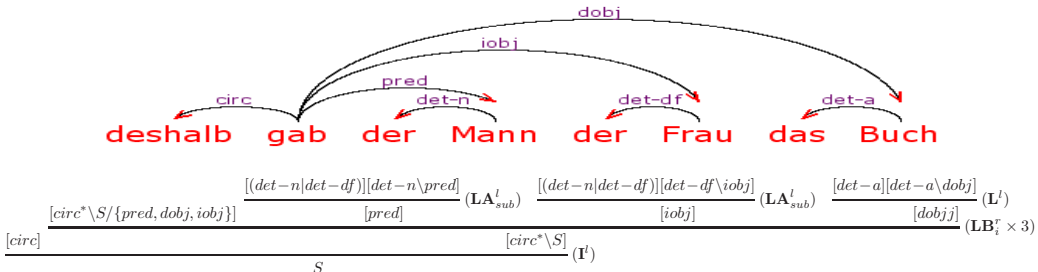


Fig. 4. Correctness proof for the DS of *deshalb gab der Mann der Frau das Buch*

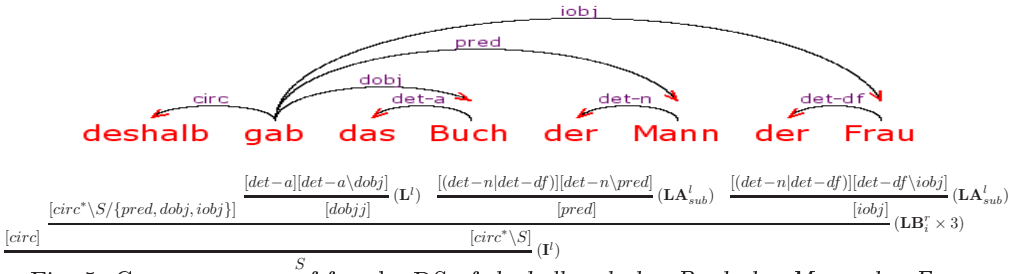


Fig. 5. Correctness proof for the DS of *deshalb gab das Buch der Mann der Frau*

Let us denote by $\mathcal{L}(fgCDG)$ the set of all languages generated by the *extended flexible type gCDG*. It is easy to prove that:
1. $\mathcal{L}(fgCDG) = \mathcal{L}(gCDG)$,
2. Theorem 1 holds for the extended gCDG as well, and
3. Parsing of $fgCDG$ has the same complexity as for $gCDG$.

Remark. One should distinguish three, in principle different, but in practice mixing together techniques: *meta-expressions for rules*, *grammar factorization*, and *conservative calculus extensions*. Early techniques (cf. link grammars [7]) used meta-expressions. Nowadays, they are accompanied by grammar factorization through classifications, inheritance etc. (cf. successful combination of the two applied to TAGs, e.g. [1]). The two techniques require compilation into a target grammar. The third techniques, applied to gCDG in this paper, needs no compilation. When it applies to an original calculus, it gives an equivalent calculus of the same complexity applied to more compact grammars without changing them. Factorization of flexible gCDG is now under study.

References

1. E. de la Clergerie. From metagrammars to factorized tag/tig parsers. In *In Proc. of IWPT'2005*, pages 190–191, Vancouver, Canada, 2005.
2. M. Dekhtyar and A. Dikovsky. Generalized categorial dependency grammars. In A. Aviron et al., editor, *Trakhtenbrot/Festschrift*, LNCS 4800, pages 230–255. Springer Verlag, 2008.
3. Michael Dekhtyar and Alexander Dikovsky. Categorial dependency grammars. In M. Moortgat and V. Prince, editors, *Proc. of Intern. Conf. on Categorial Grammars*, pages 76–91, Montpellier, 2004.
4. A. Dikovsky. A finite-state functional grammar architecture. In D. Leivant and Ruy de Queiroz, editors, *Logic, Language, Information and Computation. Proc. of the 14th Intern. Workshop WoLLIC 2007*, LNCS 4576, pages 131–146, Rio de Janeiro, Bresil, 2007. Springer Verlag.
5. A. Dikovsky. Multimodal categorial dependency grammars. In *Proc. of the 12th Conference on Formal Grammar*, pages 1–12, Dublin, Ireland, 2007.
6. Alexander Dikovsky. Dependencies as categories. In *"Recent Advances in Dependency Grammars". COLING'04 Workshop*, pages 90–97, 2004.
7. Daniel Sleator and Davy Temperly. Parsing English with a Link Grammar. In *Proc. IWPT'93*, pages 277–291, 1993.
8. K. Vijay-Shanker and D.J. Weir. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27:511–545, 1994.