Extensible Dependency Grammar: A New Methodology

Ralph Debusmann
Programming Systems Lab
Saarland University
Postfach 15 11 50
66041 Saarbrücken
Germany
rade@ps.uni-sb.de

Denys Duchier
Équipe Calligramme
LORIA – UMR 7503
Campus Scientifique, B. P. 239
54506 Vandœuvre lès Nancy CEDEX
France
duchier@loria.fr

Geert-Jan M. Kruijff
Computational Linguistics
Saarland University
Postfach 15 11 50
66041 Saarbrücken
Germany
gj@coli.uni-sb.de

Abstract

This paper introduces the new grammar formalism of Extensible Dependency Grammar (XDG), and emphasizes the benefits of its methodology of explaining complex phenomena by interaction of simple principles on multiple dimensions of linguistic description. This has the potential to increase modularity with respect to linguistic description and grammar engineering, and to facilitate concurrent processing and the treatment of ambiguity.

1 Introduction

We introduce the new grammar formalism of Extensible Dependency Grammar (XDG). In XDG, complex phenomena arise out of the interaction of simple principles on multiple dimensions of linguistic description. In this paper, we point out how this novel methodology positions XDG in between multi-stratal approaches like LFG and MTT, and mono-stratal ones like HPSG, attempting to combine their benefits and avoid their problems.

It is the division of linguistic analyses into different dimensions which makes XDG multi-stratal. On the other, XDG is mono-stratal in that its principles interact to constrain all dimensions simultaneously. XDG combines the benefits of these two positions, and attempts to circumvent their problems. From multi-stratal approaches, XDG adopts a high degree of *modularity*, both with respect to linguistic description as well as for grammar engineering. This also facilitates the statement of cross-linguistic generalizations. XDG avoids the problem of placing too high a burden on the interfaces, and allows interactions between all and not only adjacent dimensions. From mono-stratal approaches, XDG adopts a high degree of *integration*, facilitating concurrent processing and the treatment of ambiguity. At the same time, XDG does not lose its modularity.

XDG is a descendant of Topological Dependency Grammar (TDG) (Duchier and Debusmann, 2001), pushing the underlying methodology further by generalizing it in two aspects:

- number of dimensions: two in TDG (ID and LP), arbitrary many in XDG
- set of principles: fixed in TDG, extensible principle library in XDG

The structure of this paper is as follows: In §2, we introduce XDG formally, and also the XDG solver used for parsing and generation. In §3, we introduce a number of XDG principles informally, before making use of them in an idealized example grammar in §4. In §5 we argue why XDG has the potential to be an improvement over multi-stratal and mono-stratal approaches, before we conclude in §6.

2 Extensible Dependency Grammar

In this section, we introduce XDG formally and mention briefly the constraint-based XDG solver for parsing and generation.

2.1 Formalization

Formally, an XDG grammar is built up of dimensions, a lexicon and principles, and characterizes a set of well-formed analyses.

A *dimension* is a tuple $D = (Lab, Fea, Val, Pri)$ of a set *Lab* of edge labels, a set *Fea* of features, a set *Val* of feature values, and a set of one-dimensional principles *Pri*. A *lexicon* for the dimension D is a set $Lex \subseteq Fea \rightarrow Val$ of total feature assignments called lexical entries. An analysis on dimension D is a triple (V, E, F) of a set V of nodes, a set $E \subseteq V \times V \times Lab$ of directed labeled edges, and an assignment $F : V \rightarrow (Fea \rightarrow Val)$ of lexical entries to nodes. V and E form a graph. We write Ana_D for the set of all possible analyses on dimension D. The principles characterize subsets of Ana_D. We assume that the elements of *Pri* are finite representations of such subsets.

An *XDG grammar* $((Lab_i, Fea_i, Val_i, Pri_i)_{i=1}^n, Pri, Lex)$ consists of n dimensions, multi-dimensional principles *Pri*, and a lexicon *Lex*. An *XDG analysis* $(V, E_i, F_i)_{i=1}^n$ is an element of $Ana = Ana_1 \times \cdots \times Ana_n$ where all dimensions share the same set of nodes V. We call a dimension of a grammar *grammar dimension*.

Multi-dimensional principles specify subsets of *Ana*, i.e. of tuples of analyses for the individual dimensions. The lexicon $Lex \subseteq Lex_1 \times \cdots \times Lex_n$ constrains all dimensions at once, thereby synchronizing them. An XDG analysis is licensed by *Lex* iff $(F_1(v), \ldots, F_n(v)) \in Lex$ for every node $v \in V$.

In order to compute analyses for a given input, we employ a set of *input constraints* (*Inp*), which again specify a subset of *Ana*. XDG solving then amounts to finding elements of *Ana* that are licensed by *Lex*, and consistent with *Inp* and *Pri*. The input constraints determine whether XDG solving is to be used for parsing or generation. For parsing, they specify a sequence of words, and for generation, a multiset of semantic literals.

2.2 Solver

XDG solving has a natural reading as a constraint satisfaction problem (CSP) on finite sets of integers, where well-formed analyses correspond to the solutions of the CSP (Duchier, 2003). We have implemented an XDG solver using the Mozart-Oz programming system (Mozart Consortium, 2004).

XDG solving operates on all dimensions concurrently. This means that the solver can infer information about one dimension from information on another, if there is either a multi-dimensional principle linking the two dimensions, or by the synchronization induced by the lexical entries. For instance, not only can syntactic information trigger inferences in syntax, but also vice versa.

Because XDG allows us to write grammars with completely free word order, XDG solving is an NP-complete problem (Koller and Striegnitz, 2002). This means that the worst-case complexity of the solver is exponential. The average-case complexity of many smaller-scale grammars that we have experimented with seems polynomial, but it remains to be seen whether we can scale this up to large-scale grammars.

3 Principles

The well-formedness conditions of XDG analyses are stipulated by *principles*. Principles are parametrizable, e.g. by the dimensions on which they are applied, or by lexical features. They can be lexicalized or non-lexicalized, and can be one-dimensional or multi-dimensional. Principles are taken from an extensible *principle library*. So far, the set of possible principles is unrestricted, and to find restrictions for them is a topic for future research. In the following two subsections, we introduce some of the most important one-dimensional and multi-dimensional principles.

3.1 Tree principle

$tree(i)$ The analysis on dimension i must be a tree.

The *tree principle* is non-lexicalized and parametrized by the dimension i.

3.2 Dag principle

$dag(i)$ The analysis on dimension i must be a directed acyclic graph.

The *dag principle* is non-lexicalized and parametrized by the dimension i.

3.3 Valency principle

$valency(i, \mathsf{in}_i, \mathsf{out}_i)$ All nodes on dimension i must satisfy their in and out specifications.

The *valency principle* is lexicalized and serves to lexically describe dependency graphs. It is parametrized by the dimension i, the *in specification* in_i and the *out specification* out_i. For each node, in_i stipulates the licensed incoming edges, and out_i the licensed outgoing edges.

In the example grammar lexicon part in Figure 1 below, the in specification is $\mathsf{in}_{\mathrm{ID}}$ and $\mathsf{out}_{\mathrm{ID}}$ is the out specification on the ID dimension. For the common noun *Roman*, the in specification licenses zero or one incoming edges labeled subj (subj?), and zero or one incoming edges labeled obj (obj?). The out specification requires precisely one outgoing edge labeled det (det!).

3.4 Government principle

$government(i, \mathsf{cases}_i, \mathsf{govern}_i)$ All edges in dimension i must satisfy the government specification of the mother.

The *government principle* is lexicalized. Its purpose is to constrain the case feature of a dependent.[1] It is parametrized by the dimension i, the *cases specification* cases_i and the *government specification* govern. cases assigns to each word a set of possible cases, and govern a mapping from labels to sets of cases.

In Figure 1, the cases specification for the determiner *den* is $\{\mathsf{acc}\}$ (i.e. it can only be accusative). By its government specification, the finite verb *versucht* requires its subject to exhibit nominative case ($\mathsf{subj} \mapsto \{\mathsf{nom}\}$).

[1] We restrict ourselves to the case feature only for simplicity. In a fully-fledged grammar, the government principle would be used to constrain also other morphological aspects like number, person and gender.

3.5 Agreement principle

agreement$(i, \mathsf{cases}_i, \mathsf{agree}_i)$ All edges in dimension i must satisfy the agreement specification of the mother.

The *agreement principle* is lexicalized. Its purpose is to enforce the case agreement of a daughter.[2] It is parametrized by dimension i, the lexical *cases specification* cases_i, assigning to each word a set of possible cases, and the *agreement specification* agree_i, assigning to each word a set of labels.

As an example, in Figure 1, the agreement specification for the common noun *Roman* is $\{\mathsf{det}\}$, i.e. the case of the common noun must agree with its determiner.

3.6 Order principle.

order$(i, \mathsf{on}_i, \prec_i)$ On dimension i, 1) each node must satisfy its node labels specification, 2) the order of the daughters of each node must be compatible with $\prec_i$, and 3) the node itself must be ordered correctly with respect to its daughters (using its node label).

The *order principle* is lexicalized. It is parametrized by the dimension i, the *node labels specification* on_i mapping each node to set of labels from Lab_i, and the total order $\prec_i$ on Lab_i.

Assuming the node labels specification given in Figure 2, and the total order in (5), the tree in (11) satisfies the order principle. For instance for the node *versucht*: 1) The node label of *versucht* is lbf, satisfying the node labels specification. 2) The order of the daughters *Roman* (under the edge labeled vf), *Peter* (mf) and *lesen* (rbf) is compatible with the total order prescribing $\mathsf{vf} \prec \mathsf{mf} \prec \mathsf{rbf}$. 3) The node *versucht* itself is ordered correctly with respect to its daughters (the total order prescribes $\mathsf{vf} \prec \mathsf{lbf} \prec \mathsf{mf}$).

3.7 Projectivity principle

projectivity(i) The analysis on dimension i must be projective.

The *projectivity principle* is non-lexicalized. Its purpose is to exclude non-projective analyses.[3] It is parametrized by dimension i.

3.8 Climbing principle

climbing(i, j) The graph on dimension i must be flatter than the graph on dimension j.

The *climbing principle* is non-lexicalized and two-dimensional. It is parametrized by the two dimensions i and j.

For instance, the tree in (11) is flatter than the corresponding tree in (10).

[2] Again, we restrict ourselves to case for simplicity.

[3] The projectivity principle of course only makes sense in combination with the order principle.

3.9 Linking principle

linking$(i, j, \mathsf{link}_{i,j})$ All edges on dimension i must satisfy the linking specification of the mother.

The *linking principle* is lexicalized and two-dimensional. It is parametrized by the two dimensions i and j, and by the *linking specification* $\mathsf{link}_{i,j}$, mapping labels from Lab_i to sets of labels from Lab_j. Its purpose is to specify how dependents on dimension i are realized by (or linked to) dependents on dimension j.

In the lexicon part in Figure 3, the linking specification for the transitive verb *lesen* requires that its agent on the PA dimension must be realized by a subject ($\mathsf{ag} \mapsto \{\mathsf{subj}\}$), and the patient by an object ($\mathsf{pat} \mapsto \{\mathsf{obj}\}$).

4 Example grammar

In this section, we elucidate XDG with an example grammar fragment for German. With it, we demonstrate three aspects of the methodology of XDG:

- How complex phenomena such as topicalization and control arise by the interaction of simple principles on different dimensions of linguistic description.
- How the high degree of integration helps to reduce ambiguity.
- How the high degree of modularity facilitates the statement of cross-linguistic generalizations.

Note that this grammar fragment is an idealized example, and does not make any claims about XDG as a *grammar theory*. Its purpose is solely to substantiate our points about XDG as a *framework*.

4.1 Dimensions

The grammar fragment make use of two dimensions: Immediate Dominance (ID) and Linear Precedence (LP). The models on the ID dimension are unordered, syntactic dependency trees whose edge labels correspond to syntactic functions like *subject* and *object*. On the LP dimension, the models are ordered, projective topological dependency trees whose edge labels are topological fields like *Vorfeld* and *Mittelfeld*.

4.2 Labels

The set Lab_{ID} of labels on the ID dimension is:

$$Lab_{\mathrm{ID}} = \{\mathsf{det}, \mathsf{subj}, \mathsf{obj}, \mathsf{vinf}, \mathsf{part}\} \qquad (1)$$

These correspond resp. to determiner, subject, object, infinitive verbal complement, and particle.

The set Lab_{LP} of labels on the LP dimension is:

$$Lab_{\mathrm{LP}} = \{\mathsf{detf}, \mathsf{nounf}, \mathsf{vf}, \mathsf{lbf}, \mathsf{mf}, \mathsf{partf}, \mathsf{rbf}\} \qquad (2)$$

Corresponding resp. to determiner field, noun field, Vorfeld, left bracket field, Mittelfeld, particle field, and right bracket field.

4.3 Principles

On the ID dimension, we make use of the following one-dimensional principles:

$$\begin{array}{l} \mathit{tree}(\mathrm{ID}) \\ \mathit{valency}(\mathrm{ID}, \mathsf{in}_{\mathrm{ID}}, \mathsf{out}_{\mathrm{ID}}) \\ \mathit{government}(\mathrm{ID}, \mathsf{cases}_{\mathrm{ID}}, \mathsf{govern}_{\mathrm{ID}}) \\ \mathit{agreement}(\mathrm{ID}, \mathsf{cases}_{\mathrm{ID}}, \mathsf{agree}_{\mathrm{ID}}) \end{array} \quad (3)$$

The LP dimension uses the following principles:

$$\begin{array}{l} \mathit{tree}(\mathrm{LP}) \\ \mathit{valency}(\mathrm{LP}, \mathsf{in}_{\mathrm{LP}}, \mathsf{out}_{\mathrm{LP}}) \\ \mathit{order}(\mathrm{LP}, \mathsf{on}_{\mathrm{LP}}, \prec_{\mathrm{LP}}) \\ \mathit{projectivity}(\mathrm{LP}) \end{array} \quad (4)$$

where the total order $\prec_{\mathrm{LP}}$ is defined as:

$$\mathsf{detf} \prec \mathsf{nounf} \prec \mathsf{vf} \prec \mathsf{lbf} \prec \mathsf{mf} \prec \mathsf{partf} \prec \mathsf{rbf} \quad (5)$$

We make use of the following multi-dimensional principles:

$$\begin{array}{l} \mathit{climbing}(\mathrm{LP}, \mathrm{ID}) \\ \mathit{linking}(\mathrm{LP}, \mathrm{ID}) \end{array} \quad (6)$$

4.4 Lexicon

We split the lexicon into two parts. The ID and LP parts are displayed resp. in Figure 1[4] and Figure 2. The LP part includes also the linking specification for the LP,ID-application of the linking principle.[5]

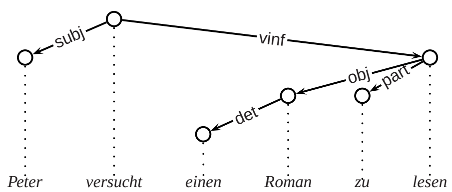
4.5 Government and agreement

Our first example is the following sentence:

Peter	*versucht*	*einen*	*Roman*	*zu*	*lesen.*	
Peter	*tries*	*a$_{acc}$*	*novel*	*to*	*read.*	(7)

Peter tries to read a novel.

We display the ID analysis of the sentence below:



(8)

Here, *Peter* is the subject of *versucht*. *lesen* is the infinitival verbal complement of *versucht*, *zu* the particle of *lesen*, and *Roman* the object of *lesen*. Finally, *einen* is the determiner of *Roman*.

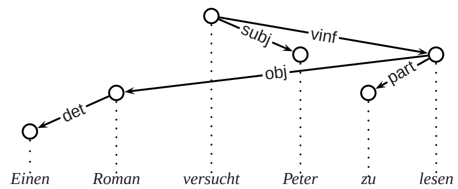
Under our example grammar, the sentence is unambiguous, i.e. the given ID tree is the only possible one. Other ID trees are ruled out by the interaction of the principles on the ID dimension. For instance, the government and agreement principles conspire to rule out the reading where *Roman* is the subject of *versucht* (and *Peter* the object). How? By the agreement principle, *Roman* must be accusative, since it agrees with its accusative determiner *einen*. By the government principle, the subject of *versucht* must be nominative, and the object of *lesen* accusative. Thus *Roman*, by virtue of being accusative, cannot become the subject of *versucht*. The only other option for it is to become the object of *lesen*. Consequently, *Peter*, which is unspecified for case, must become the subject of *versuchen* (*versuchen* must have a subject by the valency principle).

4.6 Topicalization

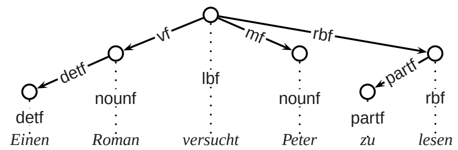
Our second example is a case of topicalization, where the object has moved into the Vorfeld, to the left of the finite verb:

Einen Roman versucht Peter zu lesen. (9)

Here is the ID tree and the LP tree analysis:



(10)



(11)

The ID tree analysis is the same as before, except that the words are shown in different positions. In the LP tree, *Roman* is in the Vorfeld of *versucht*, *Peter* in the Mittelfeld, and *lesen* in the right bracket field. *versucht* itself is (by its node label) in the left bracket field. Moreover, *Einen* is in the determiner field of *Roman*, and *zu* in the particle field of *lesen*.

Again, this is an example demonstrating how complex phenomena (here: topicalization) are explained by the interaction of simple principles. Topicalization does not have to explicitly taken care of, it is rather a consequence of the interacting principles. Here, the valency, projectivity and climbing principles conspire to bring about the "climbing up" of the NP *Einen Roman* from being the daughter of *lesen* in the ID tree to being the daughter of *versucht* in the LP tree: The out specification of *lesen* does not license any outgoing edge. Hence, *Roman* must

[4] Here, _ stands for "don't care", this means e.g. for the verb *versucht* that it has unspecified case.

[5] We do not make use of the linking specification for the German grammar fragment (the mappings are all empty), but we will do so as we switch to Dutch in §4.8 below.

	$\mathsf{in}_{\mathrm{ID}}$	$\mathsf{out}_{\mathrm{ID}}$	$\mathsf{cases}_{\mathrm{ID}}$	$\mathsf{govern}_{\mathrm{ID}}$	$\mathsf{agree}_{\mathrm{ID}}$
den	$\{\mathsf{det?}\}$	$\{\}$	$\{\mathsf{acc}\}$	$\{\}$	$\{\}$
Roman	$\{\mathsf{subj?}, \mathsf{obj?}\}$	$\{\mathsf{det!}\}$	$\{\mathsf{nom}, \mathsf{dat}, \mathsf{acc}\}$	$\{\}$	$\{\mathsf{det}\}$
Peter	$\{\mathsf{subj?}, \mathsf{obj?}\}$	$\{\}$	$\{\mathsf{nom}, \mathsf{dat}, \mathsf{acc}\}$	$\{\}$	$\{\}$
versucht	$\{\}$	$\{\mathsf{subj!}, \mathsf{vinf!}\}$	_	$\{\mathsf{subj} \mapsto \{\mathsf{nom}\}\}$	$\{\}$
zu	$\{\mathsf{part?}\}$	$\{\}$	_	$\{\}$	$\{\}$
lesen	$\{\mathsf{vinf?}\}$	$\{\mathsf{obj!}\}$	_	$\{\mathsf{obj} \mapsto \{\mathsf{acc}\}\}$	$\{\}$

Figure 1: Lexicon for the example grammar fragment, ID part

	$\mathsf{in}_{\mathrm{LP}}$	$\mathsf{out}_{\mathrm{LP}}$	$\mathsf{on}_{\mathrm{LP}}$	$\mathsf{link}_{\mathrm{LP,ID}}$
den	$\{\mathsf{detf?}\}$	$\{\}$	$\{\mathsf{detf}\}$	$\{\}$
Roman	$\{\mathsf{vf?}, \mathsf{mf?}\}$	$\{\mathsf{detf!}\}$	$\{\mathsf{nounf}\}$	$\{\}$
Peter	$\{\mathsf{vf?}, \mathsf{mf?}\}$	$\{\}$	$\{\mathsf{nounf}\}$	$\{\}$
versucht	$\{\}$	$\{\mathsf{vf?}, \mathsf{mf}{*}, \mathsf{rbf?}\}$	$\{\mathsf{lbf}\}$	$\{\}$
zu	$\{\mathsf{partf?}\}$	$\{\}$	$\{\mathsf{partf}\}$	$\{\}$
lesen	$\{\mathsf{rbf?}\}$	$\{\}$	$\{\mathsf{rbf}\}$	$\{\}$

Figure 2: Lexicon for the example grammar fragment, LP part

become the daughter of another node. The only possibility is *versucht*. The determiner *Einen* must then also "climb up" because *Roman* is its only possible mother. The result is an LP tree which is flatter with respect to the ID tree. The LP tree is also projective. If it were not be flatter, then it would be non-projective, and ruled out by the projectivity principle.

4.7 Negative example

Our third example is a negative example, i.e. an ungrammatical sentence:

$*$*Peter einen Roman versucht zu lesen.* (12)

This example is perfectly legal on the unordered ID dimension, but has no model on the LP dimension. Why? Because by its LP out specification, the finite verb *versucht* allows only one dependent to the left of it (in its Vorfeld), and here we have two. The interesting aspect of this example is that although we can find a well-formed ID tree for it, this ID tree is never actually generated. The interactions of the principles, viz. here of the principles on the LP dimension, rule out the sentence *before* any full ID analysis has been found.

4.8 From German to Dutch

For the fourth example, we switch from German to Dutch. We will show how to use the lexicon to concisely capture an important cross-linguistic generalization. We keep the same grammar as before, but with two changes, arising from the lesser degree of inflection and the higher reliance on word order in Dutch:

- The determiner *een* is not case-marked but can be either nominative, dative or accusative: $\mathsf{cases}_{\mathrm{ID}} = \{\mathsf{nom}, \mathsf{dat}, \mathsf{acc}\}$.
- The Vorfeld of the finite verb *probeert* cannot be occupied by an object (but only by an object): $\mathsf{link}_{\mathrm{LP,ID}} = \{\mathsf{vf} \mapsto \{\mathsf{subj}\}\}$.

Now to the example, a Dutch translation of (7):

Peter probeert een roman te lezen.
Peter tries a novel to read. (13)
Peter tries to read a novel.

We get only one analysis on the ID dimension, where *Peter* is the subject and *roman* the object. An analysis where *Peter* is the object of *lezen* and *roman* the subject of *probeert* is impossible, as in the German example. The difference is, however, how this analysis is excluded. In German, the accusative inflection of the determiner *einen* triggered the agreement and the government principle to rule it out. In Dutch, the determiner is not inflected. The unwanted analysis is excluded on the grounds of word order instead: By the linking principle, the Vorfeld of *probeert* must be filled by a subject, and not by an object. That means that *Peter* in the Vorfeld (to the left of *probeert*) must be a subject, and consequently, the only other choice for *roman* is that it becomes the object of *lezen*.

4.9 Predicate-Argument Structure

So far, our example grammar fragment was confined to syntax. In this section, we emphasize the extensibility aspect of XDG by showing how it allows us to extend the grammar with another dimension, Predicate-Argument Structure (PA). The models on the PA dimension are not trees but directed acyclic graphs (dags), to model re-entrancies e.g. caused by control constructions. Thanks to the modularity of XDG, the PA part of the grammar is the same for German and Dutch.

The set Lab_{PA} of labels on the PA dimension is:

$$Lab_{\mathrm{PA}} = \{\mathsf{ag}, \mathsf{pat}, \mathsf{prop}\} \quad (14)$$

Corresponding resp. to agent, patient and proposition.

The PA dimension uses the following one-dimensional principles:

$$\begin{array}{l} \textit{dag}(\text{PA}) \\ \textit{valency}(\text{PA}, \mathsf{in}_{\text{PA}}, \mathsf{out}_{\text{PA}}) \end{array} \quad (15)$$

Note that we re-use the valency principle again, as we did on the ID and LP dimensions.

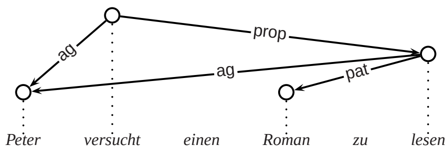
And also the following multi-dimensional principles:

$$\begin{array}{l} \textit{climbing}(\text{ID}, \text{PA}) \\ \textit{linking}(\text{PA}, \text{ID}) \end{array} \quad (16)$$

Here, we re-use the climbing and linking principles. That is, we state that the ID tree is flatter than the corresponding PA dag. This captures raising and control, where arguments of embedded infinite verbs can "climb up" and become arguments of a raising or control verb, in the same way as syntactic arguments can "climb up" from ID to LP. We use the linking principle to specify how semantic arguments are to be realized syntactically (e.g. the agent as a subject etc.).

We display the PA part of the lexicon in Figure 3.

Here is an example PA dag analysis of example sentence (7):



(17)

Here, *Peter* is the agent of *versucht*, and also the agent of *lesen*. Furthermore, *lesen* is a proposition dependent of *versucht*, and *Roman* is the patient of *lesen*.

Notice that the PA dag is indeed a dag and not a tree since *Peter* has two incoming edges: It is simultaneously the agent of *versucht* and of *lesen*. This is enforced by by the valency principle: Both *versucht* and *lesen* require an agent. *Peter* is the only word which can be the agent of both, because it is a subject and the agents of *versucht* and *lesen* must be subjects by the linking principle.[6] The climbing principle ensures that predicate arguments can be "raised" on the ID structure with respect to the PA structure. Again, this example demonstrates that XDG is able to reduce a complex phenomenon such as control to the interaction of per se fairly simple principles such as valency, climbing and linking.

[6] Note that we would have to extend the linking principle in order to account e.g. for object raising.

4.10 Scope structure

(Debusmann et al., 2004a) present a syntax-semantics interface for XDG which additionally introduces a dimension to model quantifier scope. For lack of space, we omit the discussion of it in this paper, but we mention it here to emphasize the extensibility of the framework.

5 Comparison

This section includes a more in-depth comparison of XDG with purely multi- and mono-stratal approaches.

Contrary to multi-stratal approaches like LFG or MTT, XDG is more integrated. For one, it places a lighter burden the interfaces between the dimensions. In LFG for instance, the ϕ-mapping from c-structure to f-structure is rather specific, and has to be specifically adapted to new c-structures, e.g. in order to handle a new construction with a different word order. That is, not only the grammar rules for the c-structure need to be adapted, but also the interface between c- and f-structure. As already stressed several times, in XDG, complex phenomena arise out of the interaction of simple, maximally general principles. Hence to accommodate the new construction, the grammar would ideally only need to be adapted on the word order dimension, leaving the principles in place.

Furthermore, XDG allows interactions of relational constraints between all dimensions, not only between adjacent ones (like c- and f-structure), and in all directions. For one, this gets us bidirectionality (parsing and generation with the same grammar) for free. Secondly, the interactions of XDG have the potential to help greatly in reducing ambiguity. In multi-stratal approaches, ambiguity must be duplicated throughout the system. E.g. suppose there are two candidate c-structures in LFG parsing, but one is ill-formed semantically. Then they can only be ruled out after duplicating the ambiguity on the f-structure, and then filtering out the ill-formed structure on the semantic σ-structure. In XDG on the other hand, the semantic principles can rule out the ill-formed analysis much earlier, typically on the basis of a partial syntactic analysis. Thus, ill-formed analyses are never duplicated, in fact, they are not even produced.

Contrary to mono-stratal ones, XDG is more modular. For one, as (Oliva et al., 1999) note, mono-stratal approaches like HPSG usually give precedence to the syntactic tree structure, while putting the description of other aspects of the analysis on the secondary level only, by means of features spread over the nodes of the tree. As a result,

	$\mathsf{in}_{\mathrm{PA}}$	$\mathsf{out}_{\mathrm{PA}}$	$\mathsf{link}_{\mathrm{PA,ID}}$
den	$\{\}$	$\{\}$	$\{\}$
Roman	$\{\mathsf{ag?}, \mathsf{pat?}\}$	$\{\}$	$\{\}$
Peter	$\{\mathsf{ag?}, \mathsf{pat?}\}$	$\{\}$	$\{\}$
versucht	$\{\}$	$\{\mathsf{ag!}, \mathsf{prop!}\}$	$\{\mathsf{ag} \mapsto \{\mathsf{subj}\}, \mathsf{prop} \mapsto \{\mathsf{vinf}\}\}$
zu	$\{\}$	$\{\}$	$\{\}$
lesen	$\{\mathsf{prop?}\}$	$\{\mathsf{ag!}, \mathsf{pat!}\}$	$\{\mathsf{ag} \mapsto \{\mathsf{subj}\}, \mathsf{pat} \mapsto \{\mathsf{obj}\}\}$

Figure 3: Lexicon of the example grammar fragment, PA part

it becomes a hard task to modularize grammars, e.g. into parts for syntax and semantics. Because syntax is privileged, the phenomena ascribing to semantics cannot be described independently, and whenever the syntax part of the grammar changes, the semantics part needs to be adapted. In XDG, no dimension is in any way privileged to another. Semantic phenomena can be described much more independently from syntax. This facilitates grammar engineering, and also the statement of cross-linguistic generalizations. Assuming that the semantics part of a grammar stay invariant for most natural languages, in order to accommodate a new language, ideally only the syntactic parts would need to be changed, leaving the semantics parts intact. We gave an example of this in §4.

6 Conclusion

In this paper, we introduced the XDG grammar framework, and emphasized that its new methodology places it in between the extremes of multi- and mono-stratal approaches. By means of an idealized example grammar, we demonstrated how complex phenomena can be explained as arising from the interaction of simple principles on numerous dimensions of linguistic description. On the one hand, this methodology has the potential to modularize linguistic description and grammar engineering, and to facilitate the statement of linguistic generalizations. On the other hand, as XDG is a inherently concurrent architecture, inferences from any dimension can help reduce the ambiguity on others. These inferences need not only stem from hard constraints, but can also be preferences to guide the search for solutions.

There are plenty of avenues for future research. Firstly, we plan to continue work on XDG as a framework. Here, one important goal is to find out what criteria we can give to restrict the number of principles. Secondly, we need to evolve the XDG grammar theory, and in particular the XDG syntax-semantics interface (Debusmann et al., 2004a). Thirdly, for practical use, we need to improve our knowledge about XDG solving (i.e. parsing and generation). So far, we could only obtain good results for smaller-scale handwritten grammars, but not for larger-scale grammars induced from treebanks (NEGRA, PDT) or converted from other grammar formalisms (XTAG). Here, we plan to continue research on using XDG to parse and generate with TAG grammars (Koller and Striegnitz, 2002), (Debusmann et al., 2004b). A last goal is to integrate XDG with statistics, e.g. to guide the search for solutions, in the vein of (Dienes et al., 2003).

References

Ralph Debusmann, Denys Duchier, Alexander Koller, Marco Kuhlmann, Gert Smolka, and Stefan Thater. 2004a. A relational syntax-semantics interface based on dependency grammar.

Ralph Debusmann, Denys Duchier, Marco Kuhlmann, and Stefan Thater. 2004b. Tag as dependency grammar. In *Proceedings of TAG+7*, Vancouver/CAN.

Peter Dienes, Alexander Koller, and Marco Kuhlmann. 2003. Statistical A* Dependency Parsing. In *Prospects and Advances in the Syntax/Semantics Interface*, Nancy/FRA.

Denys Duchier and Ralph Debusmann. 2001. Topological dependency trees: A constraint-based account of linear precedence. In *Proceedings of ACL 2001*, Toulouse/FRA.

Denys Duchier. 2003. Configuration of labeled trees under lexicalized constraints and principles. *Research on Language and Computation*, 1(3–4):307–336.

Alexander Koller and Kristina Striegnitz. 2002. Generation as dependency parsing. In *Proceedings of ACL 2002*, Philadelphia/USA.

Mozart Consortium. 2004. The Mozart-Oz website. http://www.mozart-oz.org/.

Karel Oliva, M. Andrew Moshier, and Sabine Lehmann. 1999. Grammar engineering for the next millennium. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium 1999 "Closing the Millennium"*, Beijing/CHI. Tsinghua University Press.