Taking Linguistic Dimensions More Seriously: The New Grammar Formalism of Extensible Dependency Grammar

Ralph Debusmann
Programming Systems Lab
Saarland University
Saarbrücken, Germany
rade@ps.uni-sb.de

Abstract

We introduce the new grammar formalism of Extensible Dependency Grammar (XDG), enabling us to take the different dimensions of linguistic description more seriously than existing grammar formalisms. XDG treats the different linguistic dimensions as separate, autonomous dependency graphs. We show how this allows us to naturally account for taxing phenomena such as free word order, control, and quantifier scope.

1 Introduction

In this paper, we argue that the existing grammar formalisms do not take the different dimensions of linguistic description seriously enough. We propose the new grammar formalism of Extensible Dependency Grammar (XDG), which deals with the different linguistic dimensions as different, autonomous dependency graphs.

The autonomous linguistic dimensions of XDG give rise to numerous advantages. The first is *modularity*: Changes in the grammar pertaining only to one linguistic dimension do not require changes to other parts of the grammar. Another is *perspicuity*: Grammars can be understood and extended more easily since the number of interactions between the linguistic dimensions is kept to a minimum. The third advantage is *concurrency*: The autonomous linguistic dimensions can be processed in parallel, without any implied directionality.

XDG is based on dependency grammar (Tesnière, 1959), (Kunze, 1975),(Mel'čuk, 1988), which is a convenient choice for a number of reasons. Firstly, dependency trees are intuitive, graphic and perspicuous, in contrast to e.g. the more machine-oriented feature structures of HPSG (Pollard and Sag, 1994). Secondly, using dependency grammar, we can easily decouple the linguistic dimensions of syntactic function and word order, whereas phrase structure grammar conflates the two. Thirdly, dependency grammar is convenient for processing: Contrary to phrase structure grammar, we can assume a finite set of nodes per analysis. We call this assumption *finite size assumption*, and exploit it in our efficient constraint-based parser for XDG.

This paper is structured as follows. In section 2, we introduce the essentials of XDG. Thereafter, in section 3, we demonstrate an XDG analysis encompassing five linguistic dimensions, all of which are treated as different, autonomous structures. We start from the linguistic dimension of word order, and go deeper towards the semantics from then on: to the dimension of syntactic function, to deep syntax, predicate argument structure and scope. We describe the multi-dimensional principles relating the linguistic dimensions to each other in section 4. In section 5, we present the XDG parser. Section 6 compares XDG with other approaches before section 7 rounds up the paper.

2 Extensible Dependency Grammar

We start with the essentials of XDG. An XDG *instance* describes an arbitrary number of linguistic *dimensions*. Each dimension is a dependency graph, i.e. a directed graph with labeled edges. All dimensions share the same set of nodes, but have different edges.

An XDG *analysis* is a tuple of graphs, one for each dimension of the XDG instance. Also, an XDG analysis assigns to each node a finite feature structure.

Given a set of input constraints (e.g. a 1:1-mapping of nodes to words in the input), an XDG analysis is *well-formed* with respect to an XDG *grammar* consisting of set of *principles* and a *lexicon*. The lexicon consists of finite feature structures, and is used by the principles to filter out the ill-formed analyzes.

We distinguish *one-dimensional* principles, stipulating constraints on one dimension only, from *multi-dimensional* principles. The latter stipulate constraints which must hold between two or more dimensions.

The XDG lexicon can be built up conveniently using lexical inheritance, and lexical disjunction as in the sense of meta-grammar *crossings* (Candito, 1996).

3 Linguistic Dimensions

In the following, we go through a five-dimensional XDG analysis of the German sentence glossed in (1) below:[1]

Ein gutes Buch versucht jeder Forscher zu lesen.
A good book$_{acc}$ *tries every researcher*$_{nom}$ *to read.*
Every researcher tries to read a good book. (1)

On each dimension, we explain the corresponding XDG analysis, and the one-dimensional principles which state the well-formedness conditions on that dimension. We start with the word order dimension.

3.1 Word Order

On the word order dimension, we make use of the traditional theory of *topological fields* (Höhle, 1986). The theory has also been successfully utilized for HPSG in (Kathol, 1995) and (Müller, 1999). It splits up German sentences into five contiguous parts called (like the theory) *topological fields*:

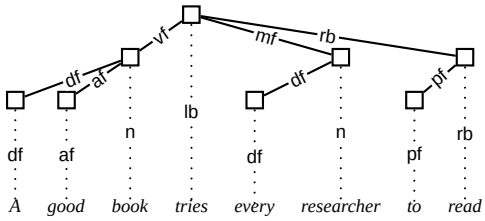
Vorfeld	(	Mittelfeld	)	Nachfeld

(2)

The two round brackets embracing the *Mittelfeld* (middle field) are called left and right sentence brackets. The Mittelfeld contains any number of dependents. In declarative sentences, the left sentence bracket is occupied by the finite verb, and the right sentence bracket by its verbal dependents. The *Vorfeld* (pre-field) contains one topicalized dependent, and the *Nachfeld* (post-field) typically contains subordinate clauses. In our example, the NP *a good book* is in the *Vorfeld*, the finite verb *tries* in the left sentence bracket, the NP *every researcher* in the Mittelfeld and the verbal dependent *read* in the right sentence bracket. The Nachfeld is empty.

3.1.1 Analysis

We encode this topological structure in the following dependency tree:



(3)

We connect words and nodes by dotted edges. Nodes are connected by solid labeled edges. The dotted edges are labeled by *node labels*, and the solid edges by *edge labels*. The label vf corresponds to the Vorfeld, lb to the left sentence bracket, mf to the Mittelfeld, and rb to the right sentence bracket. For the two NPs, we use the labels df for *determiner field*, af for *adjective field*, and n for *noun field*.

3.1.2 Principles

We turn to the question what are the principles which determine the well-formedness conditions for the word order dimension. We use the following XDG principles:

[1] We will use the English counterparts to the German words from here on.

Tree principle. We require that the dependency graphs on the word order dimension are trees.

Valency principle. We require that each each node on the word order dimension must satisfy its lexical *in specification* and its lexical *out specification*. The in specification stipulates what *incoming edges* are licensed, and the out specification what *outgoing edges* are licensed.

As an example, we display the in and out specifications for the finite verb *tries* below:

$$\textit{tries} \quad \begin{bmatrix} \mathsf{in} & : & \{\} \\ \mathsf{out} & : & \{\mathsf{vf}?, \mathsf{mf}*, \mathsf{rb}?, \mathsf{nf}?\} \end{bmatrix} \tag{4}$$

They are read as follows. By its in specification, *tries* does not license any incoming edge. By its out specification, it licenses zero or one outgoing edges into the Vorfeld ($\mathsf{vf}?$), zero or more outgoing edges into the Mittelfeld ($\mathsf{mf}*$), zero or one outgoing edges into the right sentence bracket ($\mathsf{rb}?$), and zero or one outgoing edges into the Nachfeld ($\mathsf{nf}?$).

As a second example, consider the in and out specifications of the noun *book*:

$$\textit{book} \quad \begin{bmatrix} \mathsf{in} & : & \{\mathsf{vf}?, \mathsf{mf}?\} \\ \mathsf{out} & : & \{\mathsf{df}?, \mathsf{af}*\} \end{bmatrix} \tag{5}$$

book can be placed in the Vorfeld or in the Mittelfeld ($\mathsf{vf}?$ and $\mathsf{mf}?$ in the in specification), and licenses zero or one outgoing determiner field edges, and zero or more outgoing adjective field edges ($\mathsf{df}?$ and $\mathsf{af}*$ in the out specification).

Projectivity principle. We require that the dependency graphs on the word order dimension are projective.

Order principle. We require that the dependency graphs on the word order dimension satisfy the order principle.

We assign to each node a node label, and stipulate a total order $\prec$ on the set of node and edge labels:

$$\begin{array}{l} \mathsf{df} \prec \mathsf{af} \prec \mathsf{n} \prec \\ \mathsf{vf} \prec \mathsf{lb} \prec \mathsf{mf} \prec \mathsf{rb} \prec \mathsf{nf} \end{array} \tag{6}$$

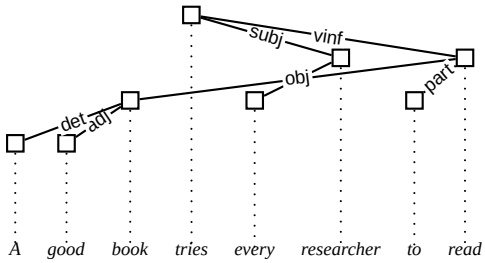
The order principle is only satisfied if for each node in the dependency graph, the order of its daughters is compatible with $\prec$. For instance all daughters with edge label vf must precede all daughters with edge label mf. In addition, the order principle requires that the mother is positioned with respect to its daughters by its node label. In our example, the finite verb *tries* has node label lb, i.e. it must be positioned between the daughters with edge label vf and the daughters with edge label mf.

3.2 Syntactic Function

On the linguistic dimension of syntactic function, we make use of traditional dependency grammar (Tesnière, 1959), (Kunze, 1975), (Mel'čuk, 1988).

3.2.1 Analysis

We display the XDG analysis on the syntactic function dimension below:



(7)

Here, *tries* is the root. It has two dependents, the subject *researcher* (edge label subj) and the infinitive *read* (vinf). *researcher* has one dependent: the determiner *every* (det). *read* has two dependents: the particle *to* (part) and the object *book* (obj). *book* also has two dependents, the determiner *a* and the adjective *good* (adj).

3.2.2 Principles

Tree principle.

Valency principle. As an example, we show the in and out specifications for *tries*:

$$\textit{tries} \quad \begin{bmatrix} \mathsf{in} & : & \{\} \\ \mathsf{out} & : & \{\mathsf{subj}!, \mathsf{vinf}!\} \end{bmatrix} \tag{8}$$

tries does not license any incoming edge (in specification). It requires precisely one outgoing subject edge and one outgoing infinitival complement edge (subj! and vinf! in the out specification).

As another example, we show the in and out specifications for *read*:

$$\textit{read} \quad \begin{bmatrix} \mathsf{in} & : & \{\mathsf{vinf}?\} \\ \mathsf{out} & : & \{\mathsf{obj}!, \mathsf{part}!\} \end{bmatrix} \tag{9}$$

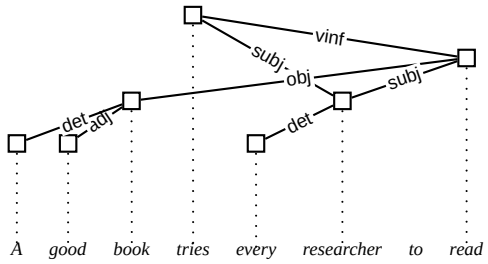
read can be an infinitival complement (vinf? in the in specification), and requires an object and a particle (obj! and part! in the out specification).

3.3 Deep syntax

We introduce the linguistic dimension of *deep syntax*, which is very similar to the dimension of syntactic function. However, there are two main differences. 1) Words without semantic content like *to*-particles and expletives are not connected, 2) The deep syntax includes edges for *deep subjects* of embedded non-finite verbs.

3.3.1 Analysis

Here is the XDG analysis on the deep syntax dimension:



(10)

Compared with the syntactic function analysis in (7), there are two differences. 1) The edge from *read* to *to* is removed because the *to*-particle has no semantic content, and 2) There is an additional deep subject edge from *read* to *researcher*. Hence, *researcher* has two incoming edges in the deep syntax: it is simultaneously the deep subject of *tries* and the deep subject of *read*. This means that the dependency graphs on the deep syntax dimension cannot be trees.

3.3.2 Principles

Dag principle. We require that the dependency graphs on the deep syntax dimension are directed acyclic graphs.

Valency principle. Here are the in and out specifications for *read*:

$$\textit{read} \quad \begin{bmatrix} \mathsf{in} & : & \{\mathsf{vinf?}\} \\ \mathsf{out} & : & \{\mathsf{obj!}, \mathsf{subj!}\} \end{bmatrix} \tag{11}$$

read can be an infinitival complement (vinf? in the in specification), and requires a deep object and a deep subject (obj! and subj! in the out specification). Contrary to its out specification on the syntactic function dimension, *read* does not license any particle edge, but does require a subject edge.

3.4 Predicate Argument Structure

On the linguistic dimension of predicate argument structure, our goal is to construct a representation similar to the flat semantic representations developed for Machine Translation in (Phillips, 1993) and (Trujillo, 1995), and also used as the basis for Minimal Recursion Semantics (MRS) (Copestake et al., 1999).

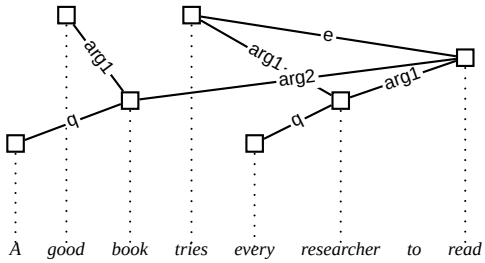
Our predicate argument structure for the example sentence is a multiset of semantic literals such as the following:

$$\begin{array}{l} \{every(x), researcher(x), \\ a(y), good(y), book(y), \\ try(e, x, e'), read(e', x, y)\} \end{array} \tag{12}$$

Here, x and y are individual variables which are implicitly universally bound. e and e' are event variables in the sense of (Davidson, 1967).

3.4.1 Analysis

We encode this predicate argument structure in the following dependency graph:



(13)

The dependency graph contains an edge labeled with q from each noun to its quantifier. The edge from *tries* to *read* labeled e represents that the event corresponding to *tries* embeds the event corresponding to *read*. The other edges fill the remaining argument positions of the predicates.

We use theory-neutral labels such as arg1 and arg2 for the argument positions, rather than thematic roles. Although thematic roles can be useful to describe predicate argument structure, there are too many arguments against using them (Dowty, 1989).

3.4.2 Principles

Dag principle.

Valency principle. Below, we show the in and out specifications of *read*:

$$\textit{read} \quad \begin{bmatrix} \mathsf{in} & : & \{\mathsf{e?}\} \\ \mathsf{out} & : & \{\mathsf{arg1!}, \mathsf{arg2!}\} \end{bmatrix} \tag{14}$$

read can be an embedded event (e? in the in specification), and has two arguments (arg1! and arg2! in the out specification).

3.5 Scope

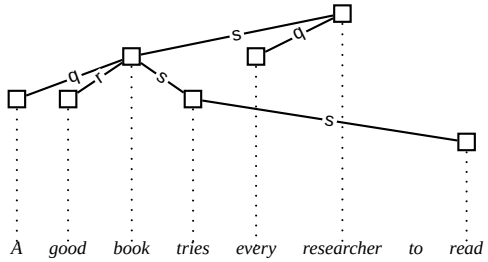
On the scope dimension, we proceed in a similar fashion as MRS: The aim of the scope dimension is to add the information needed to obtain only the admissible scope readings of a sentence. One of the two scope readings of our example is the following:

$$\begin{array}{l} every(x, researcher(x), \\ a(y, good(y) \wedge book(y), \\ try(x, read(x, y)))) \end{array} \tag{15}$$

(15) represents the the weak reading of our example, where the universal quantifier outscopes the existential quantifier. We use a fairly standard notation for generalized quantifiers: The first argument slot corresponds to the bound variable of the quantifier, the second to the restriction, and the third to the body or scope. We abandon the event variables of the flat semantic representation; instead, we properly embed the predicate corresponding to *read* as an argument of the predicate corresponding to *tries*.

3.5.1 Analysis

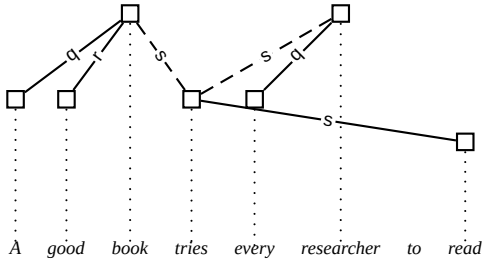
We encode the weak reading in the following dependency tree:



(16)

As in the predicate argument dimension, edges labeled q indicate go from nouns to their quantifiers. Edges labeled r go from quantified nouns to their restriction (e.g. the edge from *book* to *good*), and edges labeled s indicate that the mother outscopes the daughter. Here, *researcher* outscopes *book*, and *book* outscopes *tries*.

Often, it is convenient to underspecify the scopal relationships in a sentence. In our constraint-based setting, we get underspecification for free: We can use our parser to get *partial parses*, and only enumerate the fully specified readings on demand. We depict a partial parse for our example sentence below:



(17)

At this stage, the parser already knows that both quantified nouns *researcher* and *book* outscope *tries*, but it has not yet committed to either scopal relationship of *researcher* and *book*. In the dependency tree, this is represented by two dashed *dominance edges*: One from *researcher* to *tries*, and another one from *book* to *tries*.

3.5.2 Principles

Tree principle.

Valency principle. Below, we show the in and out specifications of *book*:

$$\textit{book} \quad \begin{bmatrix} \mathsf{in} & : & \{\mathsf{s?}\} \\ \mathsf{out} & : & \{\mathsf{q!}, \mathsf{r}*, \mathsf{s!}\} \end{bmatrix} \tag{18}$$

book can be in the scope of another node (s? in the in specification), and requires a quantifier, can have any number of elements in its restriction, and takes scope (q!, r∗ and s! in the out specification).

4 Multi-dimensional principles

The principles used so far were *one-dimensional*, i.e. they pertained only to one dimension at a time. Although we emphasize the importance of making the linguistic dimensions as autonomous as possible, we must be able to express interrelations between them. In XDG, we call principles which relate more than one dimension *multi-dimensional*.

Climbing principle. The climbing principle states that the dependency graph in one dimension must be flatter than the corresponding dependency graph in another dimension.

The climbing principle must hold between the word order and the syntactic function dimension: Each dependency tree on the word order dimension must be flatter than the corresponding dependency tree on the syntactic function dimension.

In fact, our example word order tree in (3) is flatter than the corresponding syntactic function tree in (7) to allow for topicalization of the NP *a good book*.

Linking principle. The lexicalized linking principle stipulates that in order for an edge in one dimension to be licensed, the daughter must have a certain incoming edge label in the other dimension.

As an example, we show how we use the linking principle to describe how the semantic arguments of *read* are realized in the deep syntax:

$$\textit{read} \quad \left[\, \mathsf{link} \;:\; \left[\, \begin{array}{lcl} \mathsf{arg1} & : & \{\mathsf{subj}\} \\ \mathsf{arg2} & : & \{\mathsf{obj}\} \end{array} \,\right] \,\right] \tag{19}$$

The link specification states that the first argument (arg1) of *read* is realized by the deep subject (subj), and the second argument (arg2) by the deep object (obj). By introducing the deep syntax definition, we do not have to worry about control (or raising) when we establish this linking.

Co-immediate dominance principle. The lexicalized co-immediate dominance principle stipulates that in order for an edge in one dimension to be licensed, there must a corresponding edge in the other dimension.

Contra-immediate dominance principle. The lexicalized contra-immediate dominance principle stipulates that in order for an edge in one dimension to be licensed, there must a corresponding edge in the other dimension, where mother and daughter are reversed.

As an example, we show the specifications for co- and contra- immediate dominance of the noun *book*:

$$\textit{book} \quad \left[\, \begin{array}{rcl} \mathsf{coimm} & : & \{\mathsf{q}\} \\ \mathsf{contraimm} & : & \{\mathsf{r}\} \end{array} \,\right] \tag{20}$$

Here, we use the co-immediate dominance principle (coimm feature) to state that in order for an each edge labeled q on the scope dimension to be licensed, there must be a corresponding edge on the predicate argument dimension. We use the contra-immediate dominance principle (contraimm feature) to state that in order for an edge labeled r on the scope dimension to be licensed, there must be a corresponding edge on the predicate argument dimension where mother and daughter are reversed.

Co-dominance principle. The lexicalized co-dominance principle stipulates that in order for an edge in one dimension to be licensed, the mother must dominate the daughter in the other dimension.

Contra-dominance principle. The lexicalized contra-dominance principle stipulates that in order for an edge in one dimension to be licensed, the daughter must dominate the mother in the other dimension.

As an example, we show the specifications for co- and contra- dominance of the verb *tries*:

$$\textit{tries} \quad \left[\, \begin{array}{rcl} \mathsf{codom} & : & \{\mathsf{e}\} \\ \mathsf{contradom} & : & \{\mathsf{arg1}\} \end{array} \,\right] \tag{21}$$

We use the co-dominance principle (codom feature) to state that each edge labeled e from a verb to an embedded verb on the predicate argument dimension implies that the verb takes scope over the embedded verb. We use the contra-dominance principle (contradom feature) to state that each edge labeled arg1 from a verb to a noun on the predicate argument dimension implies that the noun takes scope over the verb.

5 Parsing

XDG can already be used for practical experiments: We have written a constraint-based parser system for XDG based on *constraint propagation* (Maruyama, 1990), which is a straightforward translation of the axiomatization of XDG into a constraint satisfaction problem on finite sets. We make use of the *selection constraint* (Duchier, 1999) to efficiently prune the search space with respect to lexical ambiguity. The XDG parser is *concurrent*, i.e. all linguistic dimensions are parsed in

parallel. Hence, the XDG parser has no fixed directionality, and can also be used for generation. Information can flow from any dimension to any other: Not only can information from the syntactic influence the semantic dimensions, but also the other way round.

The worst-case complexity of XDG parsing is NP-complete. This has been proven for a two-dimensional instance of XDG (Koller and Striegnitz, 2002), but not for the general case. Still, we can already say that XDG parsing is decidable because of the *finite size assumption*: all the parameters of an XDG parse are finite.

Average-case complexity of XDG parsing depends on the grammar, but experiments suggest that it is always polynomial in practice. For a small handcrafted grammar of German, we observed polynomial parse times (e.g. 2 seconds for a 50 word sentence) with an exponent of 2.5. Parsing large grammars (induced from treebanks) is also polynomial, although it is far slower than state-of-the-art one-dimensional dependency parsers such as the one described in (Carroll and Briscoe, 2002).

6 Comparison to Other Approaches

XDG is similar to Meaning Text Theory (MTT) (Mel'čuk, 1988) with its seven *strata* of representation. But whereas the the mappings from one strata to another have not yet been formally specified, all principles of XDG are completely axiomatized.

Lexical-Functional Grammar (LFG) (Bresnan and Kaplan, 1982) is another similar grammar formalism. Both LFG and XDG are essentially dependency-based (although LFG has a context-free backbone by its c-structure). However, there are two main differences. Firstly, LFG uses the *functional* ϕ-mapping to get from its c-structure to the f-structure, where XDG uses *relational* principles to interrelate its dimensions. Thus, whereas LFG is inherently directional, XDG is not. Secondly, LFG uses feature structures as the representational device on the f-structure, whereas XDG uniformly uses arguably more perspicuous dependency graphs on all of its dimensions.

XDG is also similar to HPSG (Pollard and Sag, 1994). Both approaches are constraint-based, and essentially dependency-based, as the *headed structures* of HPSG encode nothing else but dependencies. The three main differences are the following. Firstly, HPSG does not recognize the autonomy of linguistic dimensions. Although HPSG's signs include different feature structures for different linguistic dimensions, they are too strongly tied to the syntax. Almost each change in the syntax requires changes in other parts of the grammar pertaining e.g. to semantics, and vice versa. Secondly, the machine-oriented feature structures of HPSG are not at all perspicuous. Thirdly, HPSG is not completely formally specified.

7 Conclusion

In this paper, we introduced the new grammar formalism of Extensible Dependency Grammar (XDG), which allows us to fulfill our proposal of taking the different linguistic dimensions more seriously. XDG allows to treat the different linguistic dimensions as autonomous structures, represented in a uniform way as dependency graphs. We utilize the concept of valency on all linguistic dimensions to give them a higher degree of autonomy. The different linguistic dimensions are only interrelated as loosely as possible by multi-dimensional principles.

For purposes of illustration, we explained a five-dimensional XDG analysis of a German sentence exhibiting interesting phenomena on the different linguistic dimensions: Topicalization on the word order dimension, control on the deep syntax, and scope ambiguity on the scope dimension. We demonstrated that by separating the participating linguistic dimensions, we can cover these phenomena in a natural way.

XDG has already spawned numerous related work. (Koller and Striegnitz, 2002) used the XDG parser for generation with Tree Adjoining Grammars and even outran the generator described in (Carroll et al., 1999). (Kuhlmann, 2002) uses a version of XDG parser to parse Categorial Type Logic (CTL) grammars. (Duchier and Kruijff, 2003) introduce Information Structure to XDG. (Dienes et al., 2003) present SXDG, a version of XDG equipped with a statistical component.

For the near future, our goal is to improve

XDG and the accompanying linguistic theory. Another research goal is to obtain large grammars for XDG, by grammar induction (NEGRA, PDT) and by converting from existing grammar resources (e.g. the XTAG grammar). Especially for parsing with large grammars, we need to further improve the XDG parser. One the one hand, we want to improve the efficiency of the pure XDG parser. On the other, we want to continue developing SXDG, the version of XDG enhanced with a statistical component.

References

Joan Bresnan and Ronald Kaplan. 1982. Lexical-functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. The MIT Press, Cambridge/USA.

Marie-H`eléne Candito. 1996. A principle-based hierarchical representation of LTAG. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING 1996)*, Kopenhagen/DEN.

John Carroll and Ted Briscoe. 2002. High precision extraction of grammatical relations. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*, Taipei/TW.

John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznanski. 1999. An effi cient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on NLG*, pages 86–95, Toulouse/FRA.

Ann Copestake, Dan Flickinger, and Ivan Sag. 1999. Minimal recursion semantics. an introduction. Manuscript.

Donald Davidson. 1967. Truth and meaning. *Synthese 17*, pages 304–323.

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

David R. Dowty. 1989. On the semantic content of the notion of "thematic role". In Gennaro Chierchia, Barbara H. Partee, and Ray Turner, editors, *Properties, Types and Meanings*, volume 2, pages 69–129. Kluwer, Dordrecht/NL.

Denys Duchier and Geert-Jan M. Kruijff. 2003. Information structure in topological dependency grammar. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003)*.

Denys Duchier. 1999. Axiomatizing dependency parsing using set constraints. In *6th Meeting on the Mathematics of Language (MOL 6)*, Orlando/USA.

Tilman H¨ohle. 1986. Der Begriff "Mittelfeld", Anmerkungen ¨uber die Theorie der topologischen Felder. In Walter Weiss, Herbert Ernst Wiegand, and Marga Reis, editors, *Akten des 7. Internationalen Germanisten-Kongresses*, pages 329–340, T¨ubingen/GER. Max Niemeyer Verlag.

Andreas Kathol. 1995. *Linearization-Based German Syntax*. Ph.D. thesis, Ohio State University, Ohio/USA.

Alexander Koller and Kristina Striegnitz. 2002. Generation as dependency parsing. In *Proceedings of the 40th Anniversary Meeting of the ACL (ACL 2002)*, Philadelphia/USA.

Marco Kuhlmann. 2002. Towards a constraint parser for categorial type logics. Master's thesis, Division of Informatics, University of Edinburgh, Edinburgh/UK.

J¨urgen Kunze. 1975. *Abhängigkeitsgrammatik*. Akademie Verlag, Berlin/GER.

H. Maruyama. 1990. Structural disambiguation with constraint propagation. In *The Proceedings of the 28th Annual Meeting of the ACL (ACL 1990)*, pages 31–38, Pittsburgh/USA.

Igor Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State Univ. Press of New York, Albany/USA.

Stefan M¨uller. 1999. *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*. Linguistische Arbeiten 394. Max Niemeyer Verlag, T¨ubingen/GER.

John .D. Phillips. 1993. Generation of text from logical formulae. *Machine Translation*, 8(4):209–235.

Carl Pollard and Ivan Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago/USA.

Lucien Tesni`ere. 1959. *Éléments de Syntaxe Structurale*. Klincksiek, Paris/FRA.

Indalecio Arturo Trujillo. 1995. *Lexicalist Machine Translation of Spatial Prepositions*. Ph.D. thesis, University of Cambridge, Cambridge/USA.