Movement as well-formedness conditions

Ralph Debusmann
University of Saarland
rade@coli.uni-sb.de

Abstract. We introduce a new formulation of dependency grammar recently suggested in (Duchier and Debusmann 2001) (henceforth DD). DD shares with GB (Chomsky 1986) a notion of *movement*. In GB, movement is carried out by tree transformations. In DD, it is the effect of well-formedness conditions on dependency trees and does not involve transformations. We illustrate both kinds of movement by showing how both theories analyze German verb-second clauses. Then, we point out the similarities between GB and DD, and raise the question whether GB's transformational notion of movement could be replaced by DD's constraint-based account of movement.

1 Introduction

In this article, we introduce a new formulation of dependency grammar recently suggested by (Duchier and Debusmann 2001) (henceforth DD). One of the key assets of DD is that its axiomatization can be (and has been) turned into efficient constraint-based parsers. DD analyses consist of two tree structures: a *syntactic dependency tree* (*ID tree*) and a *topological dependency tree* (*LP tree*). The ID tree is a dependency tree in the spirit of (Tesnière 1959) whose edges are labeled by grammatical roles. ID trees are unordered (and hence in a sense non-projective), as opposed to LP trees which are ordered and projective. The LP tree is inspired by topological fields theory. Its shape is essentially a flattening of the ID tree, allowing us to handle discontinuous constructions in free word order languages such as German.

The approach taken in DD is very similar to other recent dependency-based approaches tackling discontinuity, such as (Bröker 1998), (Kahane et al. 1998) and (Gerdes and Kahane 2001). It is also reminiscent of HPSG-based theories by (Reape 1994), (Kathol 2000) and (Müller 1999). DD's strong resemblance with Government-Binding (GB) theory (Chomsky 1986) is probably less obvious. We will show that, above all, DD shares with GB a notion of *movement*. In GB, movement is carried out by tree transformations, which have properties undesirable for parsing. In DD, movement is

the effect of well-formedness conditions on finite labeled trees and does not involve transformations.

The outline of this article is as follows. We start out by presenting the essentials of topological fields theory in section 2. In section 3, we introduce the dependency-based theory proposed in (Duchier and Debusmann 2001), and show how this theory handles German verb-second clauses. In section 4, we explain an analysis of German verb-second clauses in GB, as put forward by (Grewendorf 1988). Then, in section 5, we compare GB and DD, and hint at the possibility to reformulate parts of GB, movement in particular, in a constraint-based way. The conclusion in section 6 rounds up the article.

2 Topological fields theory

Both GB and DD use ideas from *topological fields theory*. Topological fields theory has a long tradition in German linguistics reaching back to the works of (Herling 1821) and (Erdmann 1886). As an example, consider the sentence below:

Einen	Mann	hat	Maria	geliebt	
A	man(acc)	has	Maria	loved	(1.1)

"A man, Maria has loved."

Topological fields theory divides (1.1) into four parts, which are called *fields*: Vorfeld, left parenthesis, Mittelfeld and right parenthesis:[1]

Vorfeld	left parenthesis	Mittelfeld	right parenthesis
Einen Mann	*hat*	*Maria*	*geliebt.*

where the finite verb *hat* in the left parenthesis and its verbal complement *geliebt* (right parenthesis) form a *bracket* around the non-verbal material in the *Mittelfeld*. The *Vorfeld*, left of the left parenthesis, can be occupied by at most one topicalized constituent, whereas the *Mittelfeld* can host any number of non-verbs. The order of the material in the Mittelfeld is almost arbitrary.

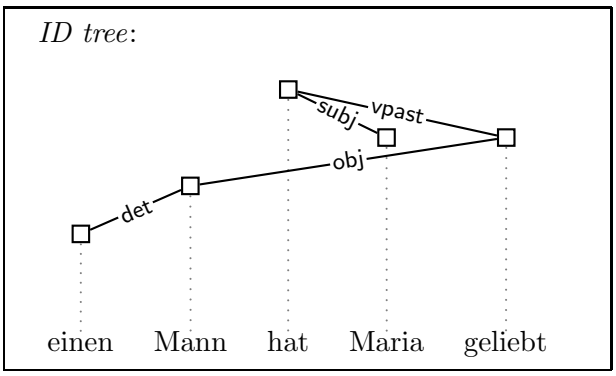
3 Verb-second clauses: A DD analysis

3.1 ID and LP trees

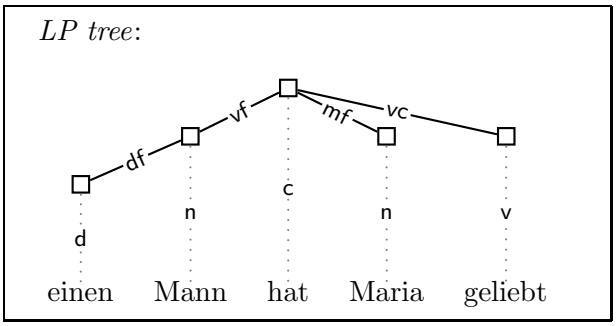
DD distinguishes two tree structures: the unordered ID tree and the partially ordered and projective LP tree. ID and LP trees share the same set of nodes, which correspond one-to-one with words, but have different edges. Below, we give an ID tree analysis of (1.1). Since ID trees are unordered, we can

[1] Actually, topological fields theory postulates one more field which is called *Nachfeld* and hosts material right of the right parenthesis.

pick an arbitrary linear arrangement for display purposes. In the picture below, we stick to the word order given in sentence (1.1).



ID edges are labelled by grammatical functions like subj (for a nominative subject), obj (for an accusative object) vpast (for a past participle complement) and det (for a determiner). The mother of a node in the ID tree is called *head* and its daughters *syntactic dependents*. Here is the corresponding topological dependency (LP tree) analysis:



The mother of a node in the LP tree is called *host* and its daughters *topological dependents*.

3.2 Ordering words in the LP tree

DD employs a set $\mathcal{F}$ of *fields* to determine the licensed linearizations. $\mathcal{F} = \mathcal{F}_{\text{ext}} \uplus \mathcal{F}_{\text{int}}$, where $\mathcal{F}_{\text{ext}} = \{\mathsf{df}, \mathsf{vf}, \mathsf{mf}, \mathsf{vc}\}$ is the set of *external fields* or LP edge labels. $\mathcal{F}_{\text{int}} = \{\mathsf{d}, \mathsf{n}, \mathsf{c}, \mathsf{v}\}$ is the set of *internal fields* or LP node labels.[2] $\mathcal{F}$ is totally ordered, which induces a partial order on LP trees:

1. The topological dependents of each node are ordered by their edge labels or external fields.

2. Each node is positioned with respect to its topological dependents by its node label or internal field.

[2]Note that for expository reasons, $\mathcal{F}$ only includes the fields needed to account for our example. For a full account of German, more fields are required.

The order is partial and not total because if two words land[3] in the same external field, they remain unordered with respect to each other.

The set $\mathcal{F}$ of fields is essentially motivated by topological fields theory. For example vf models the Vorfeld, c the left parenthesis, mf the Mittelfeld and vc the right parenthesis (or *verb cluster*). df and n are used to determine word order within noun phrases: e.g. df stands for *determiner field* and n for *noun field*. The total order on $\mathcal{F}$ is given below:

$$\mathsf{d} \prec \mathsf{df} \prec \mathsf{n} \prec \mathsf{vf} \prec \mathsf{c} \prec \mathsf{mf} \prec \mathsf{vc} \prec \mathsf{v} \tag{1.2}$$

The global total order on $\mathcal{F}$ can be decomposed into local total orders. For instance the local order $\mathsf{df} \prec \mathsf{n}$ requires determiners to precede their corresponding nouns. Conversely, the sequence $\mathsf{vf} \prec \mathsf{c} \prec \mathsf{mf} \prec \mathsf{vc}$ requires the Vorfeld (vf) to precede the left parenthesis (c) to precede the Mittelfeld (mf) to precede the verb cluster or right parenthesis (vc).

In our example, the desired linearization is attained as follows:

1. *Mann* lands in the vf, *Maria* in the mf and *geliebt* in the vc of *hat*. Since $\mathsf{vf} \prec \mathsf{mf} \prec \mathsf{vc}$ in (1.2), *Mann* must precede *Maria* and *Maria* must precede *geliebt*.

2. The internal field of *hat* is c. Because $\mathsf{vf} \prec \mathsf{c} \prec \mathsf{mf}$ in (1.2), it must be placed between the *Mann* in the vf and *Maria* in the mf.

3.3 Example lexicon

DD states well-formedness conditions for LP trees in a lexicalized fashion: a lexical entry stipulates which external fields are *offered* for topological dependents to land in and which are *accepted*. A node w' can land in an external field f of host w iff w *offers* f and w' *accepts* f. A lexical entry also assigns a set of possible internal fields to each word. Here are the lexical entries for our example[4]:

	offers	accepts	internal
einen	{}	{df}	{d}
Mann	{df}	{vf, mf}	{n}
Maria	{}	{vf, mf}	{n}
hat	{vf?, mf∗, vc?}	{}	{c}
geliebt	{}	{vc}	{v}

The set of fields offered by a node is given in *wildcard notation*: e.g. vf? indicates that there can be at most one topological dependent in the vf (as stated by topological fields theory), and mf∗ that any number of daughters can land in the mf.

[3] A node is said to *land* in external field f iff its incoming edge is labeled with f.

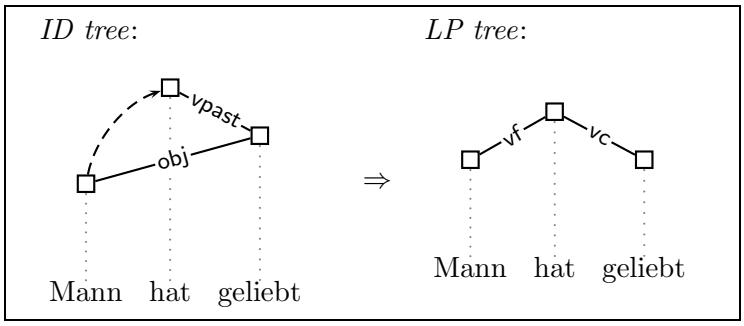
[4] We display only the LP tree part of each lexical entry. Full lexical entries also comprise ID tree information such as subcategorization and agreement.

3.4 Climbing

The well-formedness conditions for LP trees are further constrained by a *grammatical principle*[5]:

Principle 1 *a node must land on a transitive head*

which states that the host w of a node w' in the LP tree must be above w' in the ID tree. If a node lands above of its syntactic head, it is said to have *climbed*. Below, we illustrate how *Mann* climbs into the vf of *hat* (as indicated by the dashed arrow):



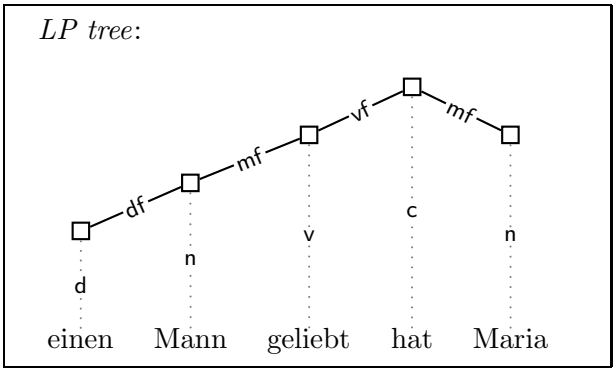
Mann is forced to climb by the well-formedness conditions stated in the lexicon: it cannot land on *geliebt* because *geliebt* offers no field.

3.5 Extending coverage

Note that the example lexicon has been designed to be as simple as possible, and is therefore not representative of DD's coverage. In fact, we have developed a German grammar for our prototype parser which also covers e.g. verb-last sentences, partial verb phrase fronting and relative clauses. In order to also handle verb-last sentences, we introduce separate lexical entries for finite verbs that have internal field v and do not offer a vf. The introduction of additional lexical entries does not really hamper the efficiency of our approach as the parser uses a novel constraint-based treatment of lexical ambiguity as introduced in e.g. (Duchier 1999).

In order to accomodate partial verb phrase fronting in the example lexicon, we would need an additional lexical entry for *geliebt*. If it lands in the Vorfeld, i.e. if it accepts {vf}, then it should offer {mf∗}. An LP tree analysis of the partial verb phrase fronting sentence *Einen Mann geliebt hat Maria* would then look like the one depicted below:

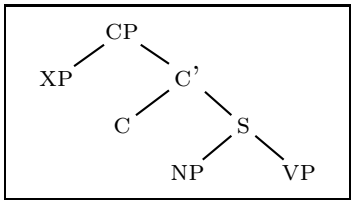
[5]We only mention the first of the three principles given in (Duchier and Debusmann 2001).



4 Verb-second clauses: a GB analysis

4.1 German sentence structure

There are two approaches to analyzing verb-second sentences in GB. We choose the approach taken in (Grewendorf 1988), as originally suggested in (Chomsky 1986). It assumes the following sentence structure for German:[6]

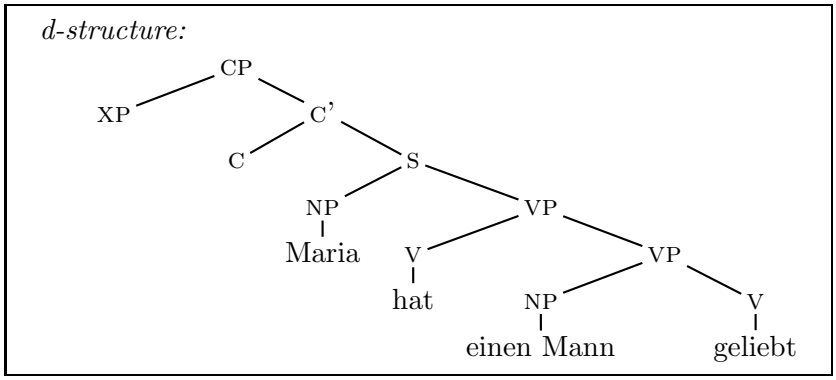


The theory of topological fields can be mapped to the proposed GB structure as follows: the Vorfeld corresponds to the [CP,XP]-position and the left parenthesis to [C',C]. The right parenthesis and the Mittelfeld have no equivalents in the above tree configuration.

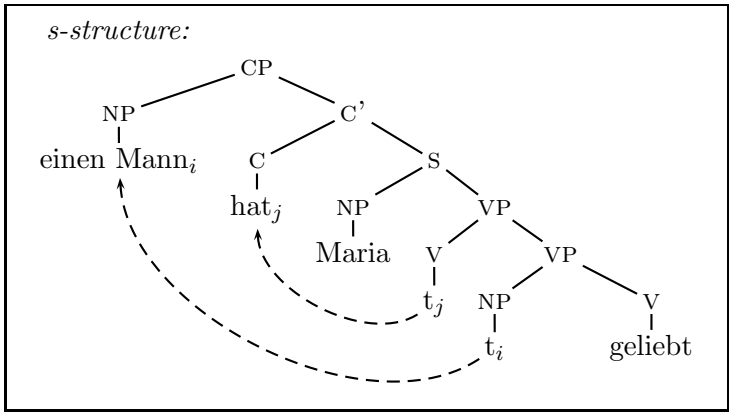
4.2 d- and s-structure

Two of the four levels of analysis that GB posits are of interest for our purposes here: d- and s-structure. Here is a d-structure analysis of (1.1):

[6]For simplicity, we do not depict the IP- and I'-nodes from the original tree shown in (Grewendorf 1988), p. 49.



GB uses an operation called move-α to mediate between d- and s-structure. Move-α moves nodes into *landing sites*, which are positions available for movement to take place to. The number of landing sites is restricted by constraints such as the θ-criterion and the Case-Filter. In the example, [CP,XP] and [C',C] function as landing sites. The XP-position [CP,XP] is a landing site only for maximal projections, whereas [C',C] is a head position and therefore only available to heads. In the s-structure shown below, move-α takes place into both landing sites: the NP *einen Mann* moves to [CP,XP] (Vorfeld), and the finite verb *hat* to [C',C] (left parenthesis):



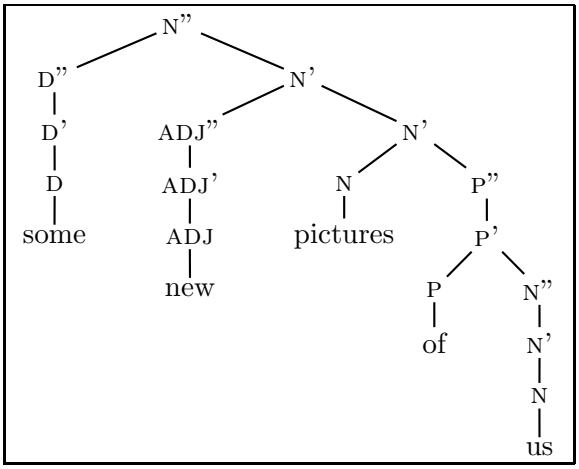
5 GB and DD: a comparison

5.1 Dependency

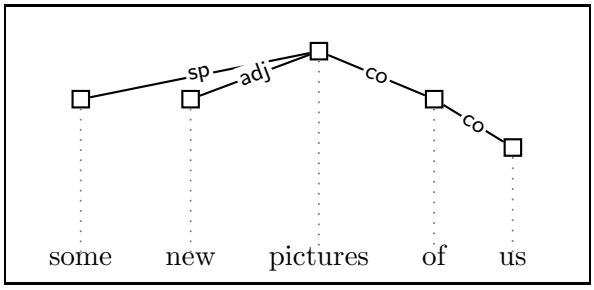
An obvious difference between GB and DD is that GB is a phrase structure-based theory and DD dependency-based. But is no crucial difference: since GB is based on X-bar theory (Jackendoff 1977), it also incorporates the notion of a *head*: X-bar theory requires that every phrase has a head which is a single word. (Covington 1990) argues that if a phrase structure analysis (1) picks out one node as the head of each phrase and (2) has no labels or features on non-terminal nodes (unless of course copied unchanged from

terminal nodes), it can be regarded as being equivalent to a dependency analysis that specifies word order.

(Covington 1992) even goes one step further by attempting to simplify GB theory by recasting it into a dependency formalism. He shows how to convert GB's X-bar-based phrase structure trees into equivalent dependency trees and then redefines government in terms of dependency. As an example, consider the GB phrase structure tree below:



The head of the phrase is *pictures*, which has a specifier (*some*), an adjunct (*new*) and a complement (*of*). The complement of *of* is *us*. Here is an equivalent dependency tree:



where *sp* stands for *specifier*, *adj* for *adjunct* and *co* for *complement*. With respect to a dependency tree, GB's notion of government is now much easier to define:

Definition 1 *A governs B iff B is an immediate dependent of A.*

By using dependency trees rather than phrase structure trees, not only principles such as government become easier to define. From a computational point of view, dependency grammar also has the advantage of postulating fewer nodes, i.e. exactly one node per word. Fewer nodes lead to improved performance because the trees to be processed are smaller.

5.2 Valency

GB and DD and in fact most linguistic theories to date share a notion of *valency*. Both GB and DD state valency requirements in the lexicon: GB uses subcategorization frames to specify the required θ-roles, and a DD lexicon includes ID tree and LP tree valency. ID tree valency is encoded by stating which syntactic roles a word *offers* and is very similar to GB's subcategorization frames. For example, a finite transitive verb offers subj and obj. On the opposite, LP tree valency specifies which fields a word offers.

5.3 Constituency

While GB includes the notion of dependency as a derived notion only, constituency is incorporated as a first class citizen. Constituents or phrases in GB are contiguous substrings of the analyzed sentence. DD includes constituency as a derived notion in both the ID and LP tree. In the ID tree, the set of nodes equal or below a head can be viewed as a constituent, but one which is not required to be contiguous (since the ID tree is unordered and non-projective). In the LP tree, the set of nodes equal or below a host forms a contiguous substring constituent (because LP trees are ordered and projective).

5.4 Movement

Both theories use a notion of *movement* to relate a deep or syntactic structure (d-structure in GB, ID tree in DD) to a surface or topological structure (s-structure, LP tree). But while in GB, move-α is a primitive operation modelled by tree transformations, climbing is a derived notion in DD: it describes the effect of well-formedness conditions. These well-formedness conditions are axiomatized in a constraint-based fashion, as outlined in (Duchier 1999) and (Duchier 2000), and can be easily turned into a parser.

GB's tree transformational account of movement severely restricts the computational usability of the theory. As (Covington 1990) argues, because transformations are tree-to-tree mappings, a parser can only undo a transformation if it has already parsed the tree structure that represents the output of the transformation. That is, the only way to parse a transformed sentence is to undo the transformation — but the only way to undo the transformation is to parse the sentence first.

GB restricts the applicability of move-α by providing a fixed set of kinds of movement, including *wh-* and NP-movement. Movement is further constrained by general principles such as the Case Filter and the θ-criterion. For instance, only $\overline{\theta}$-positions[7] may function as *landing sites* for movement

[7] A $\overline{\theta}$-position is a position which is not assigned a θ-role.

in GB. XP-positions are landing sites for maximal projections only (e.g. [CP,XP]) and head-positions for heads.

DD constraints movement in a lexicalized way. Only a small number of grammatical principles are postulated and the remaining work is done in the lexicon: a lexical entry stipulates which fields are *offered* and which are *accepted*. Making the connection to GB again, the notion of offered fields is very similar to GB's *landing sites*.

6 Conclusion

The new dependency grammar-based framework described in DD employs concepts which are strikingly similar to concepts in GB theory. Above all, both GB and DD use a notion of *movement* to mediate between levels of syntax and linear precedence. But while GB models movement as tree transformations, movement in DD is the consequence of well-formedness conditions.

We demonstrated with analyses of verb-second clauses that on a descriptive level, the notions of movement in GB and DD are yet very similar. This suggests that GB's approach to movement could be reformulated in a way similar to DD's constraint-based approach. A non-transformational account of movement based on well-formedness conditions would make GB much more attractive from a computational point of view, and could make use of techniques developed for DD, including an efficient treatment of ambiguity using finite-set constraints.

Bibliography

Bröker, N. (1998). Separating Surface Order and Syntactic Relations in a Dependency Grammar. In *COLING-ACL 98 - Proc. of the 17th Intl. Conf. on Computational Linguistics and 36th Annual Meeting of the ACL.*, Montreal/CAN.

Chomsky, N. (1986). *Barriers*. Linguistic Inquiry Monograph 13. Cambridge/MA: MIT Press.

Covington, M. A. (1990). A Dependency Parser for Variable-Word-Order Languages. Research Report AI-1990-01, Artificial Intelligence Programs, University of Georgia, Athens/GA.

Covington, M. A. (1992). GB Theory as Dependency Grammar. Research Report AI-1992-03, Artificial Intelligence Program, University of Georgia, Athens/GA.

Duchier, D. (1999). Axiomatizing Dependency Parsing Using Set Constraints. In *Sixth Meeting on the Mathematics of Language*, Orlando/FL.

Duchier, D. (2000). Configuration Of Labeled Trees Under Lexicalized Constraints And Principles. To appear.

Duchier, D. and R. Debusmann (2001). Topological Dependency Trees: A Constraint-based Account of Linear Precedence. In *39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)*, Toulouse/FRA. To appear.

Erdmann, O. (1886). *Grundzüge der deutschen Syntax nach ihrer geschichtlichen Entwicklung dargestellt*. Stuttgart/FRG: Erste Abteilung.

Gerdes, K. and S. Kahane (2001). Word Order in German: A Formal Dependency Grammar Using a Topological Hierarchy. In *39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)*, Toulouse/FRA. To appear.

Grewendorf, G. (1988). *Aspekte der deutschen Syntax. Eine Rektions-Bindungs-Analyse*. Studien zur deutschen Grammatik 33. Tübingen/FRG: Gunter Narr.

Herling, S. (1821). Über die Topik der deutschen Sprache.

Jackendoff, R. (1977). *$\bar{X}$ Syntax: A Study of Phrase Structure*. Number 2 in Linguistic Inquiry Monographs. Cambridge/MA: MIT Press.

Kahane, S., A. Nasr, and O. Rambow (1998). Pseudo-projectivity: a polynomially parsable non-projective dependency grammar. In *36th Annual Meeting of the Association for Computational Linguistics (ACL 1998)*, Montreal/CAN.

Kathol, A. (2000). *Linear Syntax*. Oxford University Press.

Müller, S. (1999). *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*. Linguistische Arbeiten 394. Tübingen/FRG: Max Niemeyer Verlag.

Reape, M. (1994). Domain Union and Word Order Variation in German. In J. Nerbonne, K. Netter, and C. Pollard (Eds.), *German in Head-Driven Phrase Structure Grammar*, pp. 151–197. Stanford/CA: CSLI.

Tesnière, L. (1959). *Eléments de Syntaxe Structurale*. Paris/FRA: Klincksiek.