Categorial Dependency Grammars: from Theory to Large Scale Grammars

Alexander Dikovsky
LINA CNRS UMR 6241, Université de Nantes
Alexandre.Dikovsky@univ-nantes.fr

Abstract

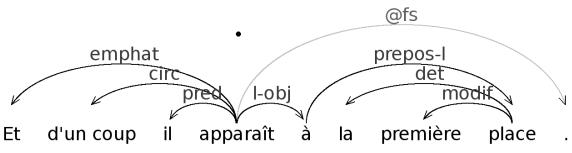
Categorial Dependency Grammars (CDG) generate unlimited projective and non-projective dependency structures, are completely lexicalized and analyzed in polynomial time.

We present an extension of the CDG, also analyzed in polynomial time and dedicated for large scale dependency grammars. We define for the extended CDG a specific method of "Structural Bootstrapping" consisting in incremental construction of extended CDG from representative samples of dependency structures. We also outline a wide coverage dependency grammar of French developed using this method.

1 Introduction

Categorial Dependency Grammars (CDG) were introduced in (Dikovsky, 2004). Since then, they were intensively studied (e.g., see (Béchet et al., 2004; Dekhtyar and Dikovsky, 2008; Dekhtyar et al., 2010)). CDG is very expressive. In particular, simple CDG generate such non-CF languages as $L^{(m)} = \{a_1^n a_2^n ... a_m^n \mid n \geq 1\}$ for all $m > 0$ and $MIX = \{w \in \{a, b, c\}^+ \mid |w|_a = |w|_b = |w|_c\}$. At the same time, CDG are recognized in polynomial time. CDG have interesting mathematical properties: an extension of CDG defines an Abstract Family of Languages (AFL) (Dekhtyar and Dikovsky, 2008; Dekhtyar et al., 2010)[1], they are equivalent to real time pushdown automata with independent counters (Karlov, 2008), interesting sufficient conditions of learning CDG in the limit were recently found (Béchet et al., 2004; Béchet et al., 2010; Béchet et al., 2011).

[1]CDG-languages are closed under all AFL operations, but iteration.

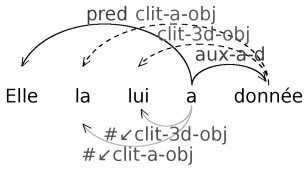


(*and all of a sudden he appears foremost*)

Figure 1: Projective DS

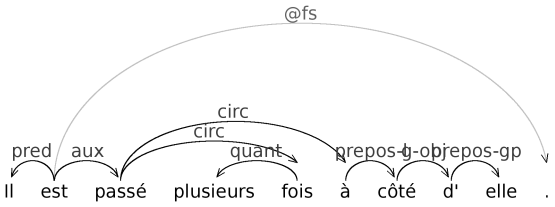
At the same time, the exact relationship between the weak generative power of the CDG and that of the so called mildly context sensitive grammars (see e.g. (Joshi et al., 1991; Shanker and Weir, 1994)) is not known.

CDG have important advantages which make them a convenient and natural means of definition of wide-coverage dependency grammars. First, they are completely lexicalized, as it is the case of all categorial, and more generally, type logical grammars (Bar-Hillel et al., 1960; Lambek, 1961; Lambek, 1999; Steedman, 1996). The second advantage of CDG is that they naturally and directly express unlimited dependency structures (DS). Basically, CDG define DS in terms of valencies of words, i.e. in terms very close to those of the traditional linguistic theories of syntax. Of course, as all dependency grammars (e.g. (Gaifman, 1961; Maruyama, 1990; Sleator and Temperly, 1993; Debusmann et al., 2001)), they express the projective DS, i.e. those in which the dependencies do not cross (as the one in Fig. 1). But they express as well the non-projective DS, in which they may cross (as in the DS shown in Fig. 2). Non-projective DS are a challenge for dependency grammars. Generally, the grammars expressing them are untractable (cf. (Debusmann et al., 2001)) or need some constraints on the DS in order to be polynomially analyzed (cf. (Ka-



(**she it[fem.] to him has given*)

Figure 2: Non-projective DS



(*he passed by her side several times*)

Figure 3: Repetitive dependency *circ*

hane et al., 1998; Bröker, 2000)). As to the CDG, they are analyzed in a reasonable polynomial time using a rather standard tabular dynamic programming algorithm (see (Dekhtyar and Dikovsky, 2008)), and this is their third advantage. Fourth, an extension of CDG by regular type expressions (`RTE`) specially designed for large scale grammars was proposed in (Dikovsky, 2009). We outline this extension below. Importantly, the extended CDG are also analyzed in polynomial time.

In this paper we define a simple and practical `Structural Bootstrapping Method` of incremental development of large scale extended CDG from representative samples of DS. Using this method and a toolkit specially designed for CDG (Alfared et al., 2011), we have developed in a short space of time a rather complete dependency grammar of French, briefly described below.

The plan of this paper is as follows. The next Section introduces the CDG. Section 3 presents their extension by RTE. In Section 4 is defined and illustrated the Method of Structural Bootstraping of extended CDG. Finally, a wide scope extended CDG of French developed by this method is outlined in Section 5.

2 Categorial Dependency Grammars

CDG originate from a straightforward encoding of DS in terms of dependency relation valencies of words. Basically, they are classical categorial grammars with subtypes interpreted as dependency valencies and with categories extended by potentials defining non-projective dependencies. Valencies of projective and of non-projective dependencies are encoded differently.

When in a DS D there is an arc $w_1 \xrightarrow{d} w_2$, we say that d is a dependency between w_1 and w_2, w_1 is the `governor` and w_2 is `subordinate` to w_1 through d. E.g., in Fig. 1, *il* (*he*) is subordinate to *apparaît* (*appears*) through *pred* and the locative case preposition *à* governs *place* through dependency $prepos-l$.

The valency of a governor of w through dependency d is encoded by d itself. In particular, the no-governor valency of the root word is encoded by S (a special symbol called `axiom`).

The valency of a left subordinate of w through projective dependency l is encoded by $l\backslash$ and that of a right subordinate through projective dependency r is encoded by $/r$. The set of all left valencies of a word is encoded by the concatenation of their codes. So, for instance, the valencies of projective dependencies of the root word *apparaît* in the DS in Fig. 1 is encoded by the expression $pred\backslash circ\backslash emphat\backslash S/@fs/l-obj$.

The repetitive dependencies are a special case. A dependency d is `repetitive` (see (Mel'čuk, 1988)) if a word may have more than one subordinates through d. The valency of left repetitive dependency d is encoded by $d^*\backslash$ (the right one is encoded by $/d^*$. So, e.g. in the DS in Fig. 3, the valencies of projective dependencies of the word *passé* (*passed*) are encoded by the expression $aux/circ^*$.

In CDG, the non-projective dependency valencies of a word w are `polarized`. They are of four kinds: $\swarrow v$, $\searrow v$ (`negative`) and $\nwarrow v$, $\nearrow v$ `positive`. E.g., wnen a word w has valency $\swarrow v$, it intuitively means that its governor through dependency v must occur somewhere on the right. Two polarized valencies with the same valency name v and orientation, but with the opposite signs are `dual`. Together they define non-projective dependency v. The (possibly empty) set of all non-projective dependencies of a word w is encoded by the concatenation of the correspond-

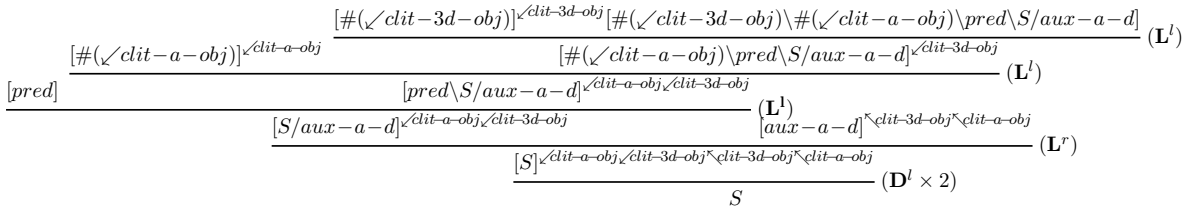


Figure 4: Dependency structure correctness proof

ing polarized valencies called `potential` of w.[2] E.g., in the DS in Fig. 2, the participle *donnée* has potential $\nwarrow clit-a-obj \nwarrow clit-3d-obj$, which means that it needs, somewhere on its left, a word subordinate through dependency $clit-a-obj$ and also another word subordinate through dependency $clit-3d-obj$. At the same time, the accusative case clitic *la* (*it*[*fem.*]) has potential $\swarrow clit-a-obj$ and the dative case clitic *lui* (*to him*) has potential $\swarrow clit–3d–obj$. The proper pairing of these dual valencies with those of the participle defines two non-projective dependencies between the participle and its cliticized complements.

The expression

$$t = [l_m \backslash \ldots l_1 \backslash h / r_1 \ldots / r_n]^P$$

$(m, n \geq 0)$ is called a `type` of a word w if:
(*i*) $l_m\backslash \ldots l_1\backslash$ and $/r_1 \ldots /r_n$ encode respectively left and right projective dependency valencies of w,
(*ii*) h is a governor (no-governor) valency and
(*iii*) P, the potential of w, encodes its valencies of non-projective dependencies.
$l_m, \ldots, l_1$ are `left subtypes` of t, $r_1, \ldots, r_n$ are its `right subtypes` and h is its `head subtype`.

Below we use CDG with non-empty head subtypes. When a type $[\alpha\backslash d/\beta]^P$ has a negative valency in its potential P, say $P = \swarrow vP'$, the word w with this type has two governors: one through v, the other through d. In such cases we use special head subtypes $d = \#(A)$, called `anchors`, to express the adjacency of w to a `host` word w_0. The anchor dependencies are displayed below the sentence for a better readability. E.g., the DS in Fig. 2 is defined by the following assignment of types to words:
elle $\mapsto [pred]$,
la $\mapsto [\#(\swarrow clit-a-obj)]^{\swarrow clit–a–obj}$,
lui $\mapsto [\#(\swarrow clit-3d-obj)]^{\swarrow clit–3d–obj}$,
donnée $\mapsto [aux-a-d]^{\nwarrow clit–a–obj \nwarrow clit–3d–obj}$,
a $\mapsto [\#(\swarrow clit–3d–obj)\backslash\#(\swarrow clit–a–obj)\backslash pred \backslash S/aux-a-d]$.
Due to the anchor subtypes $\#(\swarrow clit–3d–obj)$, $\#(\swarrow clit-a-obj)$ in the type of the auxiliary verb *a* (*has*), it serves as the host verb for both clitics and also defines their precedence order. Derivability of DS in CDG is formalized through the following calculus [3] (with C being a dependency, H being a dependency or an anchor and V being a polarized valency):

L$^{\mathbf{l}}$. $H^{P_1}[H\backslash\beta]^{P_2} \vdash [\beta]^{P_1P_2}$
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$^{\mathbf{l}}$. $\alpha^{P_1(\swarrow V)P(\nwarrow V)P_2} \vdash \alpha^{P_1PP_2}$, if the potential $(\swarrow V)P(\nwarrow V)$ satisfies the following pairing rule **FA** (first available):[4]

FA : P has no occurrences of $\swarrow V, \nwarrow V$.

L$^{\mathbf{l}}$ is the classical elimination rule. Eliminating the argument subtype $H \neq \#(\alpha)$ it constructs the (`projective`) dependency H and concatenates the potentials. $H = \#(\alpha)$ creates the `anchor dependency`. **I**$^{\mathbf{l}}$ derives $k > 0$ instances of C. **Ω**$^{\mathbf{l}}$ serves for the case $k = 0$. **D**$^{\mathbf{l}}$ creates `non-projective dependencies`. It pairs and eliminates dual valencies with name V satisfying the rule FA to create the non-projective dependency V.

In Fig. 4 we show a proof of correctness of DS in Fig. 2 with respect to the type assignment shown above.

A CDG G is defined by its dictionary W and its `lexicon` λ, an assignment of finite sets of types to words in W. G defines a DS D of a sentence $x = w_1 \ldots w_n$ and x is generated by G (denoted $D \in \Delta(G)$ and $x \in L(G)$) if it is possible to assign through λ a type t_i to every word w_i so that the obtained type string $t_1 \ldots t_n$ were reducible to the axiom S. $L(G)$ is the `language` and $\Delta(G)$ is the `structure language` generated by G.

[2] Their order is irrelevant (so one may choose a standard lexicographic order).
[3] We show left-oriented rules. The right-oriented rules are symmetrical.
[4] Cf. a different pairing rule in (Dikovsky, 2007).

$Vt(F = fin, C = a) \mapsto \{pred?, neg?, vocative?, \#(\swarrow expletive)?, circ^*\}[\{lpar?, \#(\swarrow coref)?, \#(\swarrow select)?\}$
$\backslash(\#(\swarrow compos-neg)|\#(\swarrow restr-neg)|\#(\searrow compos-neg)|\#(\searrow restr-neg))?\ \backslash interrog?\backslash emphat?\backslash S/$
$(\#(\searrow fs)|\#(\searrow qu)|\#(\searrow xl))/coordv^*/(a-obj|claus|pre-inf|inf)?/\{rpar?, \#(\searrow modif)?, \#(\searrow attr)?,$
$\#(\searrow appos)?, \#(\searrow dist-rel)?, \#(\searrow aggr)?\}]$

Figure 5: A RTE for transitive French verbs

3 Extended CDG

CDG is a theoretical model not adapted for wide coverage grammars. The main problem with wide coverage is the excessive sharing of subtypes in types. For lexicons running to hundreds of thousands of lexical units it results in a combinatorial explosion of spurious ambiguity and in a significant parsing slowdown. Wide coverage grammars face many hard problems, e.g. those of compound lexical entries including complex numbers, compound terms, proper names, etc. and also that of flexible precedence order. An extension of CDG well adapted for wide coverage grammars is proposed in (Dikovsky, 2009).

The extended CDG use classes of words in the place of words and use restricted regular expressions defining sets of types in the place of types. I.e., the dictionary W is covered by classes: $W = \bigcup_{i \in I} C_i$ and the lexicon λ assigns sets of regular expressions to classes. At that:

- all words in a class C share the types defined by the expressions assigned to C,
- every word has all types of the classes to which it belongs.

The `regular type expressions` (`RTE`) we describe below are flat (i.e. bounded depth). In these expressions, C, C_i are dependency names or anchors, B is a `primitive` type, i.e. a dependency name, or an anchor or an iterated or optional type, and H is a choice.

Choice: $(C_1|\dots|C_k)$; $(C) =_{df} C$

Optional choice: $(C_1|\dots|C_k)?$; $(C)? =_{df} C?$

Iteration: $(C_1|\dots|C_k)^*$; $(C)^* =_{df} C^*$

Dispersed subtypes expressing flexible order.

Left: $[\{\alpha_1, B, \alpha_2\}\backslash\alpha\backslash H/\beta]^P$

Right: $[\alpha\backslash H/\beta/\{\alpha_1, B, \alpha_2\}]^P$

Two-way: $\{\alpha_1, B, \alpha_2\}[\alpha\backslash H/\beta]^P$

Here is a fragment of the extended calculus:

1. Choice rules:

LC$^{\mathbf{l}}$. $C^{P_1}[(\alpha_1|C|\alpha_2)\backslash\beta]^{P_2} \vdash [\beta]^{P_1P_2}$

IC$^{\mathbf{l}}$. $C^{P_1}[(\alpha_1|C|\alpha_2)^*\backslash\beta]^{P_2} \vdash [(\alpha_1|C|\alpha_2)^*\backslash\beta]^{P_1P_2}$

ΩC$^{\mathbf{l}}$. $[(\alpha_1|C|\alpha_2)^*\backslash\beta]^P \vdash [\beta]^P$

(**DC$^{\mathbf{l}}$** is as **D$^{\mathbf{l}}$** in the CDG calculus).

2. Dispersed subtypes rules:

LD$^{\mathbf{l}}$. $H^{P_1}[\{\alpha\}\backslash H\backslash\beta/\{\gamma\}]^{P_2} \vdash [\{\alpha\}\backslash\beta/\{\gamma\}]^{P_1P_2}$

ID$^{\mathbf{l}}$. $C^{P_1}[\{\alpha_1, C^*, \alpha_2\}\backslash\beta/\{\gamma\}]^{P_2} \vdash$
$[\{\alpha_1, C^*, \alpha_2\}\backslash\beta/\{\gamma\}]^{P_1P_2}$

ΩD$^{\mathbf{l}}$. $[\{\alpha_1, C^*, \alpha_2\}\backslash\beta/\{\gamma\}]^P \vdash [\{\alpha_1, \alpha_2\}\backslash\beta/\{\gamma\}]^P$

(**DD$^{\mathbf{l}}$** as **D$^{\mathbf{l}}$** in the CDG calculus).

E.g., the rule **ID$^{\mathbf{l}}$** intuitively says that the dispersed iterated subordinates through C may be found in any position at the left of the governor with type $[\{\alpha_1, C^*, \alpha_2\}\backslash\beta/\{\gamma\}]^{P_2}$.

Fig. 5 shows an example of one of RTE assigned to the class $Vt(F = fin, C = a)$ of French transitive verbs in finite forms. It defines the simplest case where the complement is niether fronted nor cliticized. E.g., it states that the subject (subordinate through *pred*) may occur at the left or at the right of the verb, whereas the (exactly defined) position of the direct object (subordinate through $a-obj$) is at its right, and in the same position may be found a subordinate clause and a prepositional or preposition-less infinitive phrase.

The RTE and the classes do not extend the expressive power of CDG. At the same time, they dramatically reduce the grammar size. Sure, the unfolding of an extended CDG may exponentially blow up its size. However, due to the extended type calculus the polynomial time parsing algorithm of (Dekhtyar and Dikovsky, 2008) can be adapted to parse them directly, without unfolding. So the RTE are well adapted for large scale grammars. But still more, they are also ment for incremental bootstrapping of extended CDG from DS.

4 Structural Bootstrapping

In (Béchet et al., 2010; Béchet et al., 2011), it is proved that, in contrast to the constituent-structure grammars, even the projective CDG assigning one type per word cannot be learned from the DS they generate. This means that CDG cannot be automatically computed from dependency treebanks. The reason is that they express repeatable dependencies through iteration (and not through recursion). In these

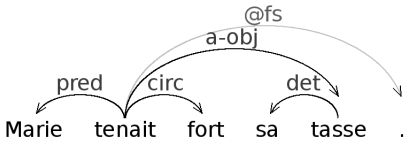


Figure 6: DS of (s_1)

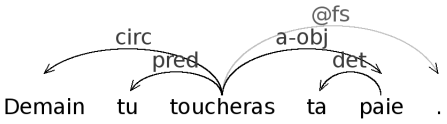


Figure 7: DS of (s_2)

papers are also defined and proved algorithms which learn from DS some subclasses of extended CDG, under reasonable conditions on the use of iteration. These partial solutions are still far from being practical. Below we present an intuitive heuristic method of construction of extended CDG from DS.

This method, we call `structural bootstrapping`, consists in that an extended CDG is incrementally constructed from a sample of DS, element by element. Ideally, the sample should be representative with respect to the surface syntax of the language. We suppose that the extended CDG is defined as $G = (W, \mathbf{D}, \mathbf{V}, S, \lambda)$, where W is its dictionary with a classification $W = \bigcup_{i \in I} C_i$, $\mathbf{D}$ is the set of dependency names, $\mathbf{V}$ is the set of valency names, S is the axiom and λ is the lexicon.

The method is based on a genericity partial order (`PO`) $\preceq$ on extended CDG, compatible with the inclusion of DS-languages: $G \preceq G' \Rightarrow \Delta(G) \subseteq \Delta(G')$. $\preceq$ is the closure by reflexivity, transitivity and by type construction of the following basic PO (below t is a subtype, X is a list of alternatives and $(t) =_{df} t$):

1. $t \lesssim (t|X)$
2. $(t|X) \lesssim (t|X)?$
3. $(t|X)? \lesssim (t|X)^*$
4. $\{\gamma\}[\{\gamma_1\}\backslash t \backslash \beta]^P \lesssim \{\gamma\}[\{t, \gamma_1\}\backslash \beta]^P$
5. $\{\gamma\}[\{t, \gamma_1\}\backslash \beta]^P \lesssim \{t, \gamma\}[\{\gamma_1\}\backslash \beta]^P$

(similar for right subtypes)

6. $\{\gamma\}[\alpha/\{t, \gamma_1\}]^P \lesssim \{t, \gamma\}[\alpha/\{\gamma_1\}]^P$.

Basically, the Structural Bootstrapping Method consists in extracting from the sample DS the vicinities of words and in merging them into minimally generalized RTE of the preceding grammar. By `vicinity` of a word w in a DS D we mean the maximal subgraph $V(w, D)$ of D with the nodes $\{w, w_1, \ldots, w_m\}$, $w_1, \ldots, w_m$ being the subordinates of w in D. Here is a schematic description of the method.

Structural Bootstrapping Method:
Input: Extended CDG G_{in};
DS D_x of a sentence x //next DS.
Output: Extended CDG G_{out} generating D_x.
let $G_{in} = (W, \mathbf{C}, \mathbf{V}, S, \lambda)$ where $W = \bigcup_{i \in I} C_i$;
if $(D_x \in \Delta(G_{in}))$
then $G_{out} = G_{in}$
else
  for every word $w \in x$
    if $(w \in W)$
    then select a class C such that $w \in C$;
    else select a class C and add w to C
    end;
    find the vicinity $V(w, D)$;
    if $(V(w, D)$ is generated by a RTE $t \in \lambda(C))$
    then $\lambda'(C) = \lambda(C)$
    else select RTE $t \in \lambda(C)$;
      find *minimal* RTE $t' \succ t$ *generating* $V(w, D)$;
      set $\lambda'(C) = (\lambda(C) - \{t\}) \cup \{t'\}$,
      $\lambda'(C_1) = \lambda(C_1)$ for every $C_1 \neq C$
    end
  until $D_x \in \Delta((W, \mathbf{C}, \mathbf{V}, S, \lambda'))$
end;
return $G_{out} = (W, \mathbf{C}, \mathbf{V}, S, \lambda')$

Let us see how may evolve RTE of transitive verbs. Suppose that the class $Vt(F = fin, C = a)$ contains the verbs *tenait* (*took*), *toucheras* (*will get*, when applied to wages) and *mettrait* (*might put*). This is how the Structural Bootstrapping Method might change this class when applied to the following sample of sentences:

(s_1) *Marie tenait fort sa tasse.* (*Mary held tight her cup.*)

(s_2) *Demain tu toucheras ta paie.* (*Tomorrow you will get your wage.*)

(s_3) *Où mettrait−elle la clé?* (*Where might she put the key?*)

From the DS of (s_1) in Fig. 6 we have:

$Vt(F = fin, C = a) \mapsto$
$[pred\backslash S/@fs/a{-}obj/circ]$.

The DS of (s_2) in Fig. 7 induces the following generalization:

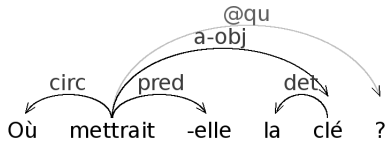


Figure 8: DS of (s_3)

$Vt(F = fin, C = a) \mapsto$
$\{circ^*\}[pred\backslash S/@fs/a-obj]$.

Finally, from the DS of (s_3) in Fig. 8 we obtain a still more general RTE:

$Vt(F = fin, C = a) \mapsto$
$\{pred, circ^*\}[S/(@fs|@qu)/a-obj]$.

In practice, the RTE generalization effected by the main operation:

find minimal RTE $t' \succ t$ generating $V(w, D)$ carries over to all other RTE $t'' \in \lambda(C)$ representing the same syntactic function as t in a compatible local context. E.g., the subject inversion as in (s_3) may also be applied to the RTE defining the coordinated clauses, but not to that defining the parenthetical clauses.

To see that this method is incremental, we should extend the partial order of generalization $\preceq$ to the extended CDG:

1. $\tau \preceq \tau'$ for sets of RTE τ, τ', if either:
(i) $\tau' = \tau \cup \{t\}$ for a RTE $t \notin \tau$ or
(ii) $\tau = \tau_0 \cup \{t'\}$ and $\tau' = \tau_0 \cup \{t''\}$
for a set of RTE τ_0 and some RTE t', t'' such that $t' \preceq t''$.

2. $\lambda \preceq \lambda'$ for two RTE assignments λ and λ', if $\lambda(C') \preceq \lambda'(C')$ for a class C' and $\lambda(C) = \lambda'(C)$ for all classes $C \neq C'$.

3. $\preceq_{gener}$ is the `genericity` PO which is the reflexive-transitive closure of the PO $\preceq$.

4. For CDG G_1 with lexicon λ and G_2 with lexicon λ', $G_1 \preceq_{gener} G_2$ if $\lambda \preceq \lambda'$.

Now, it is not difficult to see the incrementality of this method in the sense that, if $G_1 \preceq_{gener} G_2$, then $\Delta(G_1) \subseteq \Delta(G_2)$.

Application of the Structural Bootsrapping Method in practice needs several resources. First of all, being applied directly as it is defined above, the method will always give grammars with a lexicon limited to that of the sample of representative sentences. So one should choose a morpho-syntactically annotated dictionary of the language (`MS-dictionary`) and to integrate it into the grammar establishing a correspondence between its categories and the grammar's classes. Besides this, it is needed an efficient parser complete with respect to the class of all extended CDG.

5 Bootstrapping of a Wide Coverage CDG of French

The Structural Bootstrapping Method was applied to develop a wide coverage extended CDG of French. Its kernel part (Version 1) was bootstrapped from about 400 French sentences during half a year. In this phase, the method was applied completely incrementally. Then, after two months' long joint work with two colleagues, this grammar was integrared with the freely available MS-dictionary of French Lefff 3.0 (Sagot, 2010) containing 536,375 entries corresponding to 110,477 lemmas. The transition to this integrated Version 2 was non-monotone because the initial lexical classification was to be adapted to Lefff 3.0 and also because of a reorganization of prepositional dependencies. In Version 1 we more or less followed the so callled "pronominal approach" (see (van den Eynde and Mertens, 2003)), but finally we have passed to a system of pronominal and prepositional dependencies based on the case of pronouns. The Version 2 incrementally evolved to Version 3 into which were introduced various more peripheral "small syntax" constructions extracted from DS of about 200 more French sentences. The last two non-monotone updates of the grammar gave the Versions 3.1, 3.2. They were due to a reorganization of verbal RTE, leading to a simple and symmetrical system of negation dependencies and of parenthetical clauses. Basically, the bootstrapping process has stabilized already on Version 2. Till then the grammar keeps the main body of its RTE.

Version 3.2 of the CDG of French covers the major part of French syntax including:

- negation, the main binary negation: *ne ... pas | jamais | plus, ...* and the ternary restrictive negation: *ne ... que* as in *Eve n'a donné a Adam qu'une pomme* (Eve gave to Adam only one apple);

- reflexives and clitics: *Les loups ne se dévorent pas entre eux* (The wolfs do not eat up one another), see also the DS in Fig. 2;

- topicalized complements: *À ces départs s'ajoutent trois autres* (To these departures are added three more);

- clefting: *C' est très amicalement qu'Alain nous*

Examples	Classes			Regular Expressions	Dependencies	
	total	verbal	nominal		projective	non-projective
~ 600	185	46	7	~ 3120	84(9 *par*)	20(3 *par*)

(where $n(m\ par)$ means n of which m are parametrized)
Tab. 1. Parameters of the CDG for French constructed by bootstrapping

a reçu (It is very friendly that Alain has received us);

- subordinate and relative clauses: *Maintenant, tous les soirs, quand il l'avait ramenée chez elle, il fallait qu'il entrât* (Now, every evening, when he accompanied her home, he was obliged to enter);

- interrogative clauses, order inversion: *Qui cela aurait − il pu être*? (*Who this would it be ?);

- light verbs, e.g. *Le laisser faire mal à ma soeur était ma première erreur* (To let him cause damage to my sister was my first error);

- partial extraction from a compement: *Il m'en reste une très facile* (I have one [*fem.*] more resting, a very simple one);

- comparatives: *Il est deux fois plus grand qu'elle* (He is twice as great as she);

- vocatives and co-reference: *Ce truc, restons − en là, Adam*! (This matter, let us let it alone, Adam!),

- expletives: *Un voleur, de temps en temps, ça se repose* (A thief, from time to time, it takes a rest);

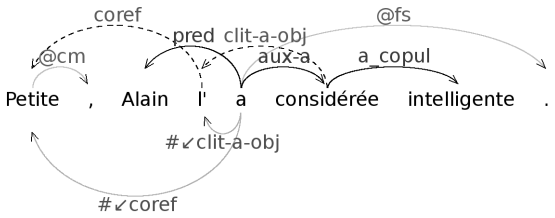
- aggregation: *Adam en a six ou sept rouges* (Adam has six or seven of them red);

- coordination with ellipsis: *J'ai reçu cette notice, et lui non.* (I have received this notice, and he not);

- extracted post-position modifiers: *Le gros chat tigré guette la souris, immobile et silencieux* (The fat stripy cat watched for a mouse, immovable and silent).

Table 1 shows some parameters of this grammar. It has 185 classes 46 of which are verbal, 7 are nominal, 9 are adjectival, 14 are adverbial and in the rest there are the multiple classes of prepositions, pronouns, numerals, determiners, conjunctions, particles, collocations and punctuation markers. The grammar uses 104 dependencies 84 of which are projective and 20 are non-projective. In fact, their number is greater because many of them are parametrized. E.g., there are 6 non-projective clitic dependencies $\swarrow$ *clit–C–obj*, in which C is the person-case parameter (cf. $\swarrow$ *clit–3d–obj*). Let us see the grammar in more detail.

Main principles. 1. This grammar is intended for analysis (not for generation) of correct sentences. So only the oppositions distinguishing between the dependency relations and the order are taken into account. E.g., the agreement in number and in gender do not count, whereas the agreement in person is partially used to define the order of clitics. At the same time, the principle of minimality of the set of oppositions (see (Mel'čuk, 1988; Mel'čuk and Iordanskaja, 2000)) is abandoned in favour of a better distributed dependencies' system and lexicon classification. E.g., in the sentences like *Petite, Alain la considérée intelligente* (see Fig. 9) is used the (rather frequent) coreference dependency *coref* and not the minimally opposed, but rare dependency *object − copredicative* (from *considérée* to *Petite*) used in (Mel'čuk and Iordanskaja, 2000).



(Small [fem.], Alain considered her intelligent [fem.])

Figure 9: Consecutive discontinuities

2. The grammar is rather intended for the development of French dependency treebanks, so the completeness criterion is prevaling over those of lower ambiguity and of more efficient parsing.

3. Basically, the grammar respects the fundamental principle of the dependency grammars (Kunze property): "words subordinate through the same dependency and belonging to the same grammatical category are substitutable"[5], but in the place of grammatical categories are considered the lexicon classes.

4. To reduce the ambiguity, a number of values are propagated through dependencies. For instance, some dependencies are parametrized by case. In French, only prepositions and pronouns mark for case. So we define the case

[5]See (Mel'čuk, 1988) and (Mel'čuk and Iordanskaja, 2000) for a weaker version.

VERBAL DEPENDENCIES			
Group	Governor (G)	Subordinate (D)	Relation
$PRED$	main verb	subject	predicative
AUX	auxiliary verb	past participle	auxiliary
$COPUL$	main verb	noun / adjective / circumstantial	copular
OBJ	verb / noun / adjective	complement	objectival
$AGENT$	past participle	preposition (e.g. *par*)	agentive
$CLIT$	verb	pre-position clitic	clitic
NEG	main verb	*ne*	negative
NEG	ne	*pas*, *plus*, etc.	composite negative
NEG	ne	restrictive *que*	retrictive negative
$CIRC$	verb	e.g., adverbs	circumstantial
$COORD$	verb	verb	verb coordination
$CLAUS$	rel. pronoun / verb	verb	clausal
NOMINAL DEPENDENCIES			
DET	noun / adjective	determiner	determinative
$MODIF$	noun	adjective / past participle	modifier
$ATTR$	noun	preposition	attributive
$QUANT$	noun	numeral	quantitative
REL	noun	pronoun	relative
$COMPAR$	noun	junction / adj.	comparative
$COREF$	pronoun	noun	co-referential
$RESTRICT$	noun	*que* / adv.	restrictive
$APPOS$	noun	noun / adj.	appositive

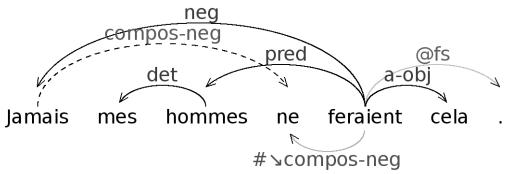
Tab. 2. A sample of verbal and nominal dependency relations

indirectly: a noun has case C if it can be replaced (in the same position) by a pronoun in case C without affecting the syntactic well-formedness. We distinguish five cases: a (accusative), d (dative), g (genitive), l (locative) and o (oblique, that of non-cliticizable complements). Respectively, we parametrize the objective dependency by the case of the subordinate complement, e.g. $a-obj$ (direct object), $d-obj$ (indirect object), etc. Moreover, when a word (e.g. an auxiliary verb) w serves as the host word for a pronoun in case C_1, the dependency from w to a subordinate word (e.g. a participle) is parametrized by C_1. The propagated parameters are used to prohibit to the subordinate to have the same case complements in their standard position (e.g. being subordinate through $aux-a-d$, the participle *donnée* in Fig. 2 cannot have complements).

Lexicon classes. As explained above, every class is defined, on the one hand, by a list of forms belonging to the class (the correspondence between the CDG classes and the Lefff categories is external with respect to the grammar), and on the other hand, by a set of RTE. Each RTE defines a set of CDG types possible for the lexical units in the list. In all, the French CDG, version 3.2 has about 3120 RTE. E.g., the RTE in Fig 5 is one of 32 RTE defining the class $Vt(F = fin, C = a)$.

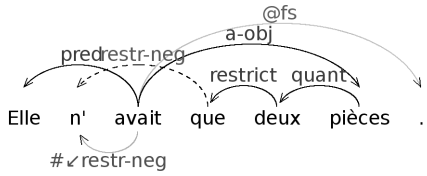
The grammar's lexicon includes four families of verbal classes: auxiliary verbs $Vaux$ (*avoir*, *être*), copulas $Vcopul$ (e.g. *être*, *devenir*), light verbs $Vlight$ (e.g. $faire$, *laisser*) and significant verbs V. Every family has four subfamilies corresponding to verb forms: $F = fin$ (finite formes), $F = pz, T = pres$ (present participle), $F = pz, T = past$ (past participle) and $F = inf$ (infinitive). Finally, the significant verbs are classified by their `government patterns`, i.e. by the number of their complements and by the complements' case (e.g. $V2t(F = fin, C1 = a, C2 = d)$). Among the nominal and adjectival classes there are also those with genitive and dative arguments. The prepositional classes are opposed by the syntactic function of the prepositional phrase (e.g., *complement* (infinitival or not), *circumstantial*, *attribute*), and by case/role (e.g. *agent*). Finally, there is a complex class UT of unknown lexical units.

Inventory of dependencies. The main advantage of the dependency syntax is that it is very close to a semantic representation of sentences. At that, dependency relations are numerous. The dependency relations used in the grammar are broken down into 39 groups:15 verbal, 14 nominal, 4 prepositional and several others: of aggregation, expletive, emphatic, junction/punctuation and deictic. Some of them are shown in Table 2.



(Never my people would do this)

Figure 10: Negation in pre-position



(It [fem.] had only two rooms)

Figure 11: Negation in post-position

Some dependency grammars and parsers flatten and distort DS because they cannot express non-projective dependencies. Such dependencies being not an obstacle for CDG, the grammar Version 3.2 uses numerous non-projective dependencies. Let us see the example of negative dependencies (group NEG).

The negation in French consists of two parts: the (main) `categorematic` part (*pas*, *plus*, *jamais*, *que*, *aucun* etc.) and the `syncategorematic` part *ne*. We distinguish between the `restrictive` verbal negation with the categorematic part *que*, *aucun*, etc. and the binary verbal negation with the categorematic part other than *que*, *aucun*, etc. because the latter is related through dependencies only with the negated verb, whereas the former is related not only with the verb, but also with one of its complements. For both kinds of negation, the categorematic part may be found in pre- and post-position with respect to the verb (cf. DS in Fig. 10 and 11).

Evaluation. The French CDG Version 3.2 was used to create an experimental dependency treebank (`DTB`). Actually, this DTB contains about 1500 DS. It was created within three months with the help of the toolkit CDG Lab (Alfared et al., 2011). The analyzed sentences originated from heterogeneous sources: French grammar (Grevisse, 1993), literary prose (E.Zola, M.Proust, La Rochefoucauld), scientific prose, periodical press (corpus Le Monde (Abeillé et al., 2003)), blogs, publicity, spoken language. These sentences vary from very short and simple to extremely long and complex. Nearly 200 of them needed application of the bootstrapping procedure in order to complete the grammar (lexically or syntactically or both). 42.8% of DS in the constructed corpus are non-projective. Among the non-projective dependencies used in these DS the most frequent are not only the negative dependencies, but also the reflexive and the clitic dependencies, as well as some nominal non-projective dependencies (e.g. $coref$ and $dist-rel$). Other non-projective dependencies are less frequent but are used in regular constructions, e.g. $C-obg$, $modif$, $attr$ (of topicalized complements, modifiers and attributes), *expletive* (of parenthetical phrases) and many other.

Parser. The CDG of French is parsed with a special CDG-complete polynomial time symbolic parser rather adapted to the parallel development of the CDG of French and of DS corpora. It computes, after every grammar update, the scores of correctness of the grammar with respect to a DS corpus and also supports a semi-automatic analysis by consecutive approximations (see (Alfared et al., 2011) for more details). A higher-performing autonomous mixed stochastic-symbolic parser is under design.

Conclusion

The extended CGD prove to be well adapted for practical development of wide scope dependency grammars and of dependency treebanks. Due to their formalization through the extended type calculus, they allow to express voluminous sets of types using well-structured and succinct restricted regular type expressions, and at the same time are analyzed in a reasonable polynomial time. A specific Structural Bootstrapping Method based on a genericity order on RTE and supported by a set of appropriate and efficient tools allows to incrementally develop in a relatively short space of time large scale dependency grammars and dependency treebanks provably correct with respect to them.

References

A. Abeillé, L. Clément, and F. Toussenel. 2003. Building a treebank for french. In A. Abeillé, editor, Treebanks.

Ramadan Alfared, Denis Béchet, and Alexander Dikovsky. 2011. "C DG Lab": a Toolbox for Dependency Grammars and Dependency Treebanks Development. In Proc. of the Int. Conf. on Dependency Linguistics (Depling'2011), Barcelona, Spain.

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

Denis Béchet, Alexander Dikovsky, Annie Foret, and Erwan Moreau. 2004. On learning discontinuous dependencies from positive data. In Proc. of the 9th Intern. Conf. "Formal Grammar 2004" (FG 2004), pages 1–16, Nancy, France.

Denis Béchet, Alexander Dikovsky, and Annie Foret. 2010. Two models of learning iterated dependencies. In Proc. of the 15th Conference on Formal Grammar (FG 2010), LNCS, to appear, Copenhagen, Denmark. [online] `http://www.angl.hu-berlin.de/FG10/fg10\_list\_of\_papers`.

Denis Béchet, Alexander Dikovsky, and Annie Foret. 2011. On dispersed and choice iteration in incrementally learnable dependency types. In Proc. of the 6th Int. Conf. "Logical Aspects of Computational Linguistics" (LACL'2011), LNAI 6736, pages 80–95.

Norbert Bröker. 2000. Unordered and non-projective dependency grammars. Traitement Automatique des Langues (TAL), 41(1):245–272.

Ralf Debusmann, Denis Duchier, and Geert-Jan. M. Kruijff. 2001. Extensible dependency grammar: A new methodology. In Proc. of the COLING 2004 Workshop on Recent Advances in Dependency Grammar, Geneva.

Michael Dekhtyar and Alexander Dikovsky. 2008. Generalized categorial dependency grammars. In Trakhtenbrot/Festschrift, LNCS 4800, pages 230–255. Springer.

Michael Dekhtyar, Alexander Dikovsky, and Boris Karlov. 2010. Iterated dependencies and kleene iteration. In Proc. of the 15th Conference on Formal Grammar (FG 2010), LNCS, to appear, Copenhagen, Denmark. [online] `http://www.angl.hu-berlin.de/FG10/fg10\_list\_of\_papers`.

Alexander Dikovsky. 2004. Dependencies as categories. In "Recent Advances in Dependency Grammars". COLING'04 Workshop, pages 90–97.

Alexander Dikovsky. 2007. Multimodal categorial dependency grammars. In Proc. of the 12th Conference on Formal Grammar, pages 1–12, Dublin, Ireland.

Alexander Dikovsky. 2009. Towards wide coverage categorial dependency grammars. In Proc. of the ESSLLI'2009 Workshop on Parsing with Categorial Grammars. Book of Abstracts, Bordeaux, France.

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

Maurice Grevisse. 1993. Le bon usage. Grammaire française. Duculot.

Aravind K. Joshi, Vijay K. Shanker, and David J. Weir. 1991. The convergence of mildly context-sensitive grammar formalisms. In Foundational issues in natural language processing, pages 31–81, Cambridge, MA.

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

Boris N. Karlov. 2008. Normal forms and automata for categorial dependency grammars. Vestnik Tverskogo Gosudarstvennogo Universiteta (Annals of Tver State University). Series: Applied Mathematics, 35 (95):23–43. (in Russ.).

J. Lambek. 1961. On the calculus of syntactic types. In Roman Jakobson, editor, Structure of languages and its mathematical aspects, pages 166–178. American Mathematical Society, Providence RI.

J. Lambek. 1999. Type grammars revisited. In Alain Lecomte, François Lamarche, and Guy Perrier, editors, Logical aspects of computational linguistics: Second International Conference, LACL '97, Nancy, France, September 22–24, 1997; selected papers, volume 1582. Springer-Verlag.

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

I. Mel'čuk and L. Iordanskaja. 2000. The notion of surface-syntactic relation revisited (valence-controlled surface-syntactic relations in french). In L. L. Iomdin and L. P. Krysin, editors, Slovo v tekste i v slovare. Sbornik statej k semidesjatiletiju Ju. D. Apresjana, pages 391–433. Jazyki russkoj kultury, Moskva.

I. Mel'čuk. 1988. Dependency Syntax. SUNY Press, Albany, NY.

B. Sagot. 2010. The lefff, a freely available and large-coverage morphological and syntactic lexicon for french. In Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10).

Vijay K. Shanker and David J. Weir. 1994. The equivalence of four extensions of context-free grammars. Mathematical Systems Theory, 27:511–545.

D. Sleator and D. Temperly. 1993. Parsing English with a Link Grammar. In Proc. IWPT'93, pages 277–291.

Mark Steedman. 1996. Surface structure and interpretation. MIT Press, Cambridge,Massachusetts.

Karel van den Eynde and Piet Mertens. 2003. La valence: l'approche pronominale et son application au lexique verbal. Journal of French Language Studies, 13:63–104.