Capturing Disjunction in Lexicalization with Extensible Dependency Grammar

Jorge Marques Pelizzoni
ICMC - Univ. de São Paulo - Brazil
Langue & Dialogue - LORIA - France
`Jorge.Pelizzoni@loria.fr`

Maria das Graças Volpe Nunes
ICMC - Univ. de São Paulo - Brazil
`gracan@icmc.usp.br`

Abstract

In spite of its potential for bidirectionality, Extensible Dependency Grammar (XDG) has so far been used almost exclusively for parsing. This paper represents one of the first steps towards an XDG-based integrated generation architecture by tackling what is arguably the most basic among generation tasks: lexicalization. Herein we present a constraint-based account of *disjunction* in lexicalization, i.e. a way to enable an XDG grammar to generate all paraphrases — along the lexicalization axis, of course — realizing a given input semantics. Our model is (i) efficient, yielding strong propagation, (ii) modular and (iii) favourable to synergy inasmuch as it allows collaboration between modules, notably semantics and syntax. We focus on constraints ensuring well-formedness and completeness and avoiding over-redundancy.

1 Introduction

In text generation the term *lexicalization* (Reiter and Dale, 2000) refers to deciding which among a choice of potentially applicable lexical items realizing a given intended meaning are actually going to take part in a generated utterance. It can be regarded as a general, necessary generation task — especially if one agrees that the term *task* does not necessarily imply pipelining — and remarkably pervasive at that. For instance, even though the realization of such a phrase as "a ballerina" owes much to *referring expression generation*, a complementary task, it is still a matter of lexicalization whether to prioritize that specific phrase over all its possible legitimate alternates, e.g. "a female dancer", "a dancing woman" or "a dancing female person". However, prior to the statement of prioritizing criteria or selection preferences and rather as the very substratum thereto, the ultimate matter of lexicalization is exactly alternation, choice — in one word, *disjunction*.

Given the combinatorial nature of language and specifically the interchangeability of lexical items yielding hosts of possible valid solutions to one same instance lexicalization task, disjunction may well become a major source of (combinatorial) complexity. Our subject matter in this paper is solely disjunction in lexicalization as a basis for more advanced lexicalization models, and our purpose is precisely to describe a constraint-based model that *(i) captures the disjunctive potential of lexicalization*, i.e. allows the generation of all mutually paraphrasing solutions (according to a given language model) to any given lexicalization task, *(ii) ensures well-formedness*, especially ruling out over-redundancy (such as found in "∗a dancing female dancer/ballerina/woman") and syntactic anomalies ("∗a dancer woman"), and does so *(iii) modularly*, in that not only are concerns neatly separated (e.g. semantics vs. syntax), but also solutions are reusable, and future extensions, likely to be developed with no change to current modules, *(iv) efficiently*, having an implementation yielding strong propagation and thus prone to keep complexity at acceptable levels, and *(v) synergicly*, inasmuch as it promotes the interplay between modules (namely syntax and semantics) and seems compatible with the concept of integrated generation architectures (Reiter and Dale, 2000), i.e. those in which tasks are not executed in pipeline, but are rather interleaved so as to avoid failed or suboptimal choices during search.

We build upon the Extensible Dependency Grammar (XDG) (Debusmann et al., 2004b; Debusmann et al., 2004a; Debusmann et al., 2005) model and its CP implementation in Oz (Van Roy and Haridi, 2004), namely the XDG Development Toolkit[1] (XDK) (Debusmann et al., 2004c).

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

In fact, all those appealing goals of modularity, efficiency and synergy are innate to XDG and the XDK, and our work can most correctly be regarded as the very first attempts at equipping XDG for generation and fulfilling its bidirectional promise.

The paper proceeds as follows. Section 2 provides background information on XDG and the XDK. Section 3 motivates our lexicalization disjunction model and describes it both intuitively and formally, while Section 4 presents implementation highlights, assuming familiarity with the XDK and focusing on the necessary additions and modifications to it, as well as discussing performance. Finally, in Section 5 we conclude and discuss future work.

2 Extensible Dependency Grammar

An informal overview of XDG's core concepts is in order; for a formal description of XDG, however, see (Debusmann and Smolka, 2006; Debusmann et al., 2005). Strictly speaking, XDG is not a grammatical framework, but rather a description language over finite labelled multigraphs that happens to show very convenient properties for the modeling of natural language, among which a remarkable reconciliation between monostratality, on one side, and modularity and extensibility, on the other.

Most of XDG's strengths stem from its *multi-dimensional 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 nodes 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 node with one respective instance of a data type of its own (lexical features) and establishes its own relations/edges between nodes using labels and principles of its own.

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 nodes, whose sole intrinsic property is identity, that provides the substratum for interdimensional communication, or rather, *mutual constraining.*

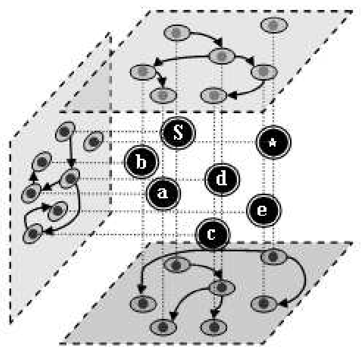

Figure 1: Three concurrent one-dimensional analyses. It is the sharing of one same set of nodes that co-relates and synchronizes them into one holistic XDG analysis.

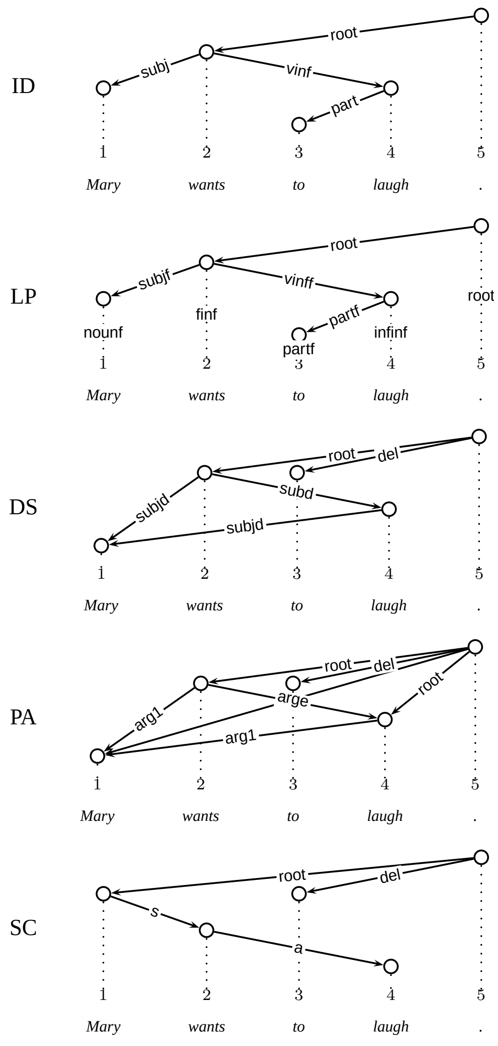

Figure 2: A 5D XDG analysis for "Mary wants to laugh." according to grammar *Chorus.ul* deployed with the XDK

`html`

That is chiefly achieved by means of two devices, namely: multidimensional principles and lexical synchronization.

Multidimensional 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 **multidimensional**. For example, the XDK library provides a host of *linking principles*, one of whose main applications is to regulate the relationship between semantic arguments and syntactic roles according to lexical specifications. The framework allows lexical entries to contain features of the type $lab(D_1) \rightarrow \{lab(D_2)\}$, i.e. mappings from edge labels in dimension D_1 to sets of edge labels in D_2. Therefore, lexical entries specifying $\{pat \rightarrow \{subj\}\}$ might be characteristic of unaccusative verbs, while those with $\{agt \rightarrow \{subj\}\,,\, pat \rightarrow \{obj\}\}$ would suit a class of transitive ones. Linking principles pose constraints taking this kind of features into account.

Lexical synchronization. The lexicon component in XDG is specified in two steps: first, each dimension declares its own lexicon entry 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 should be at least one valid assignment of lexicon entries to nodes such that all principles are satisfied. In other words, every node must be assigned a lexicon entry that *simultaneously* satisfies all 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.

Figure 2 presents a sample 5D XDG analysis involving the most standard dimensions in XDG practice and jargon, namely **(i) PA**, capturing predicate argument structure; **(ii) SC**, capturing the scopes of quantifiers; **(iii) DS**, for *deep syntax*, i.e. syntactic structure modulo control and raising phenomena; **(iv) ID**, for *immediate dominance* in surface syntax (as opposed to DS); and **(v) LP**, for *linear precedence*, i.e. a structure tightly related to ID working as a substratum for constraints on the order of utterance of words. In fact, among these dimensions LP is the only one actually to involve a concept of order. PA and DS, in turn, are the only ones not constrained to be trees, but directed acyclic graphs instead. Further details on the meaning of all these dimensions, as well as the interactions between them, would be beyond the scope of this paper and have been dealt with elsewhere. From Section 3 on we shall focus on PA and, to a lesser extent, the dimension with which it interfaces directly: DS.

Emulating deletion. Figure 2 also illustrates the rather widespread technique of deletion, there applied to infinitival "to" on dimensions DS, PA, and SC. As XDG is an eminently monostratal and thus non-transformational framework, "deletion" herein refers to an emulation thereof. According to this technique, whenever a node has but one incoming edge with a reserved label, say del, on dimension D it is considered as virtually deleted on D. In addition, one artificial root node is postulated from which emerge as many del edges as required on all dimensions. The trick also comes in handy when tackling, for instance, multiword expressions (Debusmann, 2004), which involve worthy syntactic nodes that conceptually have no semantic counterparts.

3 Modelling Lexicalization Disjunction in XDG

Generation input. Having revised the basics of XDG, it is worth mentioning that so far it has been used mostly for parsing, in which case the input type is usually rather straightforward, namely typewritten sentences or possibly other text units. Model creation is also very simple in parsing and consists of (i) creating exactly one node for each input token, all nodes being instances of one single homogeneous feature structure type automatically inferred from the grammar definition, (ii) making each node select from all the lexical entries indexed by its respective token, (iii) posing constraints automatically generated from the principles found in the grammar definition and (iv) deterministically assigning values to the order-related variables in nodes so as to reflect the actual order of tokens in input.

As concerns generation, things are not so clear, though. For a start, take input, which usually varies across applications and systems, not to mention the fact that representability and computability of meaning in general are open issues. Model creation should follow closely, as it is a direct function of input. Notwithstanding, we can to some extent and advantage tell what generation input is not. Under the hypothesis of an XDG-based generation system tackling lexicalization, input is not likely to contain some direct representation of fully specified PA analyses, much though this is usually regarded as a satisfactory output for a parsing system (!). What happens here is that *generating* an input PA analysis would presuppose lexicalization having already been carried out. In other words, PA analyses accounting for e.g. “a ballerina” and “a dancing female human being” have absolutely nothing to do with each other whereas what we wish is exactly to feed input allowing both realizations. Therefore, PA analyses are themselves part of generation output and are acceptable as parsing output inasmuch as “de-lexicalization” is considered a trivial task, which is not necessarily true, however.

Although our system still lacks a comprehensive specification of input format and semantics, we have already established on the basis of the above rationale that our original PA predicates must be decomposed into simpler, primitive predicates that expose their inter-relations. For the purpose of the present discussion, we understand that it suffices to specify that our input will contain flat first-order logic-like conjunctions such as

$$\exists x\,(dance(x) \wedge female(x) \wedge human(x))\,,$$

in order to characterize entities, even if the final accepted language is sure to have a stricter logic component than first-order logic and might involve crossings with yet other formalisms. Predicates, fortunately, are not necessarily unary; and, for example, “A ballerina tapped a lovely she-dog” might well be generated from the following input:

$$\exists e, x, y \left(\begin{array}{c} dance(x) \wedge female(x) \wedge \\ \wedge human(x) \wedge event(e) \wedge \\ \wedge\, past(e) \wedge tap(e,x,y) \wedge \\ \wedge female(y) \wedge dog(y) \wedge \\ lovely(y) \end{array} \right). \quad (1)$$

Deletion as the substance of disjunction. Naturally, simply creating one node for each input semantic literal is not at all the idea behind our model. For example, if “woman” is to be actually employed in a specific lexicalization task, then it should continue figuring as one single node in XDG analyses as usual in spite of potentially covering a complex of literals. In fact, XDG and, in specific, PA analyses should behave and resemble much the same as they used to.

However, one remarkable difference of analyses in our generation model as compared to parsing lies in the role and scope of deletion, which indeed constitutes the very substance of disjunction now. By assigning all nodes but the root one extra lexical entry synchronizing deletion on all dimensions[2], we build an unrestrained form of disjunction whereby whole sets of nodes may as well act as if not taking part in the solution. Now it is possible to create nodes at will, even one for each applicable lexical item, and rely on the fact that, many ill-formed outputs as the set of all solutions may contain, it still covers all correct paraphrases, i.e. those in which all and only the right nodes have been deleted. For example, should one node be created for each of “ballerina”, “woman”, “dancer”, “dancing”, “female” and “person”, all possible combinations of these words, including the correct ones, are sure to be generated.

Our design obviously needs further constraining, yet the general picture should be visible by now that we really intend to finish model creation — or rather, start search — with (i) a bunch of perfectly floating nodes in that not one edge is given at this time, all of which are equally willing and often going to be deleted, and (ii) a bunch of constraints to rule out ill-formed output and provide for efficiency. There are two main gaps in this summary, namely:

- what these constraints are and
- how exactly nodes are to be created.

This paper restricts itself to the first question. The second one involves issues beyond lexicalization, actually permeating all generation tasks, and is currently our research priority. Consequently, in all our experiments most of model creation was handcrafted.

In the name of clarity, we shall hereafter abstract over deletion, that is to say we shall in all respects adhere to the illusion of deletion, that nodes may cease to exist. In specific, whenever we refer to the sisters, daughters and mothers of a node, we mean those not due to deletion. In other words, all happens as if deleted nodes had no relation whatsoever to any other node. This abstraction is extremely helpful and is actually employed in our implementation, as shown in Section 4.

[2]That is, specifying valencies such that an incoming *del* edge is required on all dimensions simultaneously.

3.1 How Nodes Relate

In the following description, we shall mostly restrict ourselves to what is novel in our model as compared to current practice in XDG modelling. Therefore, we shall emphasize dimension PA and the new constraints we had to introduce in order to have only the desired PA analyses emerge. Except for sparse remarks on dimension DS and its relationship with PA, which we shall also discuss briefly, we assume without further mention the concurrence of other XDG dimensions, principles and concepts (e.g. lexical synchronization) in any actual application of our model.

Referents, arguments and so nodes meet. For the most part, ruling out ill-formed output concerns posing constraints on acceptable edges, especially when one takes into account that all we have is some floating nodes to start with. Let us first recall that dimension PA is all about predicate arguments, which are necessarily variables thanks to the flat nature of our input semantics. Roughly speaking, each PA edge relates a predicate with one of its arguments and thus "is about" one single variable. Therefore, our first concern must be to ensure that every PA edge should land on a node that "is (also) about" the same variable as the edge itself.

In order to provide for such an "aboutness" agreement, so to speak, one must first provide for "aboutness" itself. Thus, we postulate that every node should now have two new features, namely (i) $hook$, identifying the **referent** of the node, i.e. the variable it is about, and (ii) $holes$, mapping every PA edge label ℓ into the **argument** (a variable) every possible ℓ-labelled outgoing edge should be about. Normally these features should be lexicalized. The coincidence with Copestake et al.'s terminology (Copestake et al., 2001) is not casual; in fact, our formulation can be regarded as a decoupled fragment of theirs, since neither our $holes$ involves syntactic labels nor are scopal issues ever touched. As usual in XDG, we leave it for other modules such as mentioned in the previous section to take charge of scope and the relationship between semantic arguments and syntactic roles. The role of these new features is depicted in Figure 3, in which an arrow does not mean an edge but the possibility of establishing edges.

Completeness and compositionality. Next we proceed to ensure completeness, i.e. that every so-

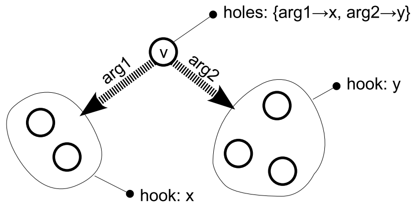

Figure 3: For every node v and on top of e.g. valency constraints, features $hook$ and $holes$ further constrain the set of nodes *able to receive* edges from v for each specific edge label.

lution should convey the whole intended semantic content. To this end, nodes must have features holding semantic information, the most basic of which is $bsem$, standing for *base semantic content*, or rather, the semantic contribution a lexical entry may make on its own to the whole. For example, "woman" might be said to contribute $\lambda x.female(x) \wedge human(x)$, while "female", only $\lambda x.\ female(x)$. Normally $bsem$ should be lexicalized.

In addition, we postulate feature sem for holding the **actual semantic content** of nodes, which should not be lexicalized, but rather calculated by a principle imposing **semantic compositionality**. In our rather straightforward formulation, for every node v, $sem(v)$ is but the conjunction of $bsem(v)$ and the sems of all its PA daughters thus:

$$\begin{gathered} sem(v) \\ = \\ bsem(v) \wedge \bigwedge \{sem(u) : \ v \rightarrow_{PA} u\} \, , \end{gathered} \tag{2}$$

where $v \xrightarrow{\ell}_D u$ denotes that node u is a daughter of v on dimension D through an edge labelled ℓ (the absence of the label just denotes that it does not matter).

Finally, completeness is imposed by means of node feature $axiom$, upon which holds the invariant

$$sem(v) \Rightarrow axiom(v) \, , \tag{3}$$

for every node v. The idea is to have $axiom$ as a lexicalized feature and consistently assign it the neutralizing constant $true$ for all lexical entries but those meant for the root node, in which case the value should equal the intended semantic content.

Coreference classes, concentrators and revisions to dimensions PA and DS. The unavoidable impediment to propagation is intrinsic choice, i.e. that between things equivalent and that we wish to remain so. That is exactly what we would like to capture for lexicalization while attempting to make the greatest amount of determinacy available to minimize failure. To this end, our strategy is to make PA analyses as flat as possible, with **coreferent nodes** — i.e. having the same referent or $hook$ — organizing in plexuses around, or rather, *directly below* hopefully one single node per plexus, thus said to be a **concentrator**. This offers advantages such as the following:

1. the number of leaf nodes is maximized, whose sem features are determinate and equals their respective $bsem$s;

2. coreferent nodes tend to be either potential sisters below a concentrator or deleted. This allows most constraints to be stated in terms of direct relationships of mother-, daughter- or sisterhood. Such proximity and concentration is rather opportune because we are dealing simply with *potential* relationships as nodes will usually be deleted. In other words, our constraints aim mostly at ruling out undesired relations rather than establishing correct ones. The latter must remain a matter of choice.

It is in order to define which are the best candidates for concentrators. Having different concentrators in equivalent alternative realizations, such as "a <u>ballerina</u>", "a female <u>dancer</u>" or "a dancing <u>woman</u>" (hypothetical concentrators are underlined), would be rather hampering, since the task of assigning "concentratingness" would then be fatally bound to lexicalization disjunction itself and not much determinacy could possibly be derived ahead of committing to this or that realization. In face of that, the natural candidate must be something that remains constant all along, namely the *article*. Certainly, what specific article and, among others, whether to generate a definite/anaphoric or indefinite/first-time referring expression is also a matter of choice, but not pertaining to lexicalization. For the sake of simplicity and scope, let us stick to the case of indefinite articles, keeping in mind that possible extensions to our model to cope with (especially definite anaphoric) referring expression generation shall certainly require some revisions.

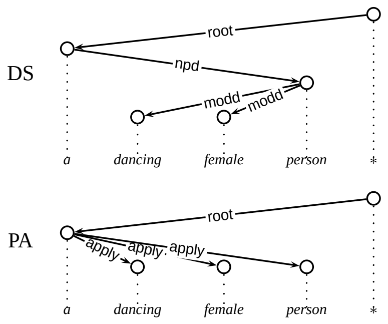

Figure 4: new PA and DS analyses for "a dancing female person". An asterisk stands for the root node

Electing articles for concentrators means that they now *directly* dominate their respective nouns and accompanying modifiers on dimension PA as shown in Figure 4 for "a dancing female person". One new edge label $apply$ is postulated to connect concentrators with their complements, the following invariants holding:

1. for every node v, $hook(v) = holes(v)(apply)$, i.e. only coreferent nodes are linked by $apply$ edges;

2. every concentrator lexical entry provides a valency allowing any number of outgoing $apply$ edges, though requiring at least one.

Roughly speaking, the intuition behind this new PA design is that the occurrence of a lexical (as opposed to grammatical) word corresponds to the evaluation of a lambda expression, resulting in a fresh *unary* predicate built from the $basesem$ of the word/node and the sems of its children. In turn, every $apply$ edge denotes the application of one such predicate to the variable/referent of a concentrator. In fact, even verbs might be treated analogously if *Infl* constituents were modelled, constituting the concentrators of verb base forms. Also useful is the intuition that PA abstracts over most morphosyntactic oppositions, such as that between nouns and adjectives, which figure as equals there. The subordination of the latter word class to the former becomes a strictly syntactic phenomenon or, in any case, other dimensions' affairs.

Dimension DS is all about such oppositions, however, and should remain much the same except that the design is rather simplified if DS maintains concentrator dominance. As a result, articles must stand as heads of noun — or rather, de-

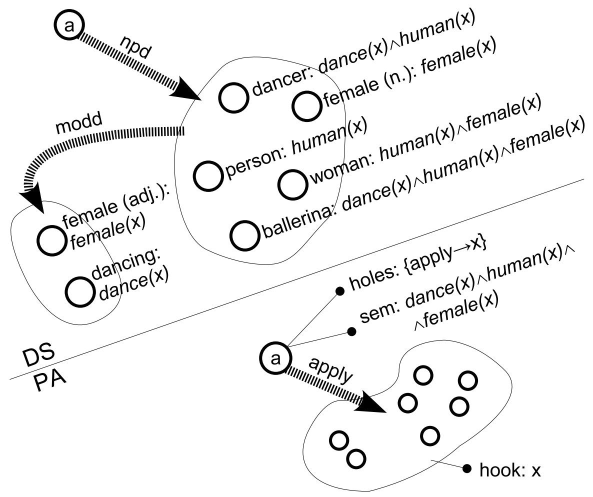
a
npd
dancer: dance(x)∧human(x)
female (n.): female(x)
modd
person: human(x)
woman: human(x)∧female(x)
ballerina: dance(x)∧human(x)∧female(x)
female (adj.): female(x)
dancing: dance(x)
holes: {apply→x}
sem: dance(x)∧human(x)∧ ∧female(x)
DS
PA
a
apply
hook: x

Figure 5: Starting conditions with perfectly floating nodes in the lexicalization of "a ballerina" and its paraphrases

terminer — phrases, which is not an unheard-of approach, just unprecedented in XDG. Naturally, standard syntactic structures should appear below determiners, as exemplified in Figure 4. Granted this, the flatness of PA and its relation to DS can straightforwardly be accomplished by the application of XDK library principles *Climbing*, whereby PA is constrained to be a flattening of DS, and *Barriers*, whereby concentrators are set as obstacles to climbing by means of special lexical features. Figure 5 thus illustrates the starting conditions for the lexicalization of "a ballerina" and its paraphrases, including the $bsem$s of nodes. Notice that we have created distinct nodes for different parts of speech of one same word, "female". The relevance of this measure shall be clarified along this section as we develop this example.

Fighting over-redundancy. We currently employ two constraints to avoid over-redundancy. The first is complete in that its declarative semantics already sums up all we desire to express in that matter, while the other is redundant, incomplete, but supplied to improve propagation.

The complete constraint is imposed between every node and each of its potential daughters. Apart from overhead reasons, it might as well be imposed between every pair of nodes. However, the set of potential daughters of a node v is best approximated by function $dcands$ thus:

$$\begin{gathered} dcands(v) \\ = \\ \left(\bigcup \{\langle x \rangle : \, x \in ran(holes(v))\}\right) - \{v\}\,, \end{gathered}$$

where $\langle x \rangle$ denotes the coreference class of variable x; and $ran(f)$, the range of function f. It is worth noticing that in generation $dcands$ is known at model creation.

Given a node u and a potential daughter $v \in dcands(u)$, this constraint involves hypothesizing what the actual semantic content of u would be like if v were *not* among its daughters. Let $hds_v(u)$ and $hsem_v(u)$ be respectively the $\underline{h}$ypothetical set of daughters of u counting v out and its "actual" semantic content in that case, which can be defined thus:

$$hds_v(u) = \{w : \, u \rightarrow_{PA} w\} - \{v\}$$

and

$$\begin{aligned} hsem_v(u) = bsem(u) \wedge \\ \wedge \bigwedge \{sem(w) : \, w \in hds_v(u)\}\,. \end{aligned} \tag{4}$$

The constraint consists of ensuring that, if the *actual* semantic content of the potential daughter v would be subsumed by the *hypothetical* semantic content of u, then v can never be a daughter of u. In other words, each daughter of u must make a difference. Formally, we have the following:

$$(hsem_v(u) \Rightarrow sem(v)) \rightarrow \neg\,(u \rightarrow_{PA} v) \tag{5}$$

where the two implication symbols, $\Rightarrow$ and $\rightarrow$ have the same interpretation in this logic statement, but are nonetheless distinguished because their implementations are radically different as shall be discussed in Section 4. Constraint (5) is especially active after some choices have been made. Suppose, in our "a ballerina" example, that "dancing" is the only word selected so far for lexicalization. Let u and v be respectively the nodes for "a" and "dancing". In this case, the consequent in (5) is false and so must be the antecedent $hsem_v(u) \Rightarrow dance(x)$, which implies that $hsem_v(u)$ can never "contain" the literal $dance(x)$. From (4) and the fact that articles have neutral base semantics — i.e. $bsem(u) = true$ — it follows that all further daughters of u must not imply $dance(x)$. As that does not hold for "ballerina" and "dancer", these nodes are ruled out as daughters of u and thus deleted for lack of mothers. Conversely, if "ballerina" had been selected in the first place, (5) would trivially detect the redundancy of all other words and analogously entail their deletion.

In turn, the redundant constraint ensures that, for every pair of coreferent nodes u and $v \in \langle upvar(u) \rangle$, if the actual semantic content of v is subsumed by u, then they can never be *sisters*. Formally:

$$\begin{aligned}(sem(u) \Rightarrow sem(v)) \rightarrow \\ (v \notin sisters_{PA}(u))\,.\end{aligned} \tag{6}$$

This constraint is remarkable for being active even in the absence of choice since it is established between potential sisters, which usually have their sems sufficiently, if not completely, determined. Surprisingly enough, the main effect of (6) is on syntax, by constraining alliances on DS. As our new version of the XDK's *Climbing* principle is now aware of sisterhood constraints, it will constrain every node on PA to have as a mother on DS either its current PA mother or some node belonging to one of its PA sister trees[3]. In ground terms, when (6) detects that e.g. "woman" subsumes "female (adj./n.)" and constrains them not to be sisters on PA, the *Climbing* principle will rule out "woman" as a potential DS mother of "female (adj.)". It is worth mentioning that once $v \notin sisters_D(u)$ is imposed, our sisterhood constraints entail $u \notin sisters_D(v)$.

Redundant compositionality constraints. Although a complete statement of semantic compositionality is given by Equation 2, we introduce two redundant constraints to improve propagation. The first of them attempts to advance detection of nodes whose semantic contribution is strictly required even before the sem features of their mothers become sufficiently constrained. It does so by means of an strategy analogous to that of (5), namely by hypothesizing, for every node v, what the *total* semantic content would be like if v were deleted. Let $root$, $hdown_v(u)$ and $htotsem_v$ be respectively the root node, the set of nodes directly or indirectly below u counting v out, and the total semantic content supposing v is deleted, which can be defined thus:

$$hdown_v(u) = down_{PA}(u) - \{v\}$$

and

$$htotsem_v = \bigwedge \{sem(u) : u \in hdown_v(root)\}\,.$$

[3]If the *subgraphs* option is active, which is the case here.

The constraint can be formally expressed thus:

$$deleted(v) \rightarrow (htotsem_v \Rightarrow sem(v))\,. \tag{7}$$

Unfortunately, (7) is not of much use in our current example, better applying to cases where there are a greater number of alternative inner nodes. For example, in the lexicalization of (1), this constraint was immediately able to infer that "lovely" must not be deleted since it was the sole node contributing $lovely(y)$.

The second redundant compositionality constraint attempts to advance detection of nodes not counting on enough potential sisters to fulfill the actual semantic content of their mothers. To this end, for every node v, the following constraint is imposed:

$$\begin{gathered}\bigwedge \{sem(u) : u \rightarrow_{PA} v\} \\ = \\ \left(\begin{gathered} \bigwedge \{bsem(u) : u \rightarrow_{PA} v\} \\ \wedge \\ \bigwedge \{sem(u) : u \in eqsis_{PA}(v)\} \end{gathered} \right),\end{gathered} \tag{8}$$

where

$$eqsis_D(v) = \begin{cases} \varnothing, & \text{iff } v \text{ is deleted on } D \\ sisters_D(v) \cup \{v\}\,, & \text{else.} \end{cases} \tag{9}$$

which reads "the actual semantic content of the mothers of a node is equal to their base semantic content in conjunction with the actual semantic content of this node and its sisters". It is worth noticing that, when v is deleted, both $\{u : u \rightarrow_{PA} v\}$ and $eqsis_{PA}(v)$ become empty so that (8) still holds. This constraint is especially interesting because our new versions of principles *Climbing* and *Barriers*, which hold between DS and PA, propagate $sisters$ constraints in both directions. In association with (6) and (8), these principles promote an interesting interplay between syntax and semantics. Resuming our example, let v be node "female (n.)". Before any selection is performed, constraint (6) infers that only "dancing", "person" and "dancer" can be sisters to v on PA and thus (now due to *Climbing*) daughters to v on DS. They cannot be mothers to v because its valency on DS and *Climbing* are enough to establish that, if v has any mother at all on DS, it is "a". Again taking the DS valency of v into account, it is possible to infer that, if v has any daughter at all on DS, it is "dancing", i.e. the only adjective in the original set of candidate daughters. It is the new sisterhood-aware version of *Barriers* that propagates this new piece of information back to PA. This principle now knows that the sisters of v on PA must come from either (i) the tree below v on DS, (ii) one of its DS sister trees or (iii) some DS tree whose root belongs to $eqsis_{DS}(inter)$ for some node $inter$ appearing — on DS — between v and one of its mothers on PA. In our example, (ii) and (iii) are known to be empty sets, while (i) is at most "dancing". Consequently, "dancing" is the only potential PA sister of v. Now (8) is finally able to contribute. As "a" is the only possible DS mother of v and any article has empty basic semantics, one is entitled to equate $\bigwedge\{sem(u) : u \rightarrow_{PA} v\}$ to $\bigwedge\{sem(u) : u \in eqsis_{PA}(v)\}$. Even though it is not known whether v will ever have mothers or daughters, (8) knows that the left-hand side of the equation yields either the whole intended semantics or nothing, while the right-hand side yields either nothing or at most $dance(x) \wedge female(x)$. Therefore, the only solution to the equation is nothing on both sides, implying that $eqsis_{PA}(v)$ is empty and thus v is deleted by definition (9).

Such strong interplay is only possible because we have created distinct nodes for the different parts of speech — or rather, the two different DS valencies — of "female". With somewhat more complicated, heavier constraints it would be possible to have the same propagation for one single node selecting from different parts of speech. Notwithstanding, that does not seem worth the effort because a model creation algorithm would be perfectly able to detect the diverging DS valencies, create as many nodes as needed and distribute the right lexical entries among them.

4 Implementation and Performance Remarks

The ideas presented in Section 3 were fully implemented in a development branch of the XDK. As with the original XDK, all development is based on the multiparadigm programming system Mozart[4].

The implementation closely follows the original CP approach of the XDK and strongly reflects the constraints we have presented after some rather standard transformations to CP, namely:

- variable identifiers in $hook$s and $holes$, as well as all semantic input literals such as $human(x)$ and $tap(e, x, y)$, are encoded as integer values. Features $bsem$/sem are implemented as set constants/variables of such integers;

- logic conjunction $\wedge$ is thus modelled by set union $\cup$. Each "big" conjunction is reduced to the form $\bigwedge\{f(v) : v \in V\}$, where V is a set variable of integer-encoded node identifiers, and modelled by a *union selection constraint* $\bigcup\langle f(1)\ \ f(2)\ \dots f(M)\rangle\,[V]$, where M is the maximum node identifier and which constrains its result — a set variable — to be the union of $f(v)$ for all $v \in V$;

- implications of the form $x \Rightarrow y$ are implemented as $y \subseteq x$, while those of the form $x \rightarrow y$ as $reify(x) \leq reify(y)$, where the result of $reify(x)$ is an integer-encoded boolean variable constrained to coincide with the truth-value of expression x.

[4]http://www.mozart-oz.org

Our branch of the XDK now counts on two new principles, namely *(i) Delete*, which requires the *Graph* principle, creates doubles for the node attributes introduced by the latter, providing the illusion of deletion, and introduces features for sisterhood constraints; and *(ii) Compsem*, imposing all constraints described in Section 3.

A few preliminary proof-of-concept experiments were carried out with input similar to (1) and linguistically and combinatorially analogous to our "ballerina" example. In all of them, the system was able to generate all paraphrases with no failed state (backtracking) in search, which means that propagation was maximal for all cases. Although our design supports more complex linguistic constructs such as relative clauses and preposition phrases and is expected to behave similarly for those cases, we have not made any such experiments so far. This is so because we are currently prioritizing the issue of model creation and coverage of other generation tasks.

5 Conclusions and Future Work

In this paper we have presented the results of the very first steps towards the application of XDG to Natural Language Generation, hopefully in an integrated architecture. Our main contribution and focus was a formulation of lexicalization disjunction in XDG terms, preserving the good properties of modularity and extensibility while achieving good propagation. We also hope to have demonstrated how strong the interplay between linguistic dimensions can be in XDG. As basic issues as the very nature of input were discussed also as an evidence that there is still a long way to go. We are currently working on extending our design to cover other generation tasks than lexicalization and perform model creation.

Acknowledgements

We would like to thank Claire Gardent and Denys Duchier, for all their invaluable insights and comments, and group Langue & Dialogue and Brazilian agencies CNPq and CAPES, for funding this research project and travel to COLING/ACL.

References

Copestake, A. A., Lascarides, A., and Flickinger, D. (2001). An algebra for semantic construction in constraint-based grammars. In *Meeting of the Association for Computational Linguistics*, pages 132–139.

Debusmann, R. (2004). Multiword expressions as dependency subgraphs. In *Proceedings of the ACL 2004 Workshop on Multiword Expressions: Integrating Processing*, Barcelona/ESP.

Debusmann, R., Duchier, D., Koller, A., Kuhlmann, M., Smolka, G., and Thater, S. (2004a). A relational syntax-semantics interface based on dependency grammar. In *Proceedings of the COLING 2004 Conference*, Geneva/SUI.

Debusmann, R., Duchier, D., and Kruijff, G.-J. M. (2004b). Extensible dependency grammar: A new methodology. In *Proceedings of the COLING 2004 Workshop on Recent Advances in Dependency Grammar*, Geneva/SUI.

Debusmann, R., Duchier, D., and Niehren, J. (2004c). The xdg grammar development kit. In *Proceedings of the MOZ04 Conference*, volume 3389 of *Lecture Notes in Computer Science*, pages 190–201, Charleroi/BEL. Springer.

Debusmann, R., Duchier, D., and Rossberg, A. (2005). Modular Grammar Design with Typed Parametric Principles. In *Proceedings of FG-MOL 2005*, Edinburgh/UK.

Debusmann, R. and Smolka, G. (2006). Multi-dimensional dependency grammar as multigraph description. In *Proceedings of FLAIRS-19*, Melbourne Beach/US. AAAI.

Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press.

Van Roy, P. and Haridi, S. (2004). *Concepts, Techniques, and Models of Computer Programming*. MIT Press.