Mildly Non-Projective Dependency Structures

Marco Kuhlmann
Programming Systems Lab
Saarland University
Germany
kuhlmann@ps.uni-sb.de

Joakim Nivre
Växjö University and
Uppsala University
Sweden
nivre@msi.vxu.se

Abstract

Syntactic parsing requires a fine balance between expressivity and complexity, so that naturally occurring structures can be accurately parsed without compromising efficiency. In dependency-based parsing, several constraints have been proposed that restrict the class of permissible structures, such as projectivity, planarity, multi-planarity, well-nestedness, gap degree, and edge degree. While projectivity is generally taken to be too restrictive for natural language syntax, it is not clear which of the other proposals strikes the best balance between expressivity and complexity. In this paper, we review and compare the different constraints theoretically, and provide an experimental evaluation using data from two treebanks, investigating how large a proportion of the structures found in the treebanks are permitted under different constraints. The results indicate that a combination of the well-nestedness constraint and a parametric constraint on discontinuity gives a very good fit with the linguistic data.

1 Introduction

Dependency-based representations have become increasingly popular in syntactic parsing, especially for languages that exhibit free or flexible word order, such as Czech (Collins et al., 1999), Bulgarian (Marinov and Nivre, 2005), and Turkish (Eryiğit and Oflazer, 2006). Many practical implementations of dependency parsing are restricted to *projective* structures, where the projection of a head word has to form a continuous substring of the sentence. While this constraint guarantees good parsing complexity, it is well-known that certain syntactic constructions can only be adequately represented by *non-projective* dependency structures, where the projection of a head can be discontinuous. This is especially relevant for languages with free or flexible word order.

However, recent results in non-projective dependency parsing, especially using data-driven methods, indicate that most non-projective structures required for the analysis of natural language are very nearly projective, differing only minimally from the best projective approximation (Nivre and Nilsson, 2005; Hall and Novák, 2005; McDonald and Pereira, 2006). This raises the question of whether it is possible to characterize a class of *mildly* non-projective dependency structures that is rich enough to account for naturally occurring syntactic constructions, yet restricted enough to enable efficient parsing.

In this paper, we review a number of proposals for classes of dependency structures that lie between strictly projective and completely unrestricted non-projective structures. These classes have in common that they can be characterized in terms of properties of the dependency structures themselves, rather than in terms of grammar formalisms that generate the structures. We compare the proposals from a theoretical point of view, and evaluate a subset of them empirically by testing their representational adequacy with respect to two dependency treebanks: the Prague Dependency Treebank (PDT) (Hajič et al., 2001), and the Danish Dependency Treebank (DDT) (Kromann, 2003).

The rest of the paper is structured as follows. In section 2, we provide a formal definition of dependency structures as a special kind of directed graphs, and characterize the notion of projectivity. In section 3, we define and compare five different constraints on mildly non-projective dependency structures that can be found in the literature: planarity, multiplanarity, well-nestedness, gap degree, and edge degree. In section 4, we provide an experimental evaluation of the notions of planarity, well-nestedness, gap degree, and edge degree, by

investigating how large a proportion of the dependency structures found in PDT and DDT are allowed under the different constraints. In section 5, we present our conclusions and suggestions for further research.

2 Dependency graphs

For the purposes of this paper, a *dependency graph* is a directed graph on the set of indices corresponding to the tokens of a sentence. We write $[n]$ to refer to the set of positive integers up to and including n.

Definition 1 A *dependency graph* for a sentence $x = w_1, \ldots, w_n$ is a directed graph[1]

$$G = (V\,;\,E), \quad \text{where } V = [n] \text{ and } E \subseteq V \times V.$$

Throughout this paper, we use standard terminology and notation from graph theory to talk about dependency graphs. In particular, we refer to the elements of the set V as *nodes*, and to the elements of the set E as *edges*. We write $i \to j$ to mean that there is an edge from the node i to the node j (i.e., $(i, j) \in E$), and $i \to^* j$ to mean that the node i *dominates* the node j, i.e., that there is a (possibly empty) path from i to j. For a given node i, the set of nodes dominated by i is the *yield* of i. We use the notation $\pi(i)$ to refer to the *projection* of i: the yield of i, arranged in ascending order.

2.1 Dependency forests

Most of the literature on dependency grammar and dependency parsing does not allow arbitrary dependency graphs, but imposes certain structural constraints on them. In this paper, we restrict ourselves to dependency graphs that form *forests*.

Definition 2 A *dependency forest* is a dependency graph with two additional properties:

1. it is acyclic (i.e., if $i \to j$, then not $j \to^* i$);
2. each of its nodes has at most one incoming edge (i.e., if $i \to j$, then there is no node k such that $k \neq i$ and $k \to j$).

Nodes in a forest without an incoming edge are called *roots*. A dependency forest with exactly one root is a *dependency tree*.

Figure 1 shows a dependency forest taken from PDT. It has two roots: node 2 (corresponding to the complementizer *proto*) and node 8 (corresponding to the final punctuation mark).

[1] We only consider unlabelled dependency graphs.

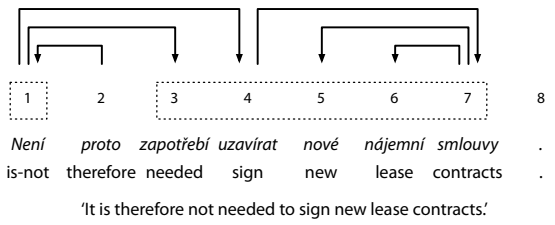


Figure 1: Dependency forest for a Czech sentence from the Prague Dependency Treebank

Some authors extend dependency forests by a special root node with position 0, and add an edge $(0, i)$ for every root node i of the remaining graph (McDonald et al., 2005). This ensures that the extended graph always is a tree. Although such a definition can be useful, we do not follow it here, since it obscures the distinction between projectivity and planarity to be discussed in section 3.

2.2 Projectivity

In contrast to acyclicity and the indegree constraint, both of which impose restrictions on the dependency relation as such, the *projectivity constraint* concerns the interaction between the dependency relation and the positions of the nodes in the sentence: it says that the nodes in a subtree of a dependency graph must form an *interval*, where an interval (with endpoints i and j) is the set

$$[i, j] \;:=\; \{\, k \in V \mid i \leq k \text{ and } k \leq j \,\}.$$

Definition 3 A dependency graph is *projective*, if the yields of its nodes are intervals.

Since projectivity requires each node to dominate a continuous substring of the sentence, it corresponds to a ban on discontinuous constituents in phrase structure representations.

Projectivity is an interesting constraint on dependency structures both from a theoretical and a practical perspective. Dependency grammars that only allow projective structures are closely related to context-free grammars (Gaifman, 1965; Obrębski and Graliński, 2004); among other things, they have the same (weak) expressivity. The projectivity constraint also leads to favourable parsing complexities: chart-based parsing of projective dependency grammars can be done in cubic time (Eisner, 1996); hard-wiring projectivity into a deterministic dependency parser leads to linear-time parsing in the worst case (Nivre, 2003).

3 Relaxations of projectivity

While the restriction to projective analyses has a number of advantages, there is clear evidence that it cannot be maintained for real-world data (Zeman, 2004; Nivre, 2006). For example, the graph in Figure 1 is non-projective: the yield of the node 1 (marked by the dashed rectangles) does not form an interval—the node 2 is 'missing'. In this section, we present several proposals for structural constraints that relax projectivity, and relate them to each other.

3.1 Planarity and multiplanarity

The notion of *planarity* appears in work on Link Grammar (Sleator and Temperley, 1993), where it is traced back to Mel'čuk (1988). Informally, a dependency graph is *planar*, if its edges can be drawn above the sentence without crossing. We emphasize the word *above*, because planarity as it is understood here does not coincide with the standard graph-theoretic concept of the same name, where one would be allowed to also use the area below the sentence to disentangle the edges.

Figure 2a shows a dependency graph that is planar but not projective: while there are no crossing edges, the yield of the node 1 (the set $\{1, 3\}$) does not form an interval.

Using the notation $linked(i, j)$ as an abbreviation for the statement 'there is an edge from i to j, or vice versa', we formalize planarity as follows:

Definition 4 A dependency graph is *planar*, if it does not contain nodes a, b, c, d such that

$$linked(a, c) \wedge linked(b, d) \wedge a < b < c < d\,.$$

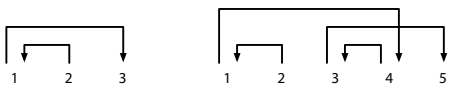
Yli-Jyrä (2003) proposes *multiplanarity* as a generalization of planarity suitable for modelling dependency analyses, and evaluates it experimentally using data from DDT.

Definition 5 A dependency graph $G = (V\,;\,E)$ is *m-planar*, if it can be split into m planar graphs

$$G_1 = (V\,;\,E_1), \ldots, G_m = (V\,;\,E_m)$$

such that $E = E_1 \uplus \cdots \uplus E_m$. The planar graphs G_i are called *planes*.

As an example of a dependency forest that is 2-planar but not planar, consider the graph depicted in Figure 2b. In this graph, the edges $(1, 4)$ and $(3, 5)$ are crossing. Moving either edge to a separate graph partitions the original graph into two planes.



(a) 1-planar (b) 2-planar

Figure 2: Planarity and multi-planarity

3.2 Gap degree and well-nestedness

Bodirsky et al. (2005) present two structural constraints on dependency graphs that characterize analyses corresponding to derivations in Tree Adjoining Grammar: the *gap degree restriction* and the *well-nestedness constraint*.

A *gap* is a discontinuity in the projection of a node in a dependency graph (Plátek et al., 2001). More precisely, let π_i be the projection of the node i. Then a gap is a pair (j_k, j_{k+1}) of nodes adjacent in π_i such that $j_{k+1} - j_k > 1$.

Definition 6 The *gap degree* of a node i in a dependency graph, $gd(i)$, is the number of gaps in π_i.

As an example, consider the node labelled i in the dependency graphs in Figure 3. In Graph 3a, the projection of i is an interval $((2, 3, 4))$, so i has gap degree 0. In Graph 3b, $\pi_i = (2, 3, 6)$ contains a single gap $((3, 6))$, so the gap degree of i is 1. In the rightmost graph, the gap degree of i is 2, since $\pi_i = (2, 4, 6)$ contains two gaps $((2, 4)$ and $(4, 6))$.

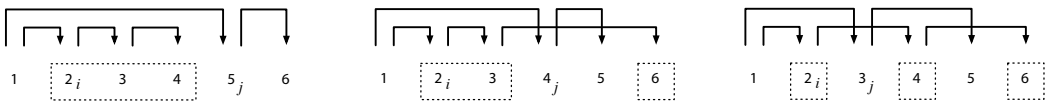
Definition 7 The *gap degree* of a dependency graph G, $gd(G)$, is the maximum among the gap degrees of its nodes.

Thus, the gap degree of the graphs in Figure 3 is 0, 1 and 2, respectively, since the node i has the maximum gap degree in all three cases.

The *well-nestedness constraint* restricts the positioning of disjoint subtrees in a dependency forest. Two subtrees are called disjoint, if neither of their roots dominates the other.

Definition 8 Two subtrees T_1, T_2 *interleave*, if there are nodes $l_1, r_1 \in T_1$ and $l_2, r_2 \in T_2$ such that $l_1 < l_2 < r_1 < r_2$. A dependency graph is *well-nested*, if no two of its disjoint subtrees interleave.

Both Graph 3a and Graph 3b are well-nested. Graph 3c is not well-nested. To see this, let T_1 be the subtree rooted at the node labelled i, and let T_2 be the subtree rooted at j. These subtrees interleave, as T_1 contains the nodes 2 and 4, and T_2 contains the nodes 3 and 5.



(a) $gd = 0, ed = 0, wn+$ (b) $gd = 1, ed = 1, wn+$ (c) $gd = 2, ed = 1, wn-$

Figure 3: Gap degree, edge degree, and well-nestedness

3.3 Edge degree

The notion of *edge degree* was introduced by Nivre (2006) in order to allow mildly non-projective structures while maintaining good parsing efficiency in data-driven dependency parsing.[2]

Define the *span* of an edge (i, j) as the interval

$$S((i, j)) \;:=\; [\min(i, j), \max(i, j)]\,.$$

Definition 9 Let $G = (V ; E)$ be a dependency forest, let $e = (i, j)$ be an edge in E, and let G_e be the subgraph of G that is induced by the nodes contained in the span of e.

- The *degree* of an edge $e \in E$, $ed(e)$, is the number of connected components c in G_e such that the root of c is not dominated by the head of e.
- The *edge degree* of G, $ed(G)$, is the maximum among the degrees of the edges in G.

To illustrate the notion of edge degree, we return to Figure 3. Graph 3a has edge degree 0: the only edge that spans more nodes than its head and its dependent is $(1, 5)$, but the root of the connected component $\{2, 3, 4\}$ is dominated by 1. Both Graph 3b and 3c have edge degree 1: the edge $(3, 6)$ in Graph 3b and the edges $(2, 4)$, $(3, 5)$ and $(4, 6)$ in Graph 3c each span a single connected component that is not dominated by the respective head.

3.4 Related work

Apart from proposals for structural constraints relaxing projectivity, there are dependency frameworks that in principle allow unrestricted graphs, but provide mechanisms to control the actually permitted forms of non-projectivity in the grammar.

The non-projective dependency grammar of Kahane et al. (1998) is based on an operation on dependency trees called *lifting*: a 'lift' of a tree T is the new tree that is obtained when one replaces one or more edges (i, k) in T by edges (j, k), where $j \rightarrow^* i$. The exact conditions under which a certain lifting may take place are specified in the rules of the grammar. A dependency tree is acceptable, if it can be lifted to form a projective graph.[3]

A similar design is pursued in Topological Dependency Grammar (Duchier and Debusmann, 2001), where a dependency analysis consists of two, mutually constraining graphs: the *ID graph* represents information about immediate dominance, the *LP graph* models the topological structure of a sentence. As a principle of the grammar, the LP graph is required to be a lift of the ID graph; this lifting can be constrained in the lexicon.

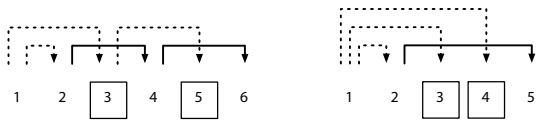
3.5 Discussion

The structural conditions we have presented here naturally fall into two groups: multiplanarity, gap degree and edge degree are *parametric constraints* with an infinite scale of possible values; planarity and well-nestedness come as *binary constraints*. We discuss these two groups in turn.

Parametric constraints With respect to the graded constraints, we find that multiplanarity is different from both gap degree and edge degree in that it involves a notion of optimization: since every dependency graph is m-planar for some sufficiently large m (put each edge onto a separate plane), the interesting question in the context of multiplanarity is about the *minimal* values for m that occur in real-world data. But then, one not only needs to show that a dependency graph *can* be decomposed into m planar graphs, but also that this decomposition is the one with the smallest number of planes among all possible decompositions. Up to now, no tractable algorithm to find the minimal decomposition has been given, so it is not clear how to evaluate the significance of the concept as such. The evaluation presented by Yli-Jyrä (2003) makes use of additional constraints that are sufficient to make the decomposition unique.

[2] We use the term *edge degree* instead of the original simple term *degree* from Nivre (2006) to mark the distinction from the notion of gap degree.

[3] We remark that, without restrictions on the lifting, *every* non-projective tree has a projective lift.



(a) $gd = 2$, $ed = 1$ (b) $gd = 1$, $ed = 2$

Figure 4: Comparing gap degree and edge degree

The fundamental difference between gap degree and edge degree is that the gap degree measures the number of discontinuities within a subtree, while the edge degree measures the number of intervening constituents spanned by a single edge. This difference is illustrated by the graphs displayed in Figure 4. Graph 4a has gap degree 2 but edge degree 1: the subtree rooted at node 2 (marked by the solid edges) has two gaps, but each of its edges only spans one connected component not dominated by 2 (marked by the squares). In contrast, Graph 4b has gap degree 1 but edge degree 2: the subtree rooted at node 2 has one gap, but this gap contains two components not dominated by 2.

Nivre (2006) shows experimentally that limiting the permissible edge degree to 1 or 2 can reduce the average parsing time for a deterministic algorithm from quadratic to linear, while omitting less than 1% of the structures found in DDT and PDT. It can be expected that constraints on the gap degree would have very similar effects.

Binary constraints For the two binary constraints, we find that well-nestedness subsumes planarity: a graph that contains interleaving subtrees cannot be drawn without crossing edges, so every planar graph must also be well-nested. To see that the converse does not hold, consider Graph 3b, which is well-nested, but not planar.

Since both planarity and well-nestedness are proper extensions of projectivity, we get the following hierarchy for sets of dependency graphs:

$$\text{projective} \subset \text{planar} \subset \text{well-nested} \subset \text{unrestricted}$$

The planarity constraint appears like a very natural one at first sight, as it expresses the intuition that 'crossing edges are bad', but still allows a limited form of non-projectivity. However, many authors use planarity in conjunction with a special representation of the root node: either as an artificial node at the sentence boundary, as we mentioned in section 2, or as the target of an infinitely long perpendicular edge coming 'from the outside', as in earlier versions of Word Grammar (Hudson, 2003). In these situations, planarity reduces to projectivity, so nothing is gained.

Even in cases where planarity is used without a special representation of the root node, it remains a peculiar concept. When we compare it with the notion of gaps, for example, we find that, in a planar dependency tree, every gap (i, j) must contain the root node r, in the sense that $i < r < j$: if the gap would only contain non-root nodes k, then the two paths from r to k and from i to j would cross. This particular property does not seem to be mirrored in any linguistic prediction.

In contrast to planarity, well-nestedness is independent from both gap degree and edge degree in the sense that for every $d > 0$, there are both well-nested and non-well-nested dependency graphs with gap degree or edge degree d. All projective dependency graphs ($d = 0$) are trivially well-nested.

Well-nestedness also brings computational benefits. In particular, chart-based parsers for grammar formalisms in which derivations obey the well-nestedness constraint (such as Tree Adjoining Grammar) are not hampered by the 'crossing configurations' to which Satta (1992) attributes the fact that the universal recognition problem of Linear Context-Free Rewriting Systems is $\mathcal{NP}$-complete.

4 Experimental evaluation

In this section, we present an experimental evaluation of planarity, well-nestedness, gap degree, and edge degree, by examining how large a proportion of the structures found in two dependency treebanks are allowed under different constraints. Assuming that the treebank structures are sampled from naturally occurring structures in natural language, this provides an indirect evaluation of the linguistic adequacy of the different proposals.

4.1 Experimental setup

The experiments are based on data from the Prague Dependency Treebank (PDT) (Hajič et al., 2001) and the Danish Dependency Treebank (DDT) (Kromann, 2003). PDT contains 1.5M words of newspaper text, annotated in three layers according to the theoretical framework of Functional Generative Description (Böhmová et al., 2003). Our experiments concern only the analytical layer, and are based on the dedicated training section of the treebank. DDT comprises 100k words of text selected from the Danish PAROLE corpus, with annotation

Table 1: Experimental results for DDT and PDT

property	DDT		PDT	
all structures	$n = 4393$		$n = 73088$	
gap degree 0	3732	84.95%	56168	76.85%
gap degree 1	654	14.89%	16608	22.72%
gap degree 2	7	0.16%	307	0.42%
gap degree 3	–	–	4	0.01%
gap degree 4	–	–	1	< 0.01%
edge degree 0	3732	84.95%	56168	76.85%
edge degree 1	584	13.29%	16585	22.69%
edge degree 2	58	1.32%	259	0.35%
edge degree 3	17	0.39%	63	0.09%
edge degree 4	2	0.05%	10	0.01%
edge degree 5	–	–	2	< 0.01%
edge degree 6	–	–	1	< 0.01%
projective	3732	84.95%	56168	76.85%
planar	3796	86.41%	60048	82.16%
well-nested	4388	99.89%	73010	99.89%
non-projective structures only	$n = 661$		$n = 16920$	
planar	64	9.68%	3880	22.93%
well-nested	656	99.24%	16842	99.54%

of primary and secondary dependencies based on Discontinuous Grammar (Kromann, 2003). Only primary dependencies are considered in the experiments, which are based on the entire treebank.[4]

4.2 Results

The results of our experiments are given in Table 1. For the binary constraints (planarity, well-nestedness), we simply report the number and percentage of structures in each data set that satisfy the constraint. For the parametric constraints (gap degree, edge degree), we report the number and percentage of structures having degree d ($d \geq 0$), where degree 0 is equivalent (for both gap degree and edge degree) to projectivity.

For DDT, we see that about 15% of all analyses are non-projective. The minimal degree of non-projectivity required to cover all of the data is 2 in the case of gap degree and 4 in the case of edge degree. For both measures, the number of structures drops quickly as the degree increases. (As an example, only 7 or 0.17% of the analyses in DDT have gap degree 2.) Regarding the binary constraints, we find that planarity accounts for slightly more than the projective structures (86.41% of the data is planar), while almost all structures in DDT (99.89%) meet the well-nestedness constraint. The difference between the two constraints becomes clearer when we base the figures on the set of non-projective structures only: out of these, less than 10% are planar, while more than 99% are well-nested.

For PDT, both the number of non-projective structures (around 23%) and the minimal degrees of non-projectivity required to cover the full data (gap degree 4 and edge degree 6) are higher than in DDT. The proportion of planar analyses is smaller than in DDT if we base it on the set of all structures (82.16%), but significantly larger when based on the set of non-projective structures only (22.93%). However, this is still very far from the well-nestedness constraint, which has almost perfect coverage on both data sets.

4.3 Discussion

As a general result, our experiments confirm previous studies on non-projective dependency parsing (Nivre and Nilsson, 2005; Hall and Novák, 2005;

[4] A total number of 17 analyses in DDT were excluded because they either had more than one root node, or violated the indegree constraint. (Both cases are annotation errors.)

McDonald and Pereira, 2006): The phenomenon of non-projectivity cannot be ignored without also ignoring a significant portion of real-world data (around 15% for DDT, and 23% for PDT). At the same time, already a small step beyond projectivity accounts for almost all of the structures occurring in these treebanks.

More specifically, we find that already an edge degree restriction of $d \leq 1$ covers 98.24% of DDT and 99.54% of PDT, while the same restriction on the gap degree scale achieves a coverage of 99.84% (DDT) and 99.57% (PDT). Together with the previous evidence that both measures also have computational advantages, this provides a strong indication for the usefulness of these constraints in the context of non-projective dependency parsing.

When we compare the two graded constraints to each other, we find that the gap degree measure partitions the data into less and larger clusters than the edge degree, which may be an advantage in the context of using the degree constraints as features in a data-driven approach towards parsing. However, our purely quantitative experiments cannot answer the question, which of the two measures yields the more informative clusters.

The planarity constraint appears to be of little use as a generalization of projectivity: enforcing it excludes more than 75% of the non-projective data in PDT, and 90% of the data in DDT. The relatively large difference in coverage between the two treebanks may at least partially be explained with their different annotation schemes for sentence-final punctuation. In DDT, sentence-final punctuation marks are annotated as dependents of the main verb of a dependency nexus. This, as we have discussed above, places severe restrictions on permitted forms of non-projectivity in the remaining sentence, as every discontinuity that includes the main verb must also include the dependent punctuation marks. On the other hand, in PDT, a sentence-final punctuation mark is annotated as a separate root node with no dependents. This scheme does not restrict the remaining discontinuities at all.

In contrast to planarity, the well-nestedness constraint appears to constitute a very attractive extension of projectivity. For one thing, the almost perfect coverage of well-nestedness on DDT and PDT (99.89%) could by no means be expected on purely combinatorial grounds—only 7% of all possible dependency structures for sentences of length 17 (the average sentence length in PDT), and only slightly more than 5% of all possible dependency structures for sentences of length 18 (the average sentence length in DDT) are well-nested.[5] Moreover, a cursory inspection of the few problematic cases in DDT indicates that violations of the well-nestedness constraint may, at least in part, be due to properties of the annotation scheme, such as the analysis of punctuation in quotations. However, a more detailed analysis of the data from both treebanks is needed before any stronger conclusions can be drawn concerning well-nestedness.

5 Conclusion

In this paper, we have reviewed a number of proposals for the characterization of mildly non-projective dependency structures, motivated by the need to find a better balance between expressivity and complexity than that offered by either strictly projective or unrestricted non-projective structures. Experimental evaluation based on data from two treebanks shows, that a combination of the well-nestedness constraint and parametric constraints on discontinuity (formalized either as gap degree or edge degree) gives a very good fit with the empirical linguistic data. Important goals for future work are to widen the empirical basis by investigating more languages, and to perform a more detailed analysis of linguistic phenomena that violate certain constraints. Another important line of research is the integration of these constraints into parsing algorithms for non-projective dependency structures, potentially leading to a better trade-off between accuracy and efficiency than that obtained with existing methods.

Acknowledgements We thank three anonymous reviewers of this paper for their comments. The work of Marco Kuhlmann is funded by the Collaborative Research Centre 378 'Resource-Adaptive Cognitive Processes' of the Deutsche Forschungsgemeinschaft. The work of Joakim Nivre is partially supported by the Swedish Research Council.

[5] The number of unrestricted dependency trees on n nodes is given by Sequence A000169, the number of well-nested dependency trees is given by Sequence A113882 in the On-Line Encyclopedia of Integer Sequences (Sloane, 2006).

References

Manuel Bodirsky, Marco Kuhlmann, and Mathias Möhl. 2005. Well-nested drawings as models of syntactic structure. In *Tenth Conference on Formal Grammar and Ninth Meeting on Mathematics of Language*.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The Prague Dependency Treebank: A three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, pages 103–127. Kluwer Academic Publishers.

Michael Collins, Jan Hajič, Eric Brill, Lance Ramshaw, and Christoph Tillmann. 1999. A statistical parser for Czech. In *37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 505–512.

Denys Duchier and Ralph Debusmann. 2001. Topological dependency trees: A constraint-based account of linear precedence. In *39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 180–187.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *16th International Conference on Computational Linguistics (COLING)*, pages 340–345.

Gülsen Eryiğit and Kemal Oflazer. 2006. Statistical dependency parsing of turkish. In *Eleventh Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Haim Gaifman. 1965. Dependency systems and phrase-structure systems. *Information and Control*, 8:304–337.

Jan Hajič, Barbora Vidova Hladka, Jarmila Panevová, Eva Hajičová, Petr Sgall, and Petr Pajas. 2001. Prague Dependency Treebank 1.0. LDC, 2001T10.

Keith Hall and Vaclav Novák. 2005. Corrective modeling for non-projective dependency parsing. In *Ninth International Workshop on Parsing Technologies (IWPT)*.

Richard Hudson. 2003. An encyclopedia of English grammar and Word Grammar. http://www.phon.ucl.ac.uk/home/dick/enc/intro.htm, January.

Sylvain Kahane, Alexis Nasr, and Owen Rambow. 1998. Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *36th Annual Meeting of the Association for Computational Linguistics and 18th International Conference on Computational Linguistics (COLING-ACL)*, pages 646–652.

Matthias Trautner Kromann. 2003. The Danish Dependency Treebank and the DTAG treebank tool. In *Second Workshop on Treebanks and Linguistic Theories (TLT)*, pages 217–220.

Svetoslav Marinov and Joakim Nivre. 2005. A data-driven parser for Bulgarian. In *Fourth Workshop on Treebanks and Linguistic Theories (TLT)*, pages 89–100.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Eleventh Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *43rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Eigth International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

Joakim Nivre. 2006. Constraints on non-projective dependency parsing. In *Eleventh Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

T. Obrębski and F. Graliński. 2004. Some notes on generative capacity of dependency grammar. In *COLING 2004 Workshop on Recent Advances in Dependency Grammar Workshop on Recent Advances in Dependency Grammar*.

Martin Plátek, Tomáš Holan, and Vladislav Kuboň. 2001. On relax-ability of word order by d-grammars. In *Third International Conference on Discrete Mathematics and Theoretical Computer Science*.

Giorgio Satta. 1992. Recognition of linear context-free rewriting systems. In *30th Meeting of the Association for Computational Linguistics (ACL)*, pages 89–95, Newark, Delaware, USA.

Daniel Sleator and Davy Temperley. 1993. Parsing English with a link grammar. In *Third International Workshop on Parsing Technologies*.

Neil J. A. Sloane. 2006. The on-line encyclopedia of integer sequences. Published electronically at http://www.research.att.com/ njas/sequences/.

Anssi Yli-Jyrä. 2003. Multiplanarity – a model for dependency structures in treebanks. In *Second Workshop on Treebanks and Linguistic Theories (TLT)*, pages 189–200.

Daniel Zeman. 2004. *Parsing With a Statistical Dependency Model*. Ph.D. thesis, Charles University, Prague, Czech Republic.