N:M Mapping in XDG –The Case for Upgrading Groups

Jorge Marques Pelizzoni and Maria das Graças Volpe Nunes

Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação
Av. do Trabalhador São-Carlense, 400. CEP 13560-970. São Carlos – SP – Brasil
`{jorgemp, gracan}@icmc.usp.br`
`http://www.nilc.icmc.usp.br`

Abstract. The eXtensible Dependency Grammar (XDG) is a very promising CP-natural framework with which to tackle varied NLP problems and their combinatorial complexity. XDG draws heavily on its non-transformational character for efficiency, which opens the issue of N:M mapping e.g. between syntactic and semantic structures. We resume discussion on this issue and attempt to demonstrate that improvement of the available solutions is at once desirable and crucial. To this end, we assess their suitability in several scenarios assuming a syntax-semantics interface: parsing *vs.* generation; treating Multiword Expressions; treating connectives introducing optional components such as adverbials; etc. Finally, we propose some guidelines to overcome the identified limitations.

1 Introduction

The intuition has been widespread for some time so far that language processing – whether natural or artificial – emerges from the interplay of various concurrent constraints operating at or between different levels of analysis. At times it seems almost possible to feel determinacy ebbing and flowing through such constraint systems e.g. as the potential ambiguity of the words in a sentence gradually reduces until satisfactory interpretations become available. This is especially true of natural processors, i.e. humans, when tackling a second language. As the paradigms of *Constraint Satisfaction* (CS) and particularly *Constraint Programming* (CP) have arisen in Computer Science exactly to mimic this ebb-and-flow intuition suitable to tame complexity in a whole range of problems, no wonder language processing is also in focus.

However, CP faces therein a twofold challenge: not only does the (in)exact nature of linguistic constraints and objects still elude us all, CP practitioners or not, but also much of the linguistic tradition, drawing heavily upon transformational primitives, is not usually amenable to straightforward or *efficient* modelling, to say the least. Therefore, the latest years have seen much effort to strengthen propagation in grammar modelling, which often led to alternative constraint-based frameworks. The move from early (and already not so mainstream) Tree Adjoining Grammar-based frameworks [10,11,12] to more recent Dependency Grammar-based ones [9,8,7] is certainly bound for stronger propagation, regardless and arguably to the detriment of explanatory adequacy.

The *eXtensible Dependency Grammar* (XDG) [4,5,6] is perhaps the latest stage in this evolutionary line and represents an important leap from its ancestors. Even though still leaving much room for further development as we shall presently see, XDG achieves an unprecedented balance between (i) complexity, which is hopefully controlled by the strong propagation it inherits from DG-based frameworks, (ii) generality, i.e. potential coverage of phenomena or application spectrum, which is significantly broadened by XDG's underspecification, extensibility and novel *multidimensional* metaphor, and (iii) instantiability, i.e. ease of instantiation or application, which benefits from XDG's enhanced support for modularity and reusability. As a generality bonus, XDG is such a CP natural that a CP implementation can actually achieve *bidirectionality*, i.e. the property that one same grammar might be used both for analysis and generation with the same search engine modulo I/O processing.

In that respect, Debusmann et al. [5] have already sketched a XDG-based relational (i.e. bidirectional) syntax-semantics interface. However, they bypassed the issue of **N:M mapping** – or **subgraph handling** – then, which is nonetheless unavoidable if one is ever to tackle most function words, especially connectives and auxiliary verbs, support verbs (such as "do" in "do the dishes" or "have" in "have an argument with") and multiword expressions (MWEs), since these usually involve worthy syntactic nodes that in spite of influencing interpretation have no semantic counterpart. The issue has been addressed by Debusmann elsewhere [2] specifically with a focus on MWEs, which were tackled in XDG by means of a restricted form of grouping and deletion, herein referred to simply as the *group* construct. This remains the sole such account so far and, ingenious as it is to provide a very promising technique, it does not go beyond MWEs and leaves, even in that matter, several open issues. The general purpose of this paper is exactly to resume discussion from that point and address some of these issues. Rather than providing definitive solutions, our highest goal is to demonstrate that improvement is at once desirable and crucial while gathering requirements for future developments.

First of all, we provide a little background on XDG (Section 2) and review Debusmann's group technique (Section 3). Next we demonstrate some of its shortcomings (Section 4), as (i) when treating connectives introducing optional components, like adverbials, (ii) when some particularities of MWEs come into play, and (iii) when one consider the convenience of an N_1:N_2:…:N_n mapping generalization. Having presented this rationale and thus made the main point of the paper, we very briefly provide some pointers as to future work on the lexicon component towards overcoming those limitations (Section 5). At all times our main concerns are those of generality (i.e. broaden the coverage of groups, also ensuring bidirectionality), instantiability (make groups usable), and complexity/scalability (make groups feasible in terms of lexicon storage and keep propagation strong).

2 XDG Background

We start by reviewing Debusmann's treatment of MWEs [2] by means of groups, which requires acquaintance with XDG's core concepts. Therefore, an informal overview of these concepts is also in order. For a formal description of XDG, however, see Debusmann et al. [4,5,6].

Most of XDG's strengths stem from its *multidimensional* metaphor (see Fig. 1), whereby an (holistic or multidimensional) XDG analysis consists of a set of concurrent, synchronized, complementary, mutually constraining one-dimensional analyses, each of which is itself a *graph* sharing the same set of vertices as the other analyses, but having its own type or *dimension*, i.e., its own edge label and lexical feature types and its own well-formedness constraints. In other words, each 1D analysis has a nature and interpretation of its own, associates each vertex with one respective instance of a data type of its own (lexical features) and establishes its own relations/edges between vertices using labels and principles of its own. For example, an XDG grammar/analysis might have four dimensions/1D analyses, two for syntax (immediate dominance and linear precedence) and two for semantics (predicate argument structure and scope), as in Debusmann et al. [5].

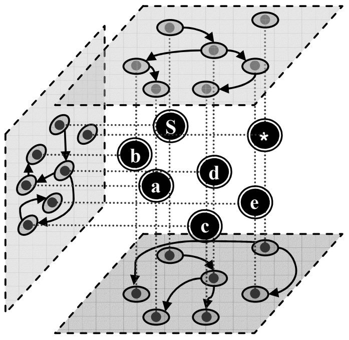

Fig. 1. Three concurrent one-dimensional analyses. It is the sharing of one same set of vertices that co-relates and synchronizes them into one holistic XDG analysis.

That might sound rather autistic at first, but the 1D components of an XDG analysis interact in fact. It is exactly their sharing one same set of vertices, whose sole intrinsic property is identity, that provides the substratum for interdimensional communication, or rather, *mutual constraining*. That is chiefly achieved by means of two devices, namely: interdimensional principles and lexical synchronization.

Interdimensional Principles. Principles are reusable, usually parametric constraint predicates used to define grammars and their dimensions. Those posing constraints between two or more 1D analyses are said *interdimensional*. For example, one XDG grammar defining one dimension to capture predicate argument (PA, modelling semantic roles) structure and another one for immediate dominance (ID, modelling syntactic relations) may prescribe a principle between them ensuring that, for every vertex *v* fulfilling some lexical precondition, whenever there is an edge *pat(ient)* from *v* to some other vertex *w* on dimension PA, then there is an edge *subj(ect)* to *w* on dimension ID. This constraint is more formally expressed thus:

$$\forall v \in V \left(unaccusative(v) \rightarrow \forall w \in V \left(v \overset{pat}{\underset{PA}{\longrightarrow}} w \quad u \quad V \left(u \overset{subj}{\underset{ID}{\longrightarrow}} w \right) \right) \right), \tag{1}$$

where *V* is the set of vertices of the current analysis.

This example principle is subsumed by the parametric principle *linkingEnd* available in the XDG Development Kit[1] (XDK). Each application of *linkingEnd* takes three parameters, namely two dimensions D_1 and D_2 and one function

$$linkf\colon V \times V \times label(D_1) \quad 2^{label(D_2)}, \tag{2}$$

where *label(D)* denotes de set of all possible edge labels on dimension *D* and whose function is thus map edges *(v,w,l)* on D_1 into sets of edge labels on D_2. The meaning of *linkingEnd(*D_1*,*D_2*,linkf)* can be more formally expressed thus:

$$\varnothing \qquad \Longrightarrow \qquad \forall v,w,l \left(v \overset{l}{\underset{D_1}{\rightarrow}} w \quad linkf(v,w,l) \qquad u,l' \left(l' \quad linkf(v,w,l) \quad u \overset{l'}{\underset{D_2}{}} w \right) \right). \tag{3}$$

Parameter *linkf* may well be set by an application of

$$defaultf(D) = \lambda(v,w,l).\ lex_D(v)('end')(l), \tag{4}$$

where $lex_D(v)$ maps vertex *v* into a function giving access to its lexical features according to dimension *D*. Feature *end*, in turn, should denote a function mapping D_1 edge labels into subsets of *label(*D_2*)*. Therefore, the constraint in Equation (**1**) can easily be implemented by *linkingEnd(PA,ID,defaultf(PA))*, provided that unaccusative verbs have lexicon entries whose *end* features for dimension *PA* map label *pat* into the singleton $\{subj\}$.

Some remarks are worth making as regards the above example, namely (i) that shared vertices constitute the real and only meeting points between 1D analyses, (ii) that interdimensional principles strongly rely on (i) to do their job, and (iii) that most of them are *lexicalized*, i.e. impose constraints that depend on the lexical features of vertices according to the dimensions involved. Further details on lexicalization are provided below.

Lexical Synchronization. As pointed out above, principles, whether intra- or interdimensional, usually resort to the lexical features of vertices. This implies that any XDG instance has a lexicon as a component, which is specified in two steps: first, each dimension declares its own lexicon entry type, i.e. an Attribute-Value Matrix (AVM) type; next, once all dimensions have been declared, lexicon entries are provided, each specifying the values for features on all dimensions. Finally, at runtime it is required of well-formed analyses that there is at least one valid assignment of lexicon entries to vertices such that all principles are satisfied. In other words, every vertex must be assigned a lexicon entry that *simultaneously* satisfies all

[1] http://www.ps.uni-sb.de/~rade/xdg.html.

principles on all dimensions, for which reason the lexicon is said to *synchronize* all 1D components of an XDG analysis.

Lexical synchronization is a major source of propagation. Resuming our unaccusative verb example and assuming a text generation scenario, if it happens to be known at some point that there are no more sources of *subject* edges available on the ID dimension, then lexicon entries corresponding to unaccusative realizations of an as yet unrealized verb shall be discarded. That might further narrow the domains of variables on various dimensions and trigger further propagation.

3 The State of the Art – Groups

It must be clear from the previous section that the sharing of one same set of vertices by the 1D components of an XDG analysis is key to the framework and that, formally speaking, deletion or insertion operations would be out of the question. On the face of it, that might seem lethal to any higher aspirations on the part of an XDG-based syntax-semantics interface. Indeed, for starters, it is reasonable to expect that MWEs such as "(to) have an argument with" and "argue with" should correspond to one same single semantic literal, say *argue*, in spite of comprising multiple vertices on syntactic dimensions (one per word). On the other hand, it is sometimes the case that a semantic subgraph should correspond to fewer syntactic nodes, as in instrument incorporation. For example, "cut with a knife" is likely to have a semantic representation comprising at least two semantic vertices, whose realization in some languages, however, would take one single word (for one, the Brazilian Sign Language). Finally, in general, one would not expect vertices corresponding to connectives (prepositions and conjunctions) to have direct counterparts on a semantic dimension, much though they should be taken into account during interpretation or somehow produced during generation.

Emulating Deletion. Debusmann [**2**] introduces a simple though clever technique to circumvent this limitation, or rather, to carry out N:M mapping on top of XDG. The basic idea behind his technique is emulating deletion thus: whenever a vertex has but one incoming edge with a reserved label, say *del*, it is considered as virtually deleted. In addition, one artificial "root" node is postulated from which emerge as many *del* edges as required on all dimensions. Given that lexicon entries usually rule the valency of vertices on each dimension separately, i.e. state which labels they accept on incoming and outgoing edges, it is straightforward to state that a vertex should be deleted on certain dimensions, it sufficing to provide lexicon entries accepting only and necessarily one incoming *del* edge on the referred dimensions.

Crucial though it is, the possibility of deletion by itself is not enough. There must also be a way to treat certain subgraphs as units, or rather *groups*, whose rationale must be given from two complementary points of view.

The Group Coherence Problem. From the point of view of parsing, the fact must be captured that the group meanings of "have a word with" or "take out" compete with other readings of their component words that are applicable in other contexts ("out" can even be an adjective!). In other words, treating "have a word with" as a block in

XDG implies that *each* word will have, among its lexicon entries, one *have-a-word-with* entry, i.e. one exclusively reserved to the group reading. Now, when parsing "write down a word", the *have-a-word-with* entry of "word" will compete with the applicable entry and should be guaranteed to lose. What is worse, any *have-a-word-with* entry should be guaranteed to lose unless enough *have-a-word-with* entries (one for each of the component words) are available at the same time for the sentence at hand.

That problem will herein be referred to as the **Group Coherence Problem (GCP)**. Further generalizing and introducing some useful terminology: (i) every group reading is built by selecting a set of **grouped lexicon entries**; (ii) any such set is said to be a **group instance**; (iii) in order for a grouped lexicon entry to be selected it is necessary that enough **cogrouping entries**, or rather, enough of its **cogroupers** are available; (iv) given two group instances G_1 and G_2, they are said to be **(group) alternates** iff there exist two entries $e_1 \in G_1$ and $e_2 \in G_2$ such that e_1 and e_2 cogroup (e.g. "take out" and "takes out" may well be parsed by group alternates); (iv) the **paradigm** of a group instance G is the union of all its alternates according to a lexicon (e.g. the set of all lexicon entries necessary to parse "take/takes/took/taking/etc. out").

The GCP can thus be rephrased as the problem of identifying well-formed group instances and ensuring that all cogroupers are simultaneously in the same state, either selected or discarded. The solution proposed by Debusmann is based again on a simple though ingenious idea. It consists of (i) capturing the inner structure of group instances and (ii) requiring it to be wholly reconstituted by cogrouping entries only. If (ii) cannot be satisfied, it means that there is at least one necessary cogrouper missing, no group instance can be built, and all involved cogroupers are discarded.

The trick is not so easily implemented now. First of all, every group paradigm P is assigned one unique identifier $gid(P)$. Next, every lexicon entry must bear its group id, even non-grouped entries (which might all take one same reserved null id). To this end, one dimension must specify a special lexical feature, say *group*, to hold this bit of information. Finally, all dimensions bearing structure (some may axiomatically hold unconnected vertices only) must specify a lexical feature, say *outgroups*, of the type

$$lex_D(v)('outgroups') : label(D) \rightarrow 2^{dom(gid)}, \qquad \textbf{(5)}$$

where $dom(f)$ denotes the domain of function f. For every dimension D having this feature, *outgroups* maps edge labels into sets of group ids and is intended to shortlist the otherwise worthy receivers of the edges emerging from any given vertex. More specifically, given a pair of vertices *src* and *target* whose respective valencies otherwise sanction an edge *(src,target,label)* on dimension D, any such edge will be nonetheless inhibited unless the group of (the selected lexicon entry for) *target* belongs to $lex_D(src)('outgroups')(label)$. This new constraint is the **group coherence principle** and must hold for every structure-bearing dimension.

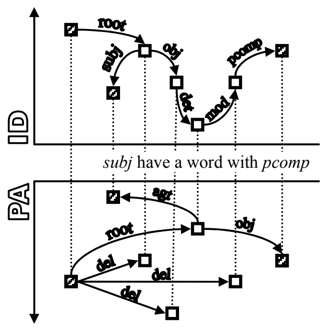

Fig. 2: analisys of an occurrence of the MWE "have a word with" showing dimensions ID (syntactic) and PA (semantic, in which there is deletion). Shaded vertices are external to the group at issue

For example, supposing that "have a word with" should have the structure depicted in Fig. 2 and correspond to one single semantic literal *talk*, the rows in Table 1 show what the respective lexicon entries would look like for a selection of relevant features. In this table, wherever relevant, features are subscripted with their respective dimensions. For example, the forth column regards feature *outgroups* on dimension ID. Apart from the already introduced features or otherwise self-explanatory ones, *in* resp. *out* compose the valency of a vertex, i.e. hold information regarding (i) which labels are accepted for incoming resp. outgoing edges and, for each listed label *l*, (ii) a cardinality constraint, stating e.g. whether at least one *l* edge is required (*l* or *l*+) or not (*l?* or *l**) and whether the vertex may accept at most one *l* edge (*l* or *l?*) or more (*l** or *l*+). On the second column, *(del)* is but a special literal reserved for deleted vertices.

Table 1. Lexical entries encoding the grouping of "talk to" and "have a word with"

word	literal	group	outgroups$_{ID}$	in$_{ID}$	out$_{ID}$	in$_{PA}$	out$_{PA}$	link$_{PA}$
talk	*talk*	g1	$\{iobj \mapsto \{g1\}\}$	$\{root\}$	$\{subj, obj\}$	$\{root\}$	$\{agt, obj\}$	$\left\{\begin{array}{l} agt \mapsto \{subj\} \\ obj \mapsto \{obj\} \end{array}\right\}$
to	(del)	g1	$\varnothing$	$\{iobj\}$	$\{pcomp\}$	$\{del\}$	$\varnothing$	$\varnothing$
have	(del)	g2	$\{obj \mapsto \{g2\}\}$	$\{root\}$	$\{subj, obj\}$	$\{del\}$	$\varnothing$	$\varnothing$
a	(del)	g2	$\{det \mapsto \{g2\}\}$	$\{obj\}$	$\{det\}$	$\{del\}$	$\varnothing$	$\varnothing$
word	*talk*	g2	$\{mod \mapsto \{g2\}\}$	$\{det\}$	$\{mod\}$	$\{root\}$	$\{agt, obj\}$	$\left\{\begin{array}{l} agt \mapsto \{subj\} \\ obj \mapsto \{obj\} \end{array}\right\}$
with	(del)	g2	$\varnothing$	$\{mod\}$	$\{pcomp\}$	$\{del\}$	$\varnothing$	$\varnothing$

Vertex Expansion. By reconsidering the same "have a word with" example now from the point of view of generation (i.e. given input on dimension PA, try to reconstruct valid analyses on dimension ID), an additional issue emerges closely related to that of model creation in CP and named herein the *Vertex Expansion Problem* (VEP). As far as regards actual input to a generation system, one might well expect to generate e.g. "They have a word with Mary" from a simple three-vertex PA graph as seen in Fig. 3, which obviously falls short of vertices. It should be clear by now that, under grouping, there is more to setting up the XDG scene – or rather, model creation – than simply transferring the input graph onto the PA dimension and naïvely looking literals up on the lexicon. A *group-oriented* lookup procedure is strictly needed, even though missing in XDG's current implementation.

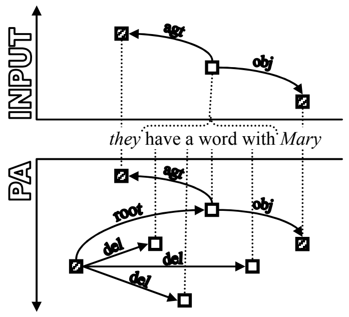

Fig. 3: an instance of the Expansion Problem in generation. Although PA is the input dimension, the actual input lacks vertices

Debusmann has already pointed out the need for such a procedure and proposed one. First, he states that, for generation, a function

$$groups : Sem \rightarrow 2^{dom(gid)} \qquad \textbf{(6)}$$

(where *Sem* is the set of all semantic literals) is needed mapping any given literal *lit* to a set containing the ids of all groups realizing *lit*. During model creation, for each literal *lit*: (i) the set *Ps* is retrieved of the paradigms[2] of all groups in *groups(lit)*; (ii) then the set *V* is created of new vertices such that $|V| = \max\{|P| : P \in Ps\}$; (iii) for each paradigm $P \in Ps$, its entries are arbitrarily assigned to nodes in *V*, i.e., an arbitrary injection $assign_P : P \rightarrow V$ is constructed; then (iv) the base set of alternative lexicon entries for each vertex $v \in V$ is constructed thus:

$$entries(v) = \bigcup_{P \in Ps} \begin{cases} assign_P^{-1}(v), & v \in dom\left(assign_P^{-1}\right) \\ \varnothing, & otherwise \end{cases}; \qquad \textbf{(7)}$$

(v) finally, the *actual* set of alternative lexicon entries for each vertex is defined thus:

$$entries'(v) = \begin{cases} entries(v), & |entries(v)| = |V| \\ entries(v) \cup Del, & otherwise \end{cases}, \qquad \textbf{(8)}$$

where *Del* is a constant lexical entry allowing simultaneous deletion on all dimensions and thus accounting for paradigms requiring fewer nodes than available.

For an example application of this method, one might consider the realization of literal *talk* according to the lexicon in Table 1. The number of vertices created for this

[2] Specializing the general definition for Debusmann's solution, a set of lexicon entries sharing one same group id is a group paradigm iff it contains all lexicon entries sharing that id.

sole literal would be four, say $\{u,w,v,x\}$, so as to hold the paradigm with the greatest number of entries ("have a word with"). One valid actual assignment of alternative lexical entries is given in Table 2, which states, for example, that vertex *u* will be selecting either the first or the fifth row/entry. Notice that vertices *v* and *x* will select the special *Del* entry in the event of "talk to" being generated (third and forth rows).

Table 2. During generation, actual lexical entries allowing selection between two alternative realizations ("talk to" and "have a word with") of one same semantic literal *talk*. The first column coindexes entries competing for one same vertex

assigned to vertex	**word**	**lit.**	**group**	**outgroups$_{ID}$**	**in$_{ID}$**	**out$_{ID}$**	**in$_{PA}$**	**out$_{PA}$**	**link$_{PA}$**
u	talk	*talk*	g1	$\{iobj \mapsto \{g1\}\}$	$\{root\}$	$\{subj, obj\}$	$\{root\}$	$\{agt, obj\}$	$\left\{\begin{matrix} agt \mapsto \{subj\} \\ obj \mapsto \{obj\} \end{matrix}\right\}$
w	to	(del)	g1	$\varnothing$	$\{iobj\}$	$\{pcomp\}$	$\{del\}$	$\varnothing$	$\varnothing$
v	(del)	(del)	null	$\varnothing$	$\{del\}$	$\varnothing$	$\{del\}$	$\varnothing$	$\varnothing$
x	(del)	(del)	null	$\varnothing$	$\{del\}$	$\varnothing$	$\{del\}$	$\varnothing$	$\varnothing$
u	have	(del)	g2	$\{obj \mapsto \{g2\}\}$	$\{root\}$	$\{subj, obj\}$	$\{del\}$	$\varnothing$	$\varnothing$
w	a	(del)	g2	$\{det \mapsto \{g2\}\}$	$\{obj\}$	$\{det\}$	$\{del\}$	$\varnothing$	$\varnothing$
v	word	*talk*	g2	$\{mod \mapsto \{g2\}\}$	$\{det\}$	$\{mod\}$	$\{root\}$	$\{agt, obj\}$	$\left\{\begin{matrix} agt \mapsto \{subj\} \\ obj \mapsto \{obj\} \end{matrix}\right\}$
x	with	(del)	g2	$\varnothing$	$\{mod\}$	$\{pcomp\}$	$\{del\}$	$\varnothing$	$\varnothing$

4 Shortcomings and Requirements Gathering

Now we are in a position to sense the limits of the solutions presented in the previous section and thus gather requirements for future enhancements to XDG and its implementation. Our main goal in this section is provide evidence pointing towards a new balance between maybe the most abstract requirements on frameworks, namely the trinity (i) generality, i.e. coverage/expressibility of phenomena, (ii) instantiability, i.e. ease of instantiation, and (iii) complexity. Instantiability is the design-time analogue of complexity, a runtime concept. In fact, even if some methodology is theoretically applicable with satisfactory accuracy (i.e. it is general enough), it is likely to be discarded if its application happens to be too costly. In other words, it is practicality, feasibility, ease, reasonable demand in resources, in summary, instantiability during development and complexity during execution that will eventually drive developers' preference for this or that framework. As regards XDG, instantiability concerns features and primitives of the grammar specification language, while complexity is related to model creation, propagation and stored lexicon size.

4.1 Generality *vs.* Expansion – Bidirectionality and Null Categories

Debusmann [2] has focused on MWEs only and implied that group-oriented lexicon lookup would be a generation trait; however, vertex expansion is not restricted to generation, which is already suggestive that such an enhanced lookup/model creation procedure is actually the general – rather than the exceptional – case. Take for instance instrument incorporation in the whole set of "cut-with-a-X" words in Brazilian Sign Language [1], not to mention manner and intensity incorporation.

Although work is lacking on this specific matter, if ellipsis is ever to be handled in XDG, that will probably resort to some kind of vertex expansion during parsing. Consider, for example, the following Portuguese sentence[3]:

Eu	**comprei**	**sapatos**	**brancos**	**e**	**ele**	**pretos.**
I	*have bought*	*shoes*	*white*	*and*	*he (has bought)*	*black (shoes).*
		(= white shoes)				

However such a hard phenomenon is to be tackled, one can count on the fact that "Eu" and "ele" cannot share the same vertices for "(has/have) bought" or "shoes", at least not directly. Generally speaking, all evidence suggests that, if **syntactic null categories** are ever to be "parsed", that will require vertex expansion as they cannot rely on the presence of words of their own to trigger the generation of their respective vertices during model creation.

4.2 Complexity *vs.* Expansion

Inflecting MWEs. One issue that Debusmann has left open in his XDG account of MWEs is related to inflection, i.e. how such alternations as "have/has/had an/∅ argument/s[4] with" are to be encoded. As we shall presently see, the current solution might have undesired side-effects on complexity, either on lexicon storage or propagation. In the discussion below, one should always take into account that there are languages considerably more inflected than English. Romance languages, for example, deliver tenths of distinct inflected forms for every single verb, counting out the so-called *compound forms*, i.e. those involving auxiliary verbs.

The obvious first impulse would be rather flawed to encode a whole grammatical paradigm into one sole group paradigm, i.e. to make all inflections of a given base MWE share the same group id, *which might even work for parsing* (strictly without vertex expansion) but not generation[5]. Take for instance the grammatical paradigm "have/has/had a word with". In addition to the relevant entries in Table 1, its encoding would also require two further entries – for "has" and "had" respectively – quite similar to that of "have". Supposing they all share the same group id and assuming a generation scenario, on application of the vertex expansion procedure described in the previous section six vertices would be created instead of the correct four, and either failure or malformed output would follow. Apropos, in generating a Romance language over forty vertices would usually sprout. This so to speak vertex prodigality stems from the fact that, in this misencoding, there is no clue whatsoever that some entries should compete for one same vertex.

[3] Thanks to Denys Duchier, originally in German.
[4] The alternation "had an argument with" *vs.* "has arguments with" can arguably be regarded as involving aspect inflection.
[5] One corollary worth deriving from the following facts is that XDG is only *potentially* bidirectional, i.e. there are grammars that might work in one direction but not in the other.

Restricting ourselves to valid inflection schemes, we were able to devise two such designs, both of which cannot help creating one exclusive group paradigm for each inflection. They differ, however, in the possibility of paradigms sharing some entries. Unfortunately, both of them entail complexity side-effects as explained below.

No Sharing: Storage Complexity and Overactivation. Let us first assume the simplest solution, namely create as many group paradigms as there are inflections ensuring that they do not share one entry whatsoever. In this design, if "have a word with" takes four lexicon entries, then so does "has a word with", and altogether this makes eight distinct entries. The sole difference between the whole lot of "a/word/with" entries lies in their group ids.

The design works but has two major disadvantages, namely: (i) storage requirements are subject to a *significant multiplying factor*, which will get much worse by the end of this section; (ii) this factor also affects model creation inasmuch as **overactivation** is likely to occur, i.e. having vertices select from loads of virtually equivalent lexicon entries. For example, in Portuguese, either when generating or parsing something like "ter uma discussão com" ("have an argument with"), there would be three vertices trying to select from over forty different entries for "discussão", "uma" and "com" respectively, which, ironically, are invariable in this MWE.

Sharing: All the Same or Spurious Symmetries. An alternate encoding method would be keep one group paradigm per inflection sharing equivalent entries (modulo group ids) with all the others. Enabling sharing would require a minor adaptation to Debusmann's original solution, namely (i) replace feature *group* (the id of the group to which an lexical entry belongs) with say *groups* (a set thereof), (ii) restate the Group Coherence Principle to hold for every candidate edge *(src,target,label)* on every structure-bearing dimension *D* thus:

$$lex_{GD}(target)('groups') \cap lex_{D}(src)('outgroups')(label) \neq \varnothing\,, \qquad \textbf{(9)}$$

where *GD* (a constant) is the dimension holding feature *groups*.

For example, in this design, "a/word/with" would take one single entry in the encoding of the whole "have a word with" grammatical paradigm. However, this solution comes in two radically different flavours depending on whether group paradigms originating from different grammatical paradigms may be sharers. In ground terms, depending on whether the entries for "a" and "with" might be shared by all inflections not only of "have a word with" but also of "have a fight/quarrel/argument/etc. with" or, more relevantly still, whether entries for the inflections of "have" might also be shared by all groups paradigms in which "have" acts as a support verb. Let us call **restrained** resp. **unrestrained sharing** the solution obtained by refusing resp. accepting those conditions.

Unrestrained sharing certainly answers both the storage complexity and overactivation problems posed by no sharing at all. However, it incurs propagation loss, especially in generation, when the order of vertices is not predetermined, which *might* otherwise help disambiguation. In either direction, odds are that propagation only will not be able to reconstitute the internal structure of groups, as various vertices may potentially belong to one given group, though not *simultaneously*, only *alternately*, which is known to kill propagation. In other words, **symmetries** are likely to be introduced by the encoding scheme – and **spurious** at that, inasmuch as not natural of the problem at hand, but rather brought in by the adopted solution. For example, consider the generation of the following sentence:

$$\text{I } \underset{\text{MWE}_1}{\underline{\text{had a word with}}} \text{ the director after } \underset{\text{MWE}_2}{\underline{\text{having a quarrel with}}} \text{ one of my students.}$$

Under unrestrained sharing, vertex expansion would yield two "a/with" vertices, respectively for MWE_1 and MWE_2. However and precisely due to sharing, both vertices would accept edges from either group, which would block propagation at some point and create a spurious choice point.

On the other hand, restrained sharing reduces spurious symmetries if only for the fact that the co-occurrence of two inflections of one same MWE is less likely. Even if that were satisfactory, it definitely does improve much on storage complexity as compared to no sharing at all, especially in the light of the following new facts.

Connectives in Optional Constituents and the TNT Effect. It is worth reminding that Debusmann's solutions are originally targeted at MWEs, and one might argue that their shortcomings are even acceptable taking into account that MWEs occur much less often than self-contained single words. We reply that the magnifying factor of whatever shortcomings a grouping solution may ever have might well be the sheer size of a whole lexicon. In other words, just imagine what if, after all, there were as many group paradigms as there are words or even word senses in a lexicon, or rather, what if virtually every occurrence of any word involved grouping no matter whether it belongs to a MWE. Minor flaws in the grouping scheme might have a disastrous (TNT) effect then.

In order to clearly see how come, it suffices to leave MWEs, assume a generation scenario and consider where connectives (prepositions and conjunctions) in optional constituents (prepositional phrases and subordinate clauses acting as adverbials or noun modifiers) are ever to come from. Even as simple a sentence as "John died for Mary/love" becomes suddenly surrounded with mystery, and it seems rather unlikely that any definitive MWE solution can be formulated before this issue has been suitably tackled. Notice that our focus is not on how the correct connectives are selected (for example, consider the alternations "at ten o'clock/on Monday/in January", which are all time adverbials and only the tip of the iceberg), although that is a very current subject of debate and research. What we are considering is a much more basic issue: the very mechanism allowing their surfacing once they are expected to have no direct semantic counterpart.

Given the current state of the art in XDG, which includes the interesting group technique by Debusmann, we advocate that the hypothesis must be tested that connectives in optional constituents can be generated by some enhanced form of grouping and vertex expansion. Although this surely is in our agenda, we have not yet carried out any such comprehensive test. Instead, we find it essential first to make sure that groups will *scale* – both in terms of instantiability and complexity – or else that simply cannot be the solution.

Preliminary Evidence. We proceed to give preliminary evidence that, assuming such a scalable grouping scheme exists, our hypothesis stands a chance. To this end, let us analyze how the underlined connectives in “Mary knitted it for John while they lived in Paris” could be generated.

Hypothesizing that “for/while” belongs to a group implies asking what its cogrouper(s) must be after all. There seems to be two options only: either “knitted/knitted” or “John/lived”. Generalizing, it is in order to decide whether a connective introducing an optional constituent must group with its governor or its governee, respectively. The second option is the only acceptable, as the resulting groups consist of two components each, namely a connective and (the root of) its governee, while the first option would involve grouping the governor with as many *optional* (i.e. deletable) components as there can be connectives simultaneously governed by the entity at hand (verb, noun, etc.). Not only is that somewhat difficult to determine, but also most of the created components would usually be inactive in most sentences.

Let us hypothetically trace the generation of “for” according to the analysis shown in Fig. 4. Looking up the semantic literal underlying “John” yields two groups, namely G_1, a singleton corresponding to a nominative or accusative occurrence of “John”, and G_2, corresponding to a prepositioned occurrence. Assuming for clarity that only one-word prepositions are possible, two vertices are created, say *prep* and *john*. Two lexicon entries compete for *john*, one (belonging to G_1) accepting either a *subj(ect)* or a *(direct) obj(ect)* edge on the ID dimension and the other (G_2) accepting only *pcomp* (preposition complement). As for vertex *prep*, several entries compete for it, one allowing deletion (because G_1 does not have two components) plus one G_2 entry per possible preposition, accepting, among others, *adv* and donating *pcomp* only to G_2 members by means of feature *outgroups*.

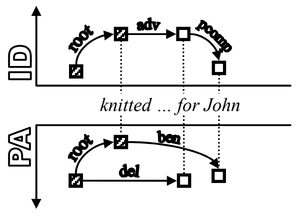

Fig. 4. Fragment of an analysis of “Mary knitted it for John”

All that remains to be explained is how “for” (or an equivalent) is to be selected among all other prepositions, or rather, how to state (i) that the specific word “for” is a possible realization of the *ben(eficiary)* relation on dimension PA and (ii) that its ID mother (“knitted”) should be the PA mother of its ID daughter (“John”). For those acquainted with XDG, that immediately suggests some interdimensional principle involving a lexical feature, more precisely some kind of *linking principle*. In fact, the XDK already provides library principle *LinkingDaughterEnd*, which is almost what we need. Given two dimensions D_1 and D_2 and some function *linkf* of the type given in Eq. (2), it ensures that, for every edge *(src,target,label)* on D_1, either there is an edge *(src,target,label')* on D_2 such that either $label' \in linkf(src,target,label)$ or $linkf(src,target,label)$ is empty.

For our purposes, a similar principle would suffice operating, though, on groups instead of directly on vertices. Given two dimensions D_1 and D_2 and some function *linkf*, hypothetical principle *LinkingDaughterGroupEnd* would ensure that, for every edge *(src,t,l)* on D_1, either (i) there is an edge *(src,t',l')* on D_2 such that $l' \in linkf(src,t,l)$ and $groups(t) \cap groups(t') \neq \varnothing$ or (ii) $linkf(src,t,l)$ is empty. Letting D_1 = ID, D_2 = PA and *linkf* access feature $PAEnd_{\mathrm{ID}}$ of edge targets thus:

$$linkf = \lambda(\_,target,label).\ lex_{ID}(target)('PAEnd')(label)\,, \qquad \textbf{(10)}$$

the application of *LinkingDaughterGroupEnd* ensures the selection of “for”, provided that its lexical entry has a value *f* for $PAEnd_{\mathrm{ID}}$ such that $'ben' \in f('adv')$.

The generation of “while” is perfectly analogous except for the fact that governed finite verbs group with governing conjunctions instead of prepositions. Finally, it is worth mentioning that this grouping solution (governing connectives with governed entities) should only be applied to optional constituents. In contrast, prepositions introducing indirect objects (such as “of” in “approve of”) should group with their governing verbs, much like MWEs.

Consequences. In the event that the hypothesis introduced above is accepted, grouping will come to play a leading role in XDG praxis. Its status might well be upgraded to that of a primitive. For a start, we have already provided evidence that group-oriented versions of library principles will be needed, and it would be no wonder if the whole original library suddenly became obsolete. In addition, lexicon language will have to be revised to make groups really instantiable, i.e. more friendly to grammar developers. And, as we shall briefly argue in Section 5, so will probably part of its operational semantics, in order to reduce storage complexity.

4.3 Instantiability vs. Expansion – Verbs, Nesting, Crossing, and Generalized N_1:…:N_n Mapping

Much of what has been discussed for connectives and their generation also applies to auxiliary verbs, even if only to such perfectly grammatical auxiliaries as English “do” in questions and negative sentences. Under our hypothesis for connective generation, it seems reasonable that at least “do” is to be generated by grouping with its main verbs. As all English verbs but a few exceptional cases (auxiliaries and “be”) may take this auxiliary, the issues discussed previously are still relevant.

Nevertheless, whether auxiliary or not, verbs are somewhat more complex objects than connectives. A significant portion of their complexity lies in pure syntax and morphology *irrespective* of semantics. For instance, although verbs such as “take”, “make”, “get”, and non-auxiliary “have” may be employed in a variety of senses, most of their syntactic and morphological behaviour remains the same all across. In XDG terms, “have” may group with “do/does/did” and is inflected “has/had/having/to have/etc.” no matter whether it should be part of “have a word with” or other MWEs or even work on its own in several alternate senses. Instantiability (i.e. modularity and separation of concerns) here demands that such obvious and productive “irrespectiveness” can be captured, i.e., that partial behaviour can be defined once and for all for later reuse. And as much as possible: it is highly desirable that such very productive paradigms as grouping with "do" and the "to" infinitive particle could be defined for a whole class of verbs at once. It is worth noticing that the abstract concept of partial and thus reusable specifications already underlies the XDK's design, although the requirements we have been gathering are not currently met, probably as a result of the underrated status groups have enjoyed thus far.

The described requirements hint at some sort of **group nesting** (e.g. "do have" might be a subgroup inside "[[do have] an argument with]", as might "on behalf of" in "[[on behalf of] Mary]"), **partial groups** (i.e. underspecified groups, or rather, groups setting a strict subset of all the required features) and the **cross product of complementary partial group paradigms**. The latter refers to the concept that **complementary partial groups** (i.e. setting disjoint subsets of features) might be combined to generate a new group. For example, English phrasal verb "cut off" might have all its morphosyntactic behaviour captured in a partial paradigm *Syn* involving nesting ("[[do cut] off]" and such like). Next, various complementary paradigms $\{Sem_1,\ldots,Sem_n\}$ should specify the possible meanings of "cut off" ("stop", "separate", etc.), which might also comprise complex subgraphs. Finally, a hypothetical special cross product $Syn \,\underline{\times} \bigcup\{Sem_1,\ldots,Sem_n\}$ would conveniently yield all the expected group paradigms, which might, in turn, still be partial and thus reusable for further crossing. Such a scheme, which we have not yet formalized but rather sketch as a teaser, would provide a convenient form of **generalized $\mathbf{N_1{:}\ldots{:}N_n}$ mapping**.

5 Future Work – Upgrading Groups and On-Demand Lexicon

The main point of this article has been to resume the discussion on grouping in XDG, give evidence as to how central the issue is to XDG development and gather requirements for enhancing the framework. We believe the way from here is to upgrade groups to the status of a primitive, if not of XDG's core, at least of the XDK's lexicon language.

Whether the upgrade makes it to the core or not, one possibility that seems rather promising and unavoidable is modifying the operational semantics of the lexicon component to circumvent storage complexity, among others. In other words, we intend further to exploit the fact that the lexicon component of an implementation does not have directly to reflect its formal counterpart. Specifically, it may well become an *on-demand producer of actual lexicon entries* on an input-by-input basis. This means that all nesting and crossing of primitive group paradigms can be performed on the fly according to the input at hand, which can be relatively easily implemented by means of higher-order programming and is likely to decrease the number of active lexicon entries dramatically even in face of massive grouping.

Such a scheme appears all the more feasible if one takes into account automatic on-the-fly generation and assignment of group ids, every new group occurrence receiving a fresh group id in the scope of the current input, which is rather straightforwardly implemented by means of functional programming and logic variables. This would spare grammar developers from dealing directly with awkward, error-prone group id features and is only possible because groups are very well-behaved: given a group, its edges are either constrained to be internal to itself or free to link to any other group.

Nesting is likely to complicate things a little, but not too much, probably it sufficing to introduce one third option, namely "or constrained to be internal to the innermost enclosing group".

We hope that a lexicon component may thereby reconcile (i) unrestrained sharing (Section 4.2) in grammar development and storage with (ii) neither sharing nor overactivation at all in model generation. All the referred constructs and a vertex expansion algorithm are currently being designed and shall be presented in due time.

Acknowledgements

This research project has been partially funded by *Conselho Nacional de Desenvolvimento Científico e Tecnológico* – CNPq, a Brazilian government agency fostering technological and scientific development.

References

1. Brito, L. F. Por uma Gramática de Línguas de Sinais. Tempo Brasileiro Ed., Departamento de Lingüística e Filologia, Universidade Federal do Rio de Janeiro (1995)
2. Debusmann, R.: Multiword Expressions as Dependency Subgraphs. In: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004, "Multiword Expressions: Integrating Processing" Workshop)
3. Debusmann, R., Postolache, O., Traat, M.: A Modular Account of Information Structure in Extensible Dependency Grammar. In: Sixth International Conference on Intelligent Text Processing and Computational Linguistics (CICLING 2005)
4. Debusmann, R., Duchier, D., Kruijff, G. J.: Extensible Dependency Grammar: A New Methodology. In: Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004, Workshop on Recent Advances in Dependency Grammar)
5. Debusmann, R., Duchier, D., Koller, A., Kuhlmann, M., Smolka, G., Thater, S.: A Relational Syntax-Semantics Interface Based on Dependency Grammar. In: Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)
6. Debusmann, R., Duchier, D., Kuhlmann, M.: Multidimensional Graph Configuration for Natural Language Processing. In: Proceedings of the International Workshop on Constraint Solving and Language Processing (2004) 59–73
7. Duchier, D.: Configuration of labeled trees under lexicalized constraints and principles. In: Journal of Language and Computation (2002)
8. Duchier, D.: Axiomatizing dependency parsing using set constraints. In: Proceedings of the 6th Meeting on the Mathematics of Language (1999)
9. Duchier, D., Debusmann, R.: Topological dependency trees: A constraint-based account of linear precedence. In: Proceedings of the 39th ACL (2001)
10. Duchier, D., Thater, S.: Parsing with Tree Descriptions: a Constraint-Based Approach. In: Sixth International Workshop on Natural Language Understanding and Logic Programming (NLULP 1999) 17–32
11. Gardent, C., Thater, S.: Generating with a Grammar Based on Tree Descriptions: a Constraint-Based Approach. In: Bird, S. (ed.): Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)
12. Koller, A., Striegnitz, K.: Generation as Dependency Parsing. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002) 17-24