"CDG Lab": an Integrated Environment for Categorial Dependency Grammar and Dependency Treebank Development

Denis BÉCHET [a], Alexander DIKOVSKY [a] and Ophélie LACROIX [a]
[a] *LINA CNRS UMR 6241, Université de Nantes, France*

Abstract. We present "CDG Lab", an integrated environment for development of dependency grammars and treebanks. It uses the Categorial Dependency Grammars (CDG) as a formal model of dependency grammars. CDG are very expressive. They generate unlimited dependency structures, are analyzed in polynomial time and are conservatively extendable by regular type expressions without loss of parsing efficiency. Due to these features, they are well adapted to definition of large scale grammars. CDG Lab supports the analysis of correctness of treebanks developed in parallel with evolving grammars.

Keywords. Categorial Dependency Grammar, Dependency Treebank

1. Introduction

There are two main technologies of automatic syntactic analysis of natural language: 1. grammatical parsing i.e symbolic parsing of a hand-crafted grammar belonging to a family of formal grammars disposing of a general purpose parser; 2. data-driven parsing, i.e. parsing with statistical parsers trained over annotated data. Both technologies need a large amount of expensive expert linguistic data. The hand-crafted wide coverage grammars are notoriously expensive and only very few of them had been successfully realized and applied to unrestricted material (cf. [5,18]). Besides this, they are prone to explosion of spurious ambiguity when parsed with general purpose parsers. On the other hand, training of statistical parsers needs voluminous high quality treebanks such as the Penn Treebank [17]. Training data of this size and quality are in fact as expensive as the hand-crafted grammars and also need a long-term hand work. Even if the results obtained in the statistical parsing during the last fifteen years are very encouraging, their quality and adequacy depends on those of the hand-crafted annotated data. This is a vital issue for dependency grammars which suffer from the shortage of high quality training data. The several existing dependency treebanks (DTB) such as the Prague Dependency Treebank of Czech [14], the TIGER treebank of German [6] or the Russian treebank [4] only partially solve the problem. First of all, they serve for particular languages. Secondly, even for these languages, the DTB use a particular inventory of dependency relations. At the same time, there is no consensus on such inventories. So the DTB are dependent on the choice of underlying syntactic theories, which makes problematic their reuse. The translation technologies (cf. [15]) consisting in acquisition

of dependency structures from high quality constituent structure treebanks also do not resolve the problem because, for technical reasons, they often flatten the genuine dependency structures and introduce into them multiple distortions. For all these reasons, there is a need in efficient and inexpensive methods and tools of development of wide coverage grammars and of training corpora.

Below we present "CDG Lab", an integrated environment supporting parallel development of wide scope dependency grammars and of DTB. It uses Categorial Dependency Grammars (`CDG`) as a formal model of dependency grammars. The CDG, a class of first-order type categorial grammars generating unlimited dependency structures (`DS`) and parsed in polynomial time, were introduced in [10]. Since then, they were intensively studied (e.g., see [7,3,8,9,2]). In [13] (this volume), one may see their definition, several examples of derivations and generated dependency structures and comments on their expressive power.

CDG in itself are formal grammars not adapted for practical use, but in [13] is defined their extension specially designed for definition of hand-crafted wide scope dependency grammars. The extended CDG is a kind of a simple meta-grammar for CDG extending the original CDG by regular type expressions (`RTE`). At that, they are also parsed in polynomial time. Moreover, in [13] is described and illustrated a method of incremental bootstrapping of large scale grammars from dependency structures.

In this paper we present and describe the functioning of "CDG Lab", an environment which includes a general purpose tabular parser of the extended CDG integrated into a friendly user interface supporting the bootstrapping method. CDG Lab makes it possible to analyse sentences generated by any extended CDG not only in the autonomous mode, but also interactively, reacting to user's negative or positive annotations on individual dependencies and lexical units. From these annotations, the parser computes the best (in a sense) next approximation to the final output dependency structure. We show how this environment may be used for parallel interactive development of large scale extended CDG and of DTB provably compatible with the grammar. The CDG Lab was successfully applied to the development of a wide coverage dependency grammar of French (CDGFr) and of several DTB. For this, the CDG Lab parser was integrated with rather representative open MS-dictionary of French Lefff 3.0 [19]. In few words, the methodology of bootstrapping of CDGFr and of interactive development of French DTB using CDG Lab is as follows. For every correct new French sentence, either CDGFr covers it, in which case the right dependency structure is computed by consecutive approximations and added to the DTB under construction, or an optimal extension of RTE is found, sufficient for the sentence to be analysed. In this way, both CDGFr and the DTB are developed in parallel and stay always compatible.

The CDG Lab parser is well performing with experimental ordinary and extended CDG, but with CDGFr it is apt to analyse only very short and simple French sentences. This is due to a very high ambiguity of this grammar principally meant for French DTB development. Nevertheless, CDG Lab will be completed by a standalone efficient parser of French using CDGFr. This mixed symbolic/statistical parser is based on the observation made while the practical work on the French DTB, that the right choice of the incoming dependency (we call it `head dependency`) is very predictive for the right category of the word. With right head dependency selection, the CDG Lab parser is capable of instantly analysing extremely long and complex sentences (the longest one numbers183 lexical units and punctuation marks). For this reason, the head dependency se-

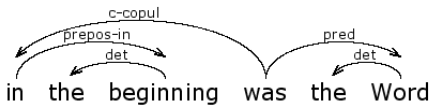


Figure 1. Projective DS

lection has become one of the modes of analysis with the CDG Lab parser. The future mixed standalone parser will use a statistical oracle defining the best head dependencies and trained on the developed DTB. Below we present promising results of the first experiments on several head dependency prediction methods.

The plan of this paper is as follows. In Section 2 we give a very short sketch of CDG and of their extension by RTE. Then the architecture and the main functionalities of "CDG Lab" are described in Section 3. And finally, Section 4 presents the the results of head dependency learning.

2. Categorial Dependency Grammars

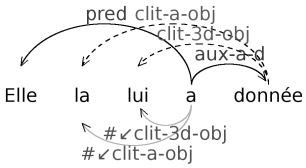
A formal definition of CDG may be found in [13]. Here we explain its intuitive meaning and illustrate it by examples.

CDG define `projective` DS (as in Fig. 1) i.e. DS in which dependencies do not cross, and also `discontinuous` DS, as in Fig. 2, in which some dependencies cross. In these graphs, the nodes correspond to the words of the sentence (their `precedence` order in the sentence is important) and the arcs represent the dependencies: named binary relations on words. Formally, a DS of a sentence x is a linearly ordered cycle-free graph with labelled arcs and the words of x as nodes. We consider connected DS with the root node. When in a DS D there is an arc $w_1 \stackrel{d}{\longrightarrow} w_2$, we say that d is a dependency between w_1 and w_2, w_1 is the `governor` and w_1 is `subordinate` to w_1 through d. E.g., in is subordinate to was in Fig. 1 and $donnée$ governs la through $clit{-}a{-}obj$ and lui through $clit{-}3d{-}obj$.

As all categorial grammars, the CDG are completely lexicalized and may be seen as assignments of categories (or types) to words in a dictionary W. CDG categories are expressions of the form

$$t = [l_1 \backslash l_2 \backslash \ldots \backslash H / \ldots / r_2 / r_1]^P.$$

A category assigned to a word $w \in W$ defines its dependencies in a rather straightforward way: its subcategories H, $l_1, l_2, \ldots$, $\ldots, r_2, r_1$ represent the claims for w to be related to other words through projective dependencies and P, called `potential` of t, defines all discontinuous dependencies of w. In particular, the `head` subcategory H, claims that w should be subordinate to a word through dependency H. When w should be the root of a DS, $H = S$ (S is a special symbol called `axiom`). The `left` subcategories $l_1, l_2, \ldots$ define the left projective dependencies of w (i.e. the dependencies through which w governs the words occurring in the sentence on its left). The `right` subcategories $\ldots, r_2, r_1$ define the rifght projective dependencies of w. For instance, the projective DS in Fig. 1 is uniquely defined by the assignment:
$in \mapsto [c{-}copul/prepos{-}in]$, $the \mapsto [det]$,



(*fr.* $^*she\ it[fem.]\ to\ him\ has\ given$)

Figure 2. Non-projective DS

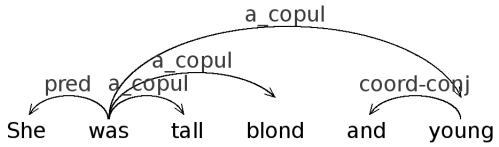


Figure 3. Iterated dependency

$Word \mapsto [det\backslash pred]$, $beginning \mapsto$
$[det\backslash prepos{-}in]$, $was \mapsto [c{-}copul\backslash S/pred]$.

The order of left and right subcategories determines the order of the subordinate words: for two words w_i, w_j preceding to w and subordinate to w through dependencies l_i and l_j respectively, $i < j$ iff w_i is closer to w than w_j (similar for right).

Left and right subcategories may also be `iterated`. The iterated subcategories define `repeatable` dependencies. E.g., $l_i = d^*$ means that w may have on it left $0, 1, 2, \ldots$ occurrences of words subordinate to it through dependency d. We also use `optional` dependencies $l_i = d?$. Assignment of category $[d?\backslash\alpha]$ is equivalent to assignment of $[d\backslash\alpha]$ and $[\alpha]$. E.g., the DS in Fig. 3 is defined by the assignment:
$she \mapsto [pred]$, $was \mapsto [pred\backslash S/a\_copul^*]$,
$tall$, $blond$, $young \mapsto [coord{-}conj?\backslash a\_copul]$,
$and \mapsto [coord{-}conj]$.

The potential P is the concatenation of so called `polarized valencies` of w. The polarized valencies are of four kinds: $\swarrow v$, $\searrow v$ (`negative`) and $\nwarrow v$, $\nearrow v$ `positive`. E.g., if a word w has valency $\swarrow v$, this 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 discontinuous dependency v. The order of polarized valencies in the potential P is irrelevant (so one may choose a standard lexicographic order). For instance, in the DS in Fig. 2, the potential $\nwarrow clit-a-obj \nwarrow clit-3d-obj$ of the participle *donnée* 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 their dual valencies with those of the participle defines two discontinuous dependencies between the participle and its cliticized complements.

Finally, in order to define for a word subordinate through a discontinuous dependency its adjacency to a `host` word, the CDG categories use expressions $\#(\swarrow v)$, $\#(\searrow v)$ called `anchor` subcategories (or just `anchors`), in which v is a valency name. In particular, so that a word w_0 would be a (left position) host for a word w_1, subordinate to some other word through a negative discontinuous dependency v, the anchor $\#(Av)$ (A being one of the four orientations) should be the head subcategory of the category of w_1 and at the same time a left subcategory of the category of w_0. Similar for right position. E.g., the DS in Fig. 2 is defined by the following assignment:
$elle \mapsto [pred]$,
$la \mapsto [\#(\swarrow clit{-}a{-}obj)]^{\swarrow clit\text{–}a\text{–}obj}$,
$lui \mapsto [\#(\swarrow clit{-}3d{-}obj)]^{\swarrow clit\text{–}3d\text{–}obj}$,
$donnée \mapsto [aux{-}a{-}d]^{\nwarrow clit\text{–}a\text{–}obj \nwarrow clit\text{–}3d\text{–}obj}$,
$a \mapsto [\#(\swarrow clit{-}3d{-}obj)\backslash\#(\swarrow clit{-}a{-}obj)\backslash pred$
$\backslash S/aux{-}a{-}d]$.
Due to the anchors $\#(\swarrow clit{-}3d{-}obj)$, $\#(\swarrow clit{-}a{-}obj)$ in the subcategory of the auxiliary verb a (has), it serves as the host verb for both clitics and also defines their precedence order.

Extended CDG. CDG are a theoretical model not adapted to 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 strong 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. [12] proposes an extension of CDG adapted to wide coverage grammars.

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 extended CDG use flat (i.e. bounded depth) `regular type expressions` (`RTE`) we describe below. 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: $(C_1|\ldots|C_k).\ (C) = C$.
Optional choice: $(C_1|\ldots|C_k)?.\ (C)? = C?$.
Iteration: $(C_1|\ldots|C_k)^*.\ (C)^* = C^*$.
Distributed 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: $\{\alpha_1, B, \alpha_2\}[\alpha\backslash H/\beta]^P$.

As the original CDG, the extended CDG are formalized by a calculus wich has special rules for every kind of RTE (see [12,2,1]). Below we informally explain the intuitive meaning of the operators used in the RTE.

The choice unites several alternative types into one. For instance, assigning the type $[(C_1|C_2)\backslash\beta]^P$ is equivalent to assign two types $[C_1\backslash\beta]^P$ and $[C_2\backslash\beta]^P$. An element (a

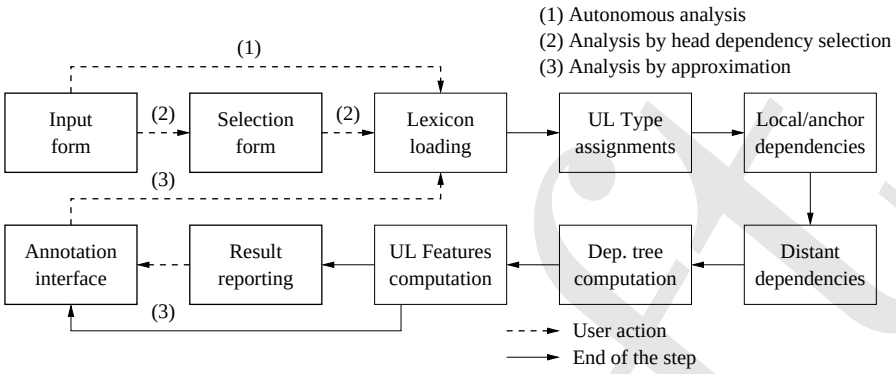


Figure 4. Architecture of the CDG Lab Parser

word or a phrase) to which is assigned RTE $[(C_1|\dots|C_k)^*\backslash\beta]^P$ allows any sequence of elements of the alternative types $C_1,\dots,C_k$ immediately on its left. On the contrary, assignment to an element of the type $[\{\alpha_1,B,\alpha_2\}\backslash\alpha\backslash H/\beta]^P$ means that an element of type B must be present in some left position. E.g. the assignments $w_0 \mapsto [\{d\}\backslash b\backslash a\backslash S]$, $w_1 \mapsto [a]$, $w_2 \mapsto [b]$, $w_3 \mapsto [d]$ define DS of sentences: $w_3w_1w_2w_0$, $w_1w_3w_2w_0$, $w_1w_2w_3w_0$ in which $w_0 \xrightarrow{d} w_3$, $w_0 \xrightarrow{a} w_1$ and $w_0 \xrightarrow{b} w_2$. The right distributed RTE is similar. The two-way distributed RTE claims that an element of type B were found in some left or right position.

The RTE and the classes do not extend the expressive power of CDG. At the same time, they dramatically reduce the grammar size. Of course, unfolding of an extended CDG may exponentially blow up its size. However, due to the extended type calculus they can be parsed directly, without unfolding. In fact, the polynomial time parsing algorithm of [8] can be adapted to the extended CDG.

3. CDG Lab

CDG Lab is an integrated environment supporting parsing with extended CDG, development and maintainance of dependency treebanks (`DTB`) and development and test of large scale extended CDG. The core element of CDG Lab is the parser of the extended CDG implemented in Steel Bank Common Lisp. Recently was issued its version 3.3.2 (below we will call this parser `CDG Lab Parser`).

All input and output data of CDG Lab Parser are XML-structures. It may analyse sentences, text corpora and DTB either with an extended CDG integrated with an external morpho-syntactic dictionary (`lexical base grammar`) or only with the internal grammar lexicon (`text grammar`). For instance, for French is developed a large scale extended CDG [13]. Its version 3.3 (called below "CDGFr Text") is integrated with the open MS-dictionary of French Lefff 3.0 [19] containing 536,375 lexical units (`LU`). In CDG Lab, Lefff is kept in object-relational database PostgreSQL. A correspondence between the classes of the CDGFr Text and the categories of Lefff is implemented through several hundreds of SQL queries. The integral grammar is called below "CDGFr LB".

Figure 5. Query Form (screenshot)

Depending on the user's command, CDG Lab Parser may be used in one of four modes:

- Analysis by head dependency selection
- Analysis by approximations
- DS analysis
- Autonomous analysis

Fig. 4 shows a scheme of functioning of CDG Lab Parser in these modes. Sentences are introduced through the input form (see Fig. 5). Through this form, the user may set various parameters, e.g. the maximal parsing time, the maximal number of DS to return, a graphical representation of DS, a language register (corresponding to specific choices of discontinuous dependencies common to official documents or to scientific or literary prose, to periodicals or to the spoken language), etc. The input sentence is lexically analysed. Composite forms are decomposed into separate tokens, as in the case of $l'homme$ (the man), which is segmented into three tokens: l, $'$ and $homme$, for which are found in the lexicon all possible lemmas. In this example, the association is ambiguous: l' may be a clitic or a determiner. All possible variants of composite LU (in particular, complex numbers and names recognized through regular expressions, multi-word LU, such as $\grave{a}\ la$ (of the kind), $\grave{a}\ travers$ (through) etc.) are also detected and unknown terms are identified.

The transitions to and from head dependency selection form are followed only in the mode of Analysis by head dependency selection. Functioning in the mode of Analysis by approximations is iterative. It skips result reporting step and goes directly from

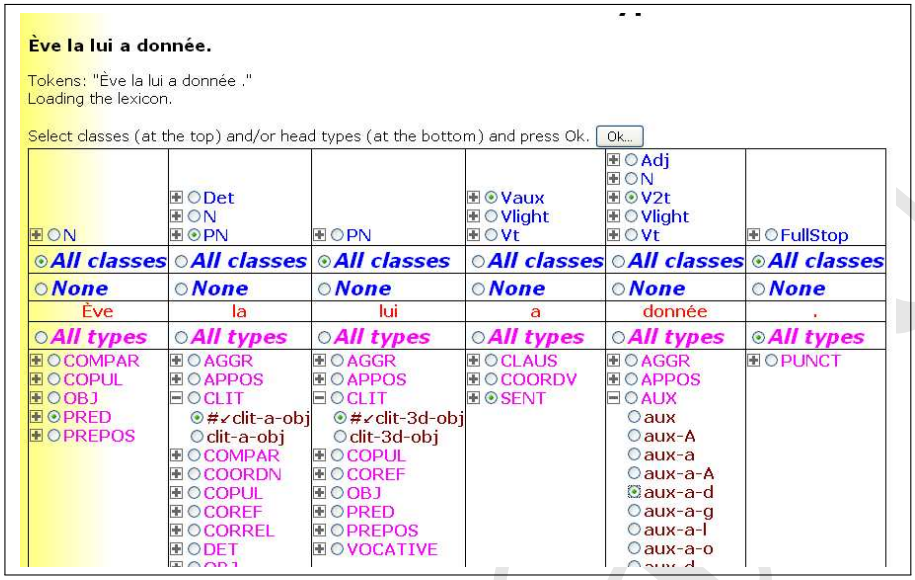


Figure 6. Selection Form (screenshot)

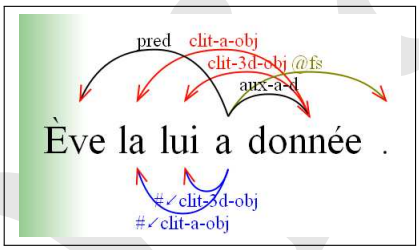


Figure 7. Resulting DS (screenshot)

annotation interface to lexicon loading. Other transitions are common for all modes. So we comment them in the head dependency selection mode.

3.1. Analysis by Head Dependency Selection.

In this mode `selection form` (see Fig. 6) is proposed, in which the user may select the proper composite LU (if and when several possibilities are detected) and for every LU, to select one of possible classes and one of possible dependency relation groups (or of their elements). The selected dependency is nothing but the head subtype of the possible types of the LU. The latter selection is in fact decisive. It corresponds to the strong constraint that the LU is subordinate through the selected dependency (or limits the choice of such dependencies to the elements of the selected group).

This selection drastically limits the search-space. As it concerns the CDG Lab Parser and the CDGFr LB, it reduces the number of possible analyses by two-three orders of magnitude, i.e. in the place of a thousand of possible DS only about ten are found, most

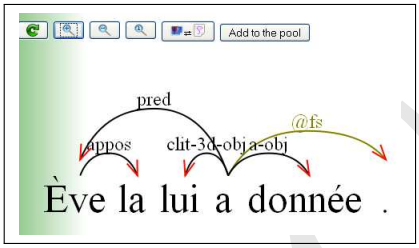


Figure 8. Incorrect DS (screenshot)

often differing between them in positions of repeatable dependencies (such as $modif$ (`modifier`), $attr$ (`attribute`) or $circ$ (`circumstantial`). E.g., in this example the genuine DS will be immediately found due to this selection (see Fig. 7).

Then the sentence's workspace (`WS`) is created, which is an XML-structure representing the subgrammar corresponding to all detected LU to which are affected the classes, the types and the features values compatible with the pre-selection. After this the steps common to all modes follow. First, all possible projective (and anchor) dependencies are computed and registered in the triangular matrix. This computation is done by an CKY-like algorithm adapted to extended CDG. In the resulting matrix (more precisely, in the submatrix in which the axiom S can be proved), all possible pairings of dual valencies providing discontinuous dependencies are independently computed and registered. It should be noted that the pairing principle may be chosen for a CDG and for a particular discontinuous dependency. By default, this is the **FA** rule: the closest available dual polarized valencies are paired. But in rare exceptional situations, such as that of unlimited cross-serial dependencies in subordinate clauses in Dutch, a different rule **FC** (`first cross`) defined in [11] may be used. This independence of computations of projective and discontinuous dependencies is founded on the fundamental `projection independence property` of CDG proved for the rule **FA** in [7,8] and for rule **FC** in [11]. Till the end of this step the parsing algorithm is polynomial. The resulting triangular matrix is in fact a packed chart from which it is possible to enumerate all possible DS of the sentence. Given that the number of these DS may be exponential with respect to the size of the matrix, the next step is exponential in space in the worst case. In this step, the DS are generated from the matrix in a certain order and the feature values are assigned to LU in every generated DS. Finally, the parser generates the HTML report page, which includes various useful statistics. An XML structure representation of every DS including all necessary information, in particular the CDG classes and the feature values is also generated and saved to be used by other programs.

3.2. Analysis by Approximations.

This important mode represents another user-guided strategy of parsing. It allows to find the needed DS starting from any obtained DS by consecutive approximations computed from user's annotations in the DS. There are three possible annotations of dependency relations: `positive`, `negative` and `neutral`. The positively annotated dependencies are those adequate. They will be kept during the whole sequence of approximations (if not discarded). The neutrally annotated dependencies are kept till they are compatible with the positively annotated ones. The negatively annotated dependencies are to be

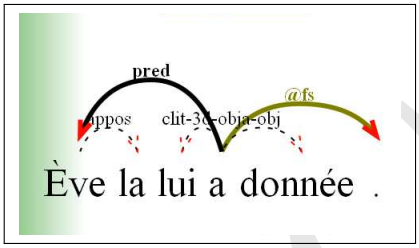


Figure 9. First annotated DS (screenshot)

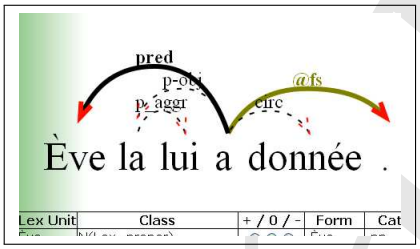


Figure 10. Next approximation (screenshot)

eliminated from the DS. When used in this mode, the Parser computes for every DS the total number of positively annotated dependencies and that of negatively annotated dependencies. The obtained DS are sorted first by negative annotations' weight (the fewer negative annotations the better) then by the positive annotations' weight (the more positive annotations the better).

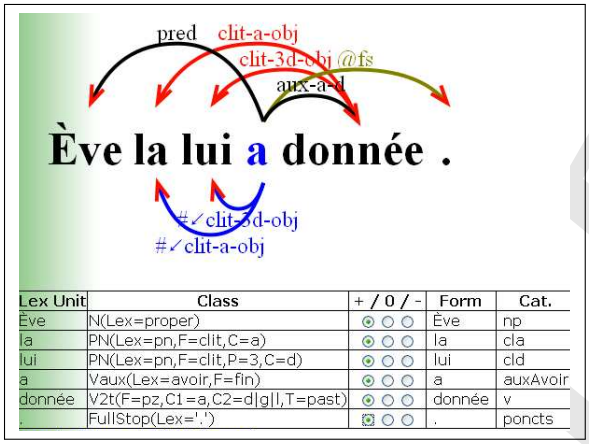
Suppose, that the approximations start from the (partially incorrect) DS of the sentence *Ève la lui a donnée* (*Eve it*$[fem.]$ *to him has given*) shown in Fig. 8. There are only two correct dependencies in this DS: the predicative one: $pred$ and the punctuation dependency $@fs$. We annotate both positively (this annotation being displayed by boldface arcs). The other three dependencies $appos$, $clit{-}3d{-}obj$ and $a{-}obj$ are erroneous. We annotate them as negative (which is displayed by dashed arcs). So we obtain the first annotated DS shown in Fig. 9.

From this annotated DS, the Parser computes the next approximation shown in Fig. 10, which is also incorrect. It has the same two correct dependencies and three other incorrect: $p{-}obj$, $p{-}aggr$ and $circ$. We annotate the three as negative, as it is shown in Fig. 10.

From this annotated DS, the parser finally computes the right one shown in Fig. 11. Not only this final approximation is correct, but it is also annotated as such. This difference is very important for the other mode of use of CDG Lab Parser, that of DS analysis.

3.3. DS, DTB and Grammar Analysis.

The CDG Lab Parser considers every dependency of a DS as annotated: The default value of the annotation of a dependency is `neutral`. Moreover, not only the dependencies, but also the LU may be annotated. The LU may have only two annotations: `positive` and `neutral`. Annotating a LU w positively is equivalent to positively annotate all de-

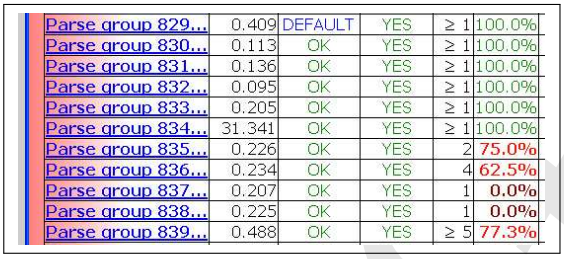


Lex Unit	Class	+ / 0 / -	Form	Cat.
Ève	N(Lex=proper)	⊙ ○ ○	Ève	np
la	PN(Lex=pn,F=clit,C=a)	⊙ ○ ○	la	cla
lui	PN(Lex=pn,F=clit,P=3,C=d)	⊙ ○ ○	lui	cld
a	Vaux(Lex=avoir,F=fin)	⊙ ○ ○	a	auxAvoir
donnée	V2t(F=pz,C1=a,C2=d|g|l,T=past)	⊙ ○ ○	donnée	v
.	FullStop(Lex='.')	⊡ ○ ○	.	poncts

Figure 11. Final approximation (screenshot)

pendencies in the sub-structure with the root w. This is seen in Fig 11, were the positive annotation of the root (displayed in a contrasting color) implies the positive annotations of dependencies (displayed in boldface). More than that, the class and the feature values assigned to every LU in DS may also be annotated as `positive`, `negative` or `neutral`. In the fragment of the class/feature table shown in Fig 11, it is seen that not only the dependencies, but also the class/feature assignments for its LU are all annotated as positive. So this analysis is 100% correct. It is using this integral annotation weight, that CDG Lab Parser evaluates the DS. Now, for two DS of the same sentence it is possible to measure the difference of their weights. This simple measure turns out to be an efficient means of analysis of DTB and of the CDG used while their development. Every sentence processed by CDG Lab Parser using CDGFr LB obtains its `status`. The status includes the analysis result ('NO', when there are no parses, 'YES' otherwise) and for a parsable sentence, also the maximal number of returned DS and the difference (in percents) of annotation weights (of dependencies and of LU) between the best obtained DS and the one present in the DTB (if any) before this sentence processing. When the grammar is updated, the DS of sentences in the DTB become potentially irrelevant. For this case, CDG Lab Parser has a special function of `re-parsing` of a DTB, which computes the difference between the DS before and after update, comparing their statuses. The user may choose between keeping or not the same head subtypes while re-parsing. Using this function, one may easily find all sentences to be revised.

Moreover, this test applies to the grammar itself. The CDGFr Text was created using Structural Bootstrapping Method [13], a method specific to the extended CDG and consisting in an `incremental` transformation of DS of a sample of sentences σ into an extended CDG $G(\sigma)$ generating these sentences. The incrementality is interpreted in the strong sense: $\Delta(G(\sigma)) \subseteq \Delta(G(\sigma \cup \{s\}))$ for every new sentence s. The bootstrapping of CDGFr Text was basically incremental in this sense, except three important revisions which were not. Taking in mind the size and the complexity of this grammar (it consists of more than 3800 RTE distributed between 185 lexicon classes, has 84 projective and 20 discontinuous dependencies), it was a very hard task to find all sentences in the sample wrongly analysed using the updated grammar. Indeed, to find them, it was necessary



Parse group 829...	0.409	DEFAULT	YES	≥ 1	100.0%
Parse group 830...	0.113	OK	YES	≥ 1	100.0%
Parse group 831...	0.136	OK	YES	≥ 1	100.0%
Parse group 832...	0.095	OK	YES	≥ 1	100.0%
Parse group 833...	0.205	OK	YES	≥ 1	100.0%
Parse group 834...	31.341	OK	YES	≥ 1	100.0%
Parse group 835...	0.226	OK	YES	2	75.0%
Parse group 836...	0.234	OK	YES	4	62.5%
Parse group 837...	0.207	OK	YES	1	0.0%
Parse group 838...	0.225	OK	YES	1	0.0%
Parse group 839...	0.488	OK	YES	≥ 5	77.3%

Figure 12. Re-parse results (screenshot)

to look through thousands of DS of hundreds of sentences in order to find linguistically adequate DS (the simple existence of a generated DS is of course not sufficient). The situation has completely changed after the implementation of CDG Lab Parser. Indeed, now all sentences in the sample are initially annotated. The procedure of re-parsing of the sample finds all inconsistencies in several minutes.

In Fig. 12 we show a fragment of the table representing the results of re-parsing applied to a DTB. In this table:
- the first column is the reference to the DS of a sentence,
- the second column shows the (folded) characteristics of parsing complexity,
- in the third column, 'OK' means that all LU of the sentence are present in the grammar lexicon and 'DEFAULT' means that there is at least one LU absent in the lexicon and replaced by the default unit,
- in the fourth column, 'YES' means that the re-parsing was successful in the sense that a successful analysis was found and 'NO' means the contrary,
- in the fifth column the maximal number of DS requested for re-parsing is given ($\geq k$ means that there are more than the k requested DS),
- the sixth column shows the part (in percents) of annotation status coincidence of the DS before and after the update (so it is 0% for new sentences).

3.4. Autonomous Analysis.

This is the mode of non-user-guided analysis. CDG Lab Parser may be used as a general purpose parser for the extended CDG. In CDG Lab there is a possibility to upload one's own grammar or to introduce it through the Sandbox. For itself, CDG Lab Parser is rather efficient. Even when used with the CDGFr LB, it is capable to analyse half a thousand of sentences of various complexity in about 10 minutes. The problem is that it generates the DS not in the order of their adequacy. With ambiguous CDG, such as the CDGFr LB, it generates hundreds of spurious structures per sentence. So for very long and complex sentences, it is practically impossible to know whether a reasonable DS was computed. This is why we consider CDG Lab Parser as a tool of development of DTB using head dependency selection, approximations and re-parsing.

3.4.1. Desambiguation by Head Dependency Selection

The analysis by head dependency selection (through selection forms), the automatic head dependency selection used when re-parsing (based on dependency annotations) and the automatic head dependency tagging presented in Section 4 lower impressively the anal-

ysis ambiguity and increase the performance of the parser. To illustrate this result, the section presents an evaluation of the automatic head dependency tagging of Section 4 in term of ambiguity and efficiency when compared to the autonomous analysis.

As we have stated above, the performance of the parser in the autonomous mode is very low. In this mode it produces too many spurious DS and, still worse, too many parses fail. There are two reasons for the latter:

- selected head dependencies contradict the assigned CDGFr categories, so the analysis fails;
- the analysis is stopped because of time limits (typically, for long and/or complex sentences).

Respectively, we use the following evaluation criteria: the numbers of completed and uncompleted parses, the sentence's length and the average parsing time. In Table 1, we show the values of these measures in our experiments.

Autonomous analysis

Nb of choices	Number of sentences						LU/sentence			sec./sentence
	CP	(%)	UP-C	(%)	UP-T	(%)	CP	UP-C	UP-T	CP+UP
-	1150	(41.4)	3	(00.1)	1625	(58.5)	7.2	7.3	17.6	6.74

Analysis by automatic head dependency selection

Nb of choices	Number of sentences						LU/sentence			sec./sentence
	CP	(%)	UP-C	(%)	UP-T	(%)	CP	UP-C	UP-T	CP+UP
1	1805	(65.0)	969	(34.9)	4	(00.1)	11.5	16.5	52.5	0.10
1 to 2	2054	(73.9)	718	(25.8)	6	(00.2)	11.6	17.7	56.1	0.13
1 to 5	2335	(84.1)	438	(15.8)	5	(00.2)	12.0	20.0	49.4	0.16
1 to 10	2505	(90.2)	262	(09.4)	11	(00.4)	12.2	22.5	42.8	0.22

Table 1. *Autonomous analysis vs. analysis by automatic head dependency selection. The number of completed parses (CP), the number of uncompleted parses where head dependency selection is not consistent with grammar's rules (UP-C), and where parsing is cancelled (UP-T) are compared. The parsing time is limited to 10 seconds. The average number of lexical units (LU) per sentence in all the three cases above and also the average parsing time (in seconds) for every sentence are shown.*

In the course of analysis in autonomous mode, one cannot restrict the number of possible head dependencies per LU. In contrast, the automatic head dependency selection sets the maximal number of head dependency choices to 10. This hard limit lowers both the ambiguity and the parsing time (however the parsing time may somewhat grow with the growing number of alternative dependency head choices). It also allows to analyse longer sentences. On the other hand, due to this limit some correct sentences may not obtain a DS because the permitted choices may all conflict with the grammar. But in total we have observed a substantial growth of the number of completed parses. The best result is obtained with 1 to 10 pre-selected head dependencies and gives 2505 completed parses among 2778 (90.2%). At that 1999 parses give the best (i.e. the original) DS.

3.5. DTB Development.

The annotation based development of DTB in CDG Lab leads to a notable change in the point of view on the quality of treebanks. It is now the grammar, implementing a set of linguistic subjective expert knowledge, which will serve as the "golden standard". As to the DTB, they should all be correct with respect to the grammar and should be tested for correctness after every non-incremental grammar update. By definition, the incremental grammar updates preserve correctness of DS.

Besides the means based on DS annotation, CDG Lab has rather standard means for creation and updates of DTB and for search of DS by projective and discontinuous dependency names and by LU in the sentences.

3.6. Grammar Development.

Besides the described above general purpose means supporting non-incremental grammar updates, CDG Lab has some means specific for CDG of French integrated with Lefff 3.0. In particular, it has several functions for completion of the lexicon of these CDG. Basically, there are two problems: the first is to automatically complete the lexicon by all forms of a missing word (this concerns mainly the verbs), the second is to compute the argument frame of a missing word from that of a present word. For the former problem, CDG Lab has several functions based on updates of the lexicon of the CDGFr Text. The latter problem concerns the deverbals. The work on completions of this kind is in progress.

4. Automatic Head Dependency Selection

One of our main goals is to integrate into CDG Lab an efficient mixed symbolic/statistical parser (`Mixed Parser`). Basically, it will follow the line of analysis in the head dependency selection mode, with the fundamental difference that the selection will be not user-guided, but automatic. For this, the Mixed Parser will use an oracle trained on the previously created DS corpora. From the conventional learning perspective, this statistical oracle may be seen as a procedure tagging the text with head dependency names/groups. In this preliminary phase, we simplify this task using the right delimitation of the composite LU and of their grammatical classes in CDGFr. For training the oracle we use the CRF (Conditional Random Fields) [20], in particular its implementation by Wapiti [16].

In the experiments whose results we present below, the LU are to be tagged with one of the 117 head dependency names used in the current CDGFr. For that we use a sort of subcategorization given by the grammatical CDG classes of LU. We choose 86 of them used in the CDGFr. Every LU in the input text is tagged by one of them.

The training is performed on a corpus of 2778 sentences, divided in 10 parts. The experiment consists of 10 tests, each consisting of two phases: training and tagging. Every training phase is performed on 90% of the sentences, whereas the tagging phase is perfomed on the resting 10%. The 10 best sequences of tags (head dependency names) are computed for every sentence. It happens that the sequences are rather similar: there are very rarely 10 different head dependencies for one LU. In Table 2, the evaluations of the tagger for 1, 2, 5 and 10 best assigned tags are shown.

	Accuracy
Top 1	91.1
Top 2	93.2
Top 5	95.5
Top 10	96.6

Table 2. *Evaluation (accuracy) of head dependency tagging using CRF. The results represent only the rates of the correctly tagged LU among the 1, 1 to 2, 1 to 5 and 1 to 10 assigned tags.*

As a result, we obtain the corpus in which every LU is tagged with 1 to 10 head dependency names. The corpus is analysed by the CDG Lab parser using these tags. Then the best DS is selected for every successfully analysed sentence. *Best* means the nearest with respect to the DS associated to the sentence in the corpus. The *nearest* means having the maximum of correctly labelled dependencies as compared to that in the corpus. Currently, the output DS are not sorted by this distance. In Table 3, the labelled and unlabelled attachment scores are shown. These scores reflect the precision of the tagger on the parsed sentences [1]. We also show the precision for the discontinuous dependencies in separate columns.

Nb of heads	All dependencies		Discontinuous dependencies	
	LAS	UAS	LAS	UAS
1	93.7	96.7	92.4	93.7
1 to 2	95.1	97.3	94.3	95.5
1 to 5	96.2	97.8	94.4	95.5
1 to 10	96.4	97.9	94.5	95.4

Table 3. *Tagging precision. Evaluation wrt the best (nearest to the original) DS produced by the parser for each parsed sentence. LAS (labelled attachment score) is the quota of the LU correctly attached to their governors with the correct label. UAS (unlabelled attachment score) is the quota of the LU correctly attached to their governor.*

The more there are head dependency choices, the greater is the chance to find the correct dependency. This explains why the results are slightly better where there are more choices. As it concerns the discontinuous dependencies, one may see that the precision is not as high as in the general case. On the whole, the head dependency tagger ensures a reasonably high rate of correct dependencies (both projective and discontinuous). Importantly, it also significantly affects ambiguity and performance of syntactic analysis. Subsection 3.4.1 gives the results of evaluation of this influence.

5. Conclusion

CDG Lab integrates into a unified friendly user interface several efficient means of incremental parallel development of wide scope categorial dependency grammars and of

[1] Not all sentences are parsed. Some analyses fail without producing a DS. See Subsection 3.4.1 for more details.

dependency treebanks correct with respect to the grammars. These means were successfully tested during the development of a wide scope categorial dependency grammar of French and of several dependency treebanks numbering more than 3000 DS. Some of these means are general purpose. E.g., the annotation weight difference test applies to any kind of structural incremental development based on expert annotations. Some others, such as head dependency selection and consecutive approximations, may be used with other classes of dependency grammars and implemented in tabular dependency grammar parsers. Some means are specific to the CDG Lab Parser and to the CDG of French integrated with Lefff. Several important means of this environment are still under construction, but even this experimental version has proved its high efficiency.

References

[1] D. Béchet, A. Dikovsky, and A. Foret. On dispersed and choice iteration in incrementally learnable dependency types. In S. Pogodalla and J.-P. Prost, editors, *Logical Aspects of Computational Linguistics, 6th International Conference, LACL 2011, Montpellier, France, June 29 – July 1, 2011. Proceedings*, volume 6736 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 80–95. Springer-Verlag, 2011.

[2] D. Béchet, A. Dikovsky, and A. Foret. Two models of learning iterated dependencies. In P. de Groote and M.-J. Nederhof, editors, *Formal Grammar, 15th and 16th International Conferences, FG 2010, Copenhagen, Denmark, August 2010, FG 2011, Lubljana, Slovenia, August 2011, Revised Selected Papers*, volume 7395 of *Lecture Notes in Computer Science (LNCS)*, pages 17–32. Springer-Verlag, 2012.

[3] D. Béchet, A. Dikovsky, A. Foret, and E. Moreau. On learning discontinuous dependencies from positive data. In P. Monachesi, editor, *Proceedings of the 9th International Conference on Formal Grammar (FGNancy), Nancy, France, 7-8 August 2004*, pages 1–16, 2004.

[4] I. Boguslavsky, S. Grigorieva, N. Grigoriev, L. Kreidlin, and N. Frid. Dependency treebank for Russian: Concept, tools, types of information. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING'2000)*, 2000.

[5] G. Bouma, G. van Noord, and R. Malouf. Alpino: Wide-coverage computational analysis of Dutch. In *Proceedings of the Conference on Computational Linguistics in the Netherlands*, pages 45–59, 2000.

[6] S. Brants and S. Hansen. Developments in the TIGER annotation scheme and their realization in the corpus. In *Proceedings of the Second International Conference on Language Ressources and Evaluation (LREC'02)*, 2002.

[7] M. Dekhtyar and A. Dikovsky. Categorial dependency grammars. In M. Moortgat and V. Prince, editors, *Proceedings of the International Conference on Categorial Grammars (CG2004), Montpellier, France, June 2004*, pages 76–91, 2004.

[8] M. Dekhtyar and A. Dikovsky. Generalized categorial dependency grammars. In *Trakhtenbrot/Festschrift*, volume 4800 of *Lecture Notes in Computer Science (LNCS)*, pages 230–255. Springer-Verlag, 2008.

[9] M. Dekhtyar, A. Dikovsky, and B. Karlov. Iterated dependencies and Kleene iteration. In P. de Groote and M.-J. Nederhof, editors, *Formal Grammar, 15th and 16th International Conferences, FG 2010, Copenhagen, Denmark, August 2010, FG 2011, Lubljana, Slovenia, August 2011, Revised Selected Papers*, volume 7395 of *Lecture Notes in Computer Science (LNCS)*, pages 66–81. Springer-Verlag, 2012.

[10] A. Dikovsky. Dependencies as categories. In *Proceedings of the COLING'04 Workshop on Recent Advances in Dependency Grammars*, pages 90–97, 2004.

[11] A. Dikovsky. Multimodal categorial dependency grammars. In *Proceedings of the 12th Conference on Formal Grammar, Dublin, Ireland*, pages 1–12, 2007.

[12] A. Dikovsky. Towards wide coverage categorial dependency grammars. In *Proceedings of the ESSLLI'2009 Workshop on Parsing with Categorial Grammars. Book of Abstracts, Bordeaux, France*, 2009.

[13] A. Dikovsky. Structural bootstrapping of large scale categorial dependency grammars. In *Computational Dependency Theory*. IOS Press, 2013. See also Proceedings of the International Conference on Dependency Linguistics (Depling'2011), Barcelona, Spain, 2011.

[14] E. Hajicova, J. Panevova, and P. Sgall. Language resources need annotations to make them really reusable: The Prague dependency treebank. In *Proceedings of First International Conference on Language Ressources and Evaluation (LREC)*, pages 713–718, 1998.

[15] J. Hockenmaier and M. Steedman. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn treebank. *Computational Linguistics*, 33(3):355–396, 2007.

[16] T. Lavergne, O. Cappé, and F. Yvon. Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 504–513. Association for Computational Linguistics, July 2010.

[17] M. Markus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:313–330, 1993.

[18] S. Riezler, T. King, R. Kaplan, R. Crouch, J. Maxwell, and M. Johnson. Parsing the Wall Street Journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics (ACL)*, pages 271–278. Association for Computational Linguistics, 2002.

[19] B. Sagot. The Lefff, a freely available and large-coverage morphological and syntactic lexicon for French. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, 2010.

[20] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.