Multiplanarity – a Model for Dependency Structures in Treebanks

Anssi Yli-Jyrä
Department of General Linguistics, University of Helsinki, Finland
anssi.yli-jyra@helsinki.fi

1 Introduction

The number of treebanks available for different languages is growing steadily. A considerable portion of the recent treebanks use annotation schemes that are based on dependency syntax. In this paper, we give a model for linguistically adequate classes of dependency structures in treebanks. Our model is tested using the Danish Dependency Treebank [13].

The modern dependency syntax was pioneered by Tesnière [27]. His core concepts, binary dependencies and unique heads, are mostly shared in the recent dependency syntactic theories [6, 9, 18, 24, 26]. These theories stress the functional structure, while paying much less attention to linear word order.

Lecerf's *projectivity* hypothesis [15, 17] assumes a constraint on linear word-order in dependency analyses. It says that if a word A depends directly on a word B and some word C intervenes between them in linear order, then C depends directly on A or on B or some other intervening word[23]. The projectivity constraint has been a popular simplification in many computational dependency grammars since 1960's [4]. It does not only make parsing algorithms simple and efficient [16], but also equips us with neat ways to visualize analyses, with an equivalent, constituent based representation for dependency trees [4, 7], and with a criterion for stylistically marked analyses [27, 22] and for abnormal information structure [10]. The tendency that sentences admit projective analyses has been observed in many languages, including French [15], Swedish [20], Finnish [1], and Turkish [21].

Unfortunately, projectivity does not lend itself to adequate treatment of certain non-local syntactic phenomena which are extensively studied in the literature of constituent-based theories such as TG, GB, GPSG, TAG, and LFG. Among these

phenomena are scrambling, topicalizations, WH-movements, cleft sentences, discontinuous NPs, and discontinuous negation.

While projectivity is a linguistically motivated preference for simpler analyses, Tesnière's dependency syntax is *non-projective*. Dependency syntactic accounts of non-projective word-order are usually based on simultaneous specification of syntactic order and linear order (eg. [9, 10]) or on mechanisms relating syntactic tree and the surface (eg. [18]). However, there is a tremendous need for a simple theory-independent model that would somehow generalize the projectivity constraint to non-projective analyses.

Earlier, a few relaxed models somewhat similar to projectivity have been proposed. These include *quasi-projectivity* [17], *planarity* [25], *pseudo-projectivity* [12], *meta-projectivity* [19], and *polarized dependency grammars* [3]. None of the these models is motivated by formal language theory.

The current work presents a new word-order model with a clear connection to formal language theory. The model, *multiplanarity with a bounded number of planes*, is based on planarity [25], which is itself a generalization of projectivity.

The multiplanarity is defined in Section 2. In Section 3 we restrict sets of multiplanar analyses to a mildly context-sensitive subset. Section 4 formularizes some more standard constraints such as head-uniqueness and the existence of a root. The adequateness of the restricted multiplanarity property is evaluated in Section 5.

2 Dependency Graphs

Consider a sentence $S = w_1 w_2 ... w_n$ and the set of words $W = \{w_1, w_2, \ldots, w_n\}$ of the sentence S. Assume an (unlabeled) syntactic structure given by the binary relation w_i **governs** w_j. We define that w_i **linked** w_j if and only if w_i **governs** $w_j \vee$ w_j **governs** w_i. We define that w_i **prec** w_j if $i + 1 = j$. For any binary relation R, let R^* denote its reflexive transitive closure.

A *dependency graph of the sentence* S is a directed graph $G = (W, \textbf{governs})$, where the set of words W is its set of nodes and the **governs** is its set of arcs. A *dependency graph with linear precedence* or *dependency analysis* is a directed graph $H = (W, \textbf{prec} \cup \textbf{governs})$, where **governs** $\cup$ **prec** is the set of arcs in H.

Projectivity

The standard definition of projectivity constraint [4] is as follows: if A **prec*** B and B **prec*** C and A **linked** C then A **governs*** B or C **governs*** B. Projectivity forbids all crossing dependencies and cycles of length 3 or more [20]. However,

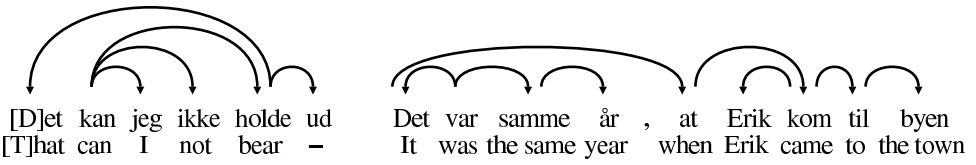
projectivity implies neither that the graph be connected nor that it be acyclic.

Planarity

Planarity of a dependency tree is the requirement that the links do not cross when drawn above the sentence [25]. Despite of its similarity with projectivity, it is a looser property [7]. Formally, the *planarity* restriction says: if A **prec*** B and B **prec*** C and A **linked** C then A **linked*** B or C **linked*** B. It should be noted that a planar dependency graph may have cycles of any length.

(not true)

Planarity is a linguistically well-motivated relaxation to the projectivity constraint. In the Danish Dependency Treebank [13], for example, there are trees that are planar, but not projective (Figure 1).



(typo) these are planar but not projective

Figure 1: These sentences are not planar because both of them have a pair of linked nodes (det .. holde, det .. at) that are governed by an intervening node (kan, var).

Multiplanarity

We have chosen to use planarity as the core notion that we generalize in the following definition: A dependency graph with linear precedence, $H = (W, \textbf{prec} \cup \textbf{governs})$, is *$m$-planar*, if the arcs in **prec** $\cup$ **governs** can be divided into m disjoint sets **prec**, $\textbf{governs}_1$, $\textbf{governs}_2$, $\ldots$, $\textbf{governs}_m$ such that for all $i = 1..m$, the subgraph $H_i = (W, \textbf{prec} \cup \textbf{governs}_i)$ is planar.

If a class C of dependency graphs is m-planar for some m, we say that the class C is a *multiplanar with a bounded number of planes*. Multiple planes adds a discrete third dimension. The arcs, or links in different planes are arranged together by means of the first two dimensions shared by the subgraphs (Figure 2).

A linguistically interesting problem is to find out if there is a bound for the number of planes needed to give dependency analyses for natural language sentences.

The sequential presentation

Each planar dependency graph $H = (W, \textbf{prec} \cup \textbf{governs})$ can be represented as a *bracketed string*. In order to construct such a presentation, take the string of

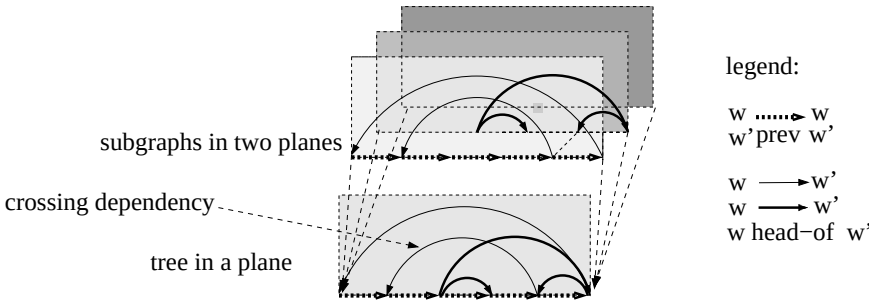


Figure 2: We represent crossing arcs in separate planes. More expressive power is gained by laying a finite number of planes above each other. Crucially, linear order of words is part of each subgraph.

graph nodes, i.e. the sentence $w_1 w_2 \dots w_n$. Immediately before each word w_i in the string, insert a right bracket for every dependency link connecting the word to something on the left. Immediately after each word w_i in the string, insert a left bracket for every dependency link connecting the word to something on the right. To indicate the direction of the dependency links, two different sets of marked brackets are used: when the head is on the left end of the link, one set of brackets, $\{[?,]\}$, is used; when the head is on the right end of the link, the other one, $\{[, ?]\}$, is used;

Now, if we have a dependency graph that is m-planar, we can take each of the m subgraphs $H_i = (W, \mathbf{prec} \cup \mathbf{governs}_i)$ and represent it by means of brackets. If distinguished sets of brackets are used in the encoding of each subgraph, all the brackets can be inserted into a single copy of the sentence S. Let λ denote any string of left brackets and ρ any string of right brackets. The structure of the resulting bracketed string will be as follows:

$$w_1 \lambda \rho w_2 \lambda \rho w_3 \lambda ... \rho w_{n-1} \lambda \rho w_n .$$

When the dependencies have labels as they do in the usual annotation, we can indicate the corresponding label in the brackets. We just do not want to complicate the model of the word-order. In figures of this paper, we indicate the label in the bracket on the dependent's side (Figure 3).

The correspondence to multi-pushdown automata

Our representation of dependency graphs by means of distinguished brackets is actually a simulation of a multi-pushdown automaton, stack alphabets of which contain only one symbol. Informally, an opening bracket corresponds to a PUSH

[mod ?][mod ?][mod ?][subj ?]

Figure 3: The analogy of bracketed string and dependency links. In more complicated analyses (eg. Figure 4), the sequences of closing brackets are stacked downwards under the corresponding word and the sequences of opening brackets are stacked upwards under the corresponding word. The highest plane occupies the topmost lines under the sentence, while the lines of the first plane are in the bottom.

operation, which writes a stack symbol to a pushdown, and a closing bracket corresponds to a POP operation, which reads a symbol from a pushdown. The dependency graphs, which we presented as unlabeled, can obviously also be extended so that each dependency type has a unique label, corresponding to one of the symbols in an extended stack alphabet of a pushdown automaton. In any case, using an automaton with several pushdowns is complexity-theoretically dangerous, because two pushdowns (stacks) can simulate movements of a Turing machine [8].

3 Alignment constraints

Joshi has presented the hypothesis [11] that grammars of the so-called *mildly context-sensitive languages (MCSLs)* are capable of associating correct syntactic structures to natural language sentences. One of the defining criteria for MCSLs is that the kinds of dependencies captured by the class are restricted to nested dependencies and to certain crossing dependencies. These exclude the crossing dependencies of the so-called MIX language, the strings of which consist of an equal number of a's, b's, and c's that may be in any order.

Multiplanarity is a very loose constraint: for every dependency graph H, there exists an interger m such that H is m-planar. Therefore, if we want to describe *only* MCSLs, we have to assume a bound for the number of planes allowed, or we have to give some other sufficient restriction. In formal language theory, restrictions to the pushdown operations have been proposed. For instance, *Extended Linear Indexed Grammars* (ELIGs) [28], which can be implemented with a multi-pushdown automaton, impose the following restriction: "you may pop symbols from any pushdown what you want, but you must push symbols to the same pushdown than last time, or to start using the next pushdown in the predefined order". It has also been known for a while that Linear Indexed Grammars (LIGs) [5] are among the grammars that generate MCSLs.

(typo) I meant grammars with composite storages, (Wartena 2001)

Following the technique applied in ELIGs, we give the following alignment constraints on the multiplanar dependency graphs:

Plane Locking If there are words w_i, w_j and w_k such that $i < j < k$, and a plane p_u containing a link l_1 that connects words w_i and w_k, the a plane p_v, $v < u$, is not allowed to contain any link l_2 such that l_2 connects the word w_j to the right.

Left Conjoin There are no pair of links l_1 and l_2 both connecting a word w_i to the right so that the links l_1 and l_2 occupy different planes;

Continuous Tiling For any word w_i, there is no link connecting it to the right in the plane p, if the plane p is not the bottom plane and there is no link l in the planes p and $p-1$ connecting a pair of words w_j and w_k such that $j < i < k$.

The last constraint is not necessary, but it is added in order to minimize the number of planes. The three constraints given here can be applied to a dependency analysis in order to determine the arrangement of links in different planes and the minimal number of planes needed when the constraints are satisfied. Moreover, it seems feasible to incorporate these constraints into a dependency parser that implements a multi-stack automaton.

Kruijff [14] has pointed out the need of a language theoretic complexity hierarchy for dependency grammars. It seems that comparing multiplanar dependency grammars (with the alignment constraints) to extended linear indexed grammars is a possible way to establish a language theoretic complexity hierarchy for dependency grammars, because Wartena [28] has shown that extended linear indexed grammars with n index stacks (pushdowns) give rise to the nth member of the Weir's hierarchy [29] of mildly context-sensitive grammars.

4 Other constraints

We will now give more familiar constraints that apply to dependency syntactic structures. These constraints are known under many different names, but there is a high degree of agreement on them in dependency syntactic theories:

Single Head Each word has at most one head;

Unique Root All the words in the sentence have a head, except the root which is unique;

No Self-Dependencies No word depends on itself.

In addition to these, it is often required in the dependency syntactic theories that the dependency graph does not contain longer cycles either. Because we already

know, that each word has at most one head, there are two possibilities. Either the graph is cyclic and unconnected, or it is acyclic and connected. In the latter case the graph is a tree. The following constraint is implies acyclicity in a dependency structure. Although it cannot detect acyclicity in all cases, it licenses us to describe a large class of dependency trees without resorting to the acyclicity test, which is of a global nature.

Cycle-Cutting If there are words w_i, w_j and w_k such that $i < j, k$, w_i **governs** w_j, and w_k **governs** w_i, then one of the following conditions hold:

- for every word w_r, it is not the case that w_r **governs** w_k.
- for every word w_r, it is not the case that w_j **governs** w_r.
- There is an index s, $min(j,k) < s \leq \max(j,k)$ such that it is not the case that there are indices t, u, v, x such that $t, u < s \leq v, x$ and w_t **governs** w_v and w_x **governs** w_u.

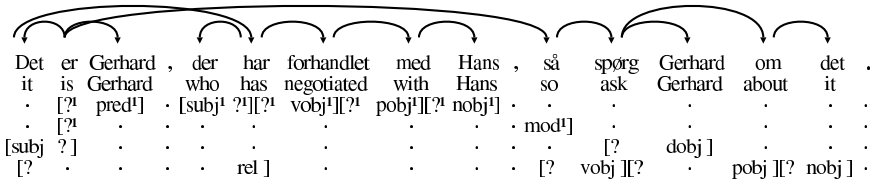


Figure 4: An analysis that satisfies the Cycle-Cutting constraint (consider w_i="er" or w_i="sporg") (tr. "It is Gerhard who has negotiated with Hans Jorgen, so ask him about it.")

5 The Danish data

In order to see how our model fits actual data we carried out experiments with the Danish Dependency Treebank (DDT) [13]. The treebank consists of 5540 sentences and 100200 words. The dependency graphs in DDT originally contained secondary, referentially motivated links, called SRC,REF, and FILLC. We discarded these links, because they provided secondary heads to some words. We also removed all dependency links for punctuation marks, because we suspected that their annotation is quite dependent on conventions. Our position was that the function of punctuation marks is related to their linear position rather than to a head-modifier relationship. If they, however, were included, one or two more planes would have been needed, because connecting the root verb to the sentence boundary enforces projectivity to planar analyses [7].

We observed in DDT that crossing dependencies were typically caused by discontinuous coordinators, relative clauses, and cleft sentences. A surprising observation was that these phenomena combined relatively often in such a way that two or more crossing dependencies resulted. There were only two sentences that were not multiplanar with three planes (Figure 5).

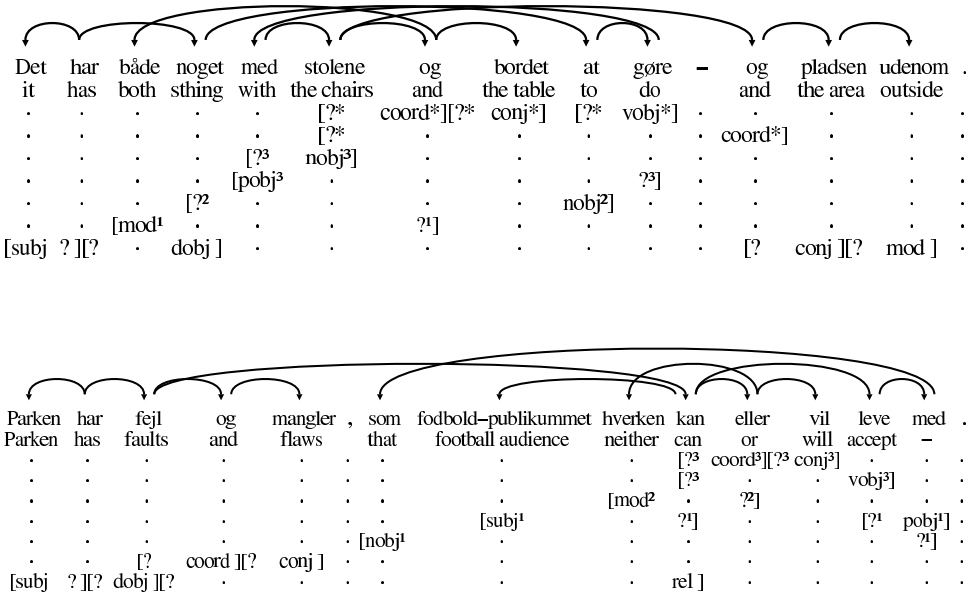


Figure 5: Two complex sentences that are not 3-planar under the constraints of the section 4. (tr. “It has something to do both with the chairs and the table – and the area outside.” “Parken [the Copenhagen football stadion] has faults and flaws that football audience neither can nor will accept.”)

We made an unsuccessful attempt to prove that different planes correspond to some kind of hierarchy of dependency types, as it was our original guess. Nevertheless, it became clear in our model that a higher plane is typically introduced after an interrogative pronoun, informal subject or some other occurrence of a *function word*.

The sentences like those in Figure 1 are not projective. We tried to arrange the non-projective links to different planes so that projectivity was satisfied separately in each plane. However, this kind of strategy resulted into many more planes, which we considered worse than adoption of planarity.

We also tried other ways to generalize the notion of projectivity to alignment planes. For example, we examined the direction of dependencies between two planes. Unfortunately, we observed in our model that a governor of a lower plane was sometimes governed by a governor in the higher plane, and sometimes it was the other way round. It became obvious the generalized projectivity does not work

out very well with our alignment constraints. Arranging dependencies to a minimal number of planes while keeping in mind the generalized projectivity constraint probably requires a constraint solver, which we did not use at the time of the current experiments.

It should be noted that the failure of some tested hypotheses in this study depend crucially on the actual choice of the collection of constraints. We argue, however, that our collection of constraints is quite robust, for two reasons: (i) they seem to minimize the number planes, and, (ii) the sufficient condition for acyclicity of dependency graphs held almost in 100% of the cases. In addition to the special cases involving the dependencies *qobj* (direct quotations) and *voc* (vocatives), there were only two sentences that did not satisfy the Cycle Cutting constraint (Figure 6).

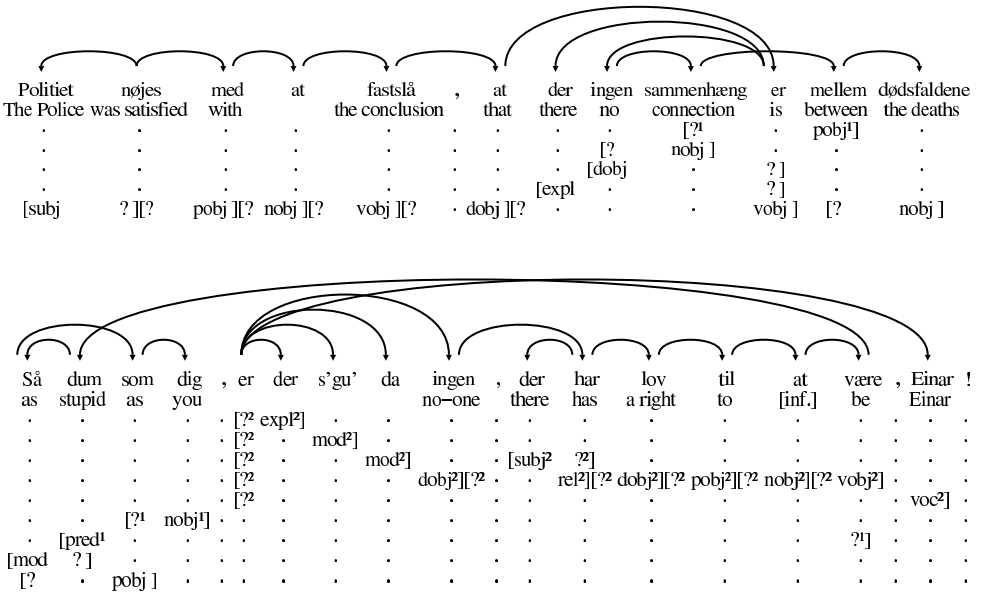


Figure 6: Two sentences that failed to satisfy the Cycle Cutting-constraint (with w_i="ingen", or w_i="er"). (tr. "The police was satisfied with the conclusion that there is no connection with the deaths." "No-one is allowed to be as stupid as you, Einar.")

6 Concluding remarks

We have studied a hypothesis of constraints that can be imposed on non-projective dependency analyses. In its most general form, our relaxed version of projectivity, called *multiplanarity*, is a very wide characterization for structures. It accepts not only dependency trees, but also more general dependency graphs. The relevance

of the unrestricted model is questioned by the considerations related to formal language theory. This motivates a restricted model of multiplanarity, based on *alignment constraints* between separate planes, similar to certain restrictions used in Extended Linear Indexed Grammars [28].

(typo) wrong ref.

In the restrictions imposed to define dependency *trees*, we did not make use of the acyclicity and connectedness constraints, which are often stipulated in dependency parsing [20, 2]. Instead, a reasonably good coverage was achieved by a *sufficient* condition for acyclicity.

The final model was tested with the Danish Dependency Treebank [13], and it produced mainly very shallow multiplanar representations. The simplicity of our model suggests many applications in processing of dependency trees. The model can be used, for example, as a tool for validation of dependency structures in dependency treebanks. It is also useful in data mining: it provides us with a way to find out non-prototypical analyses. Our model suggests a basis for the complexity hierarchy of dependency grammars, and motivates a class of efficient parsers that cope with mildly context-sensitive sets of non-projective dependency trees. Finally, it seems feasible that such a parser can be easily approximated using finite state methods. All these possibilities may have an effect on the way the treebanks are built in the future.

References

[1] Harri Arnola. On parsing binary dependency structures deterministically in linear time. In *Workshop on Processing of Dependency-Based Grammars (COLING-ACL'98)*, pages 68–77, Université de Montréal, Quebec, Canada, 1998.

[2] Ralph Debusmann and Denys Duchier. A meta-grammatical framework for dependency grammar. In *ACL'03*, Budapest, Hungary, 2003.

[3] Alexander Dikovski. Polarized non-projective dependency grammars. In P. de Groote, G. Morril, and C. Retoré, editors, *LACL 2001*, *LNAI 2099*, pages 139–17. Springer-Verlag Berlin Heidelberg, 2001.

[4] H. Gaifman. Dependency systems and phrase-structure systems. *Information and Control*, 8(3):304–337, 1965.

[5] Gerald Gazdar. Applicability of indexed grammars to natural languages. In U. Reyle and C. Rohrer, editors, *Natural Language Parsing and Linguistic Theories*, pages 69–94. Reidel, Dordrecht, 1988.

[6] Eva Hajičová. "Free" word order described without unnecessary complexity. *Theoretical Linguistics*, 17:99–106, 1991.

[7] Stefan Höfler. Link2tree: A dependency-constituency converter. Lizentiatsarbeit, Institute of Computational Linguistics, University of Zürich, 2002.

[8] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley, Reading, MA, 1979.

[9] Richard Hudson. *English Word Grammar*. Basil Blackwell, Oxford, UK and Cambridge, MA, 1991.

[10] Timo Järvinen and Pasi Tapanainen. Towards an implementable dependency grammar. In Sylvain Kahane and Alain Polguère, editors, *Processing of Dependency-Based Grammars, COLING-ACL'98*, pages 1–10, Montreal, Cananda, 1998.

[11] Aravind K. Joshi. Tree Adjoining Grammars: how much context-sensitivity is required to provide reasonable structural descriptions? In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, Cmabridge, 1985.

[12] Sylvain Kahane, Alexis Nasr, and Owen Rambow. Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *COLING-ACL'98*, volume I, pages 646–652, 1998.

[13] M. T. Kromann, L. Mikkelsen, and S. K. Lynge. The Danish Dependency Treebank Website. Dept. of Computational Linguistics, Copenhagen Business School. `http://www.id.cbs.dk/~mtk/treebank`, 2003.

[14] Geertjan Kruijff. A formal language hierarchy based on dependency grammars. MSc Thesis proposal. Universität des Saarlandes, Posted 26 April, 2002. Revised 8 October 2002, 2002.

[15] Yves Lecerf. Programme des conflits, modèle des conflits. *Bulletin bimestriel de l'ATALA*, 4,5, 1960.

[16] Vincenzo Lombardo and Leonardo Lesmo. An Earley-type recognizer for dependency grammar. In *16th COLING*, volume 2, pages 723–728, 1996.

[17] Solomon Marcus. Sur la notion de projectivité. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 11:181–192, 1965.

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

[19] Alexis Nasr. *Un système de reformulation automatique de phrases fondé sur la Théorie Sens-Texte: application aux langues contrôlées*. PhD thesis, Université Paris 7, 1996.

[20] Joakim Nivre. An efficient algorithm for projective dependency parsing. In *8th Int'l Workshop on Parsing Technologies (IWPT 03)*, Nancy, France, 2003.

[21] Kemal Oflazer. Dependency parsing with an extended finite state approach. In *Proceedings of the 37th ACL*, Maryland, USA, 1999.

[22] Vladimir Pericliev and Ilarion Ilarionov. Testing the projectivity hypothesis. In *11th COLING*, pages 56–58, Budapest, 1986.

[23] Jane J. Robinson. Dependency structures and transformational rules. *Language*, 46(2):259–285, 1970.

[24] Petr Sgall, Eva Hajičová, and Jarmila Panevová. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. D. Reidel Publishing Company, Dordrecht, Boston, London, 1986.

[25] Daniel Sleator and Davy Temperley. Parsing english with a link grammar. In *3rd International Workshop on Parsing Technologies*, pages 277–291, 1993.

[26] Stanley Starosta. *The Case for Lexicase: An Outline of Lexicase Grammatical Theory*. Pinter Publishers, London and New York, 1988.

[27] Lucien Tesnière. *Éléments de Syntaxe Structurale*. Editions Klincksieck, Paris, 1959.

[28] Christian Wartena. Extending linear indexed grammars. In *Proceedings of the 5th Workshop on Tree-Adjoining Grammars and Related Formalisms (TAG+5)*, Paris, 2000.

[29] David Weir. A geometric hierarchy beyond context-free languages. *Theoretical Computer Science*, 104(4):235–261, 1992.

[This version: recompiled and the errata added in May 2005 by Anssi Yli-Jyrä]