

# Parsing ill-formed inputs with constraint graphs

Philippe Blache & David-Olivier Azulay

LPL, Université de Provence  
29 Avenue Robert Schuman  
13621 Aix-en-Provence, France  
`pb@lpl.univ-aix.fr`

**Abstract.** We present in this paper a constraint-based account of robust parsing. More precisely, we propose a parsing technique relying on constraint satisfaction and its implementation by means of graphs. We show how constraint graphs constitute a flexible parsing framework both in terms of representation and implementation. They allow in particular to take into account non-connected elements, frequent in particular when parsing spoken languages. This approach is illustrated with some problematic examples (hesitations, phatics, repairs, etc.) taken from spoken french corpora.

## 1 Introduction

An important problem for robustness in NLP, in particular since building treebanks is concerned (see for example [11]), comes from parsing ill-formed inputs. Several techniques have been proposed, most of them consisting in recovering parsing errors or compensating inadequate parsing techniques with heuristics (see [10], [16]). One method consists for example in modifying the grammar by introducing new rules or new categories (e.g. an anonymous one).

Rather than ad hoc techniques, we propose in this paper a general approach relying on the idea that a linguistic description should be flexible by definition. More precisely, many problems encountered in robust parsing come from the linguistic formalisms themselves which cannot generate non-canonical structures. Moreover, they rely on the idea that a syntactic representation must connect all the linguistic objects (this is the case for generative theories as well as dependency ones). We propose to give up this restriction and allow the possibility of building non connected structures. One immediate consequence is the necessity of replacing trees with graphs. Such a representation offers other advantages such as the possibility of implementing crossing relations. This can be useful, according to the chosen linguistic formalism, for the analyze of some complex constructions such as parasitic gaps or adjunct extraction (see [13] or [6]). The interpretation of such structures, as shown in the next section, relies on an explicit representation of semantic relations.

Representing syntactic structures with graphs can be done in several ways. We adopt here the most general point of view which consists in representing directly the linguistic information by means of sub-graphs. We present in the first

section of this paper an account of linguistic information in terms of constraints. We propose in particular the use of a limited set of constraints for the representation of syntax. The second section shows the equivalence between constraint sets and their representation by means of graphs. The third section describes how to use such graphs for building a syntactic representation. Finally, the last section presents this approach in the perspective of parsing spoken language inputs.

## 2 Linguistic constraints

One of the important results of modern linguistics is that all linguistic information can be conceived as constraints over linguistic objects (see for example [12], [13], [14], [3] or [8] for a description of parsing in terms of constraint satisfaction). Constraints are typically relations between two (or more) objects: it is then possible to represent information in a flat manner by means of such relations. The first step in this work consists in identifying the kind of relations usually used in syntax. This can be done empirically and we suggest, adapting a proposal from [2], the set of following constraints: *linearity*, *dependency*, *obligation*, *exclusion*, *requirement* and *uniqueness*. In a phrase-structure perspective all these constraints participate to the description of a phrase. The following figure roughly sketches their respective roles, illustrated with some examples for the *NP*.

Constraint	Definition	Example
<b>Linearity</b> ( $\prec$ )	Linear precedence constraints.	$Det \prec N$
<b>Dependency</b> ( $\rightsquigarrow$ )	Dependency relations between categories.	$AP \rightsquigarrow N$
<b>Obligation</b> ( <i>Oblig</i> )	Set of compulsory and unique categories. One of these categories (and only one) has to be realized in a phrase.	$Oblig(NP) = \{N, Pro\}$
<b>Exclusion</b> ( $\nexists$ )	Restriction of cooccurrence between sets of categories.	$N[pro] \nexists Det$
<b>Requirement</b> ( $\Rightarrow$ )	Mandatory cooccurrence between sets of categories.	$N[com] \Rightarrow Det$
<b>Uniqueness</b> ( <i>Uniq</i> )	Set of categories which cannot be repeated in a phrase.	$Uniq(NP) = \{Det, N, AP, PP, Pro\}$

In this approach, describing a phrase consists in specifying a set of constraints over some categories that can constitute it. These constraints are specified in the same manner as GPSG (cf. [9]) first introduced the linear precedence statement. This information was stipulated in terms of *possible* realization: a LP constraint of the form  $a \prec b$  indicates that  $b$  cannot precede  $a$ . This constraint is relevant only when these two categories cooccur into a phrase. In the present approach, a constraint is specified as follows. Let  $\mathcal{R}$  a symbol representing a constraint relation between two (sets of) categories. A constraint of the form  $a\mathcal{R}b$  stipulates that if  $a$  and  $b$  are realized, then the constraint  $a\mathcal{R}b$  must be satisfied. The set of constraints describing a phrase can be represented as a graph connecting several categories. The following example illustrates some constraints for the *NP*.

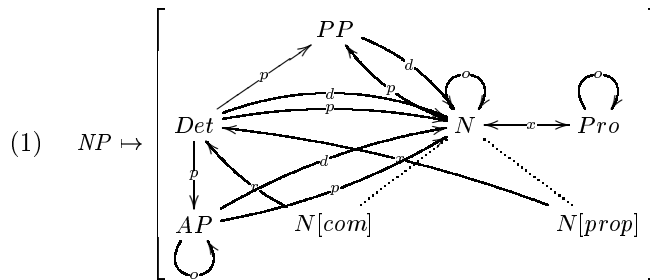
- *Linearity*:  $Det \prec N; Det \prec AP; AP \prec N; N \prec PP$
- *Requirement*:  $N[com] \Rightarrow Det$
- *Exclusion*:  $N \not\Leftarrow Pro; N[prop] \not\Leftarrow Det$
- *Dependency*:  $Det \rightsquigarrow N; AP \rightsquigarrow N; PP \rightsquigarrow N$
- *Obligation*:  $Oblig(NP) = \{N, Pro, AP\}$

This toy description, in addition to the classical linear constraints, proposes a requirement relation between the common noun and the determiner. Such a constraint implements the complementation relation. The opposite relation, exclusion, indicates cooccurrence restriction. This kind of constraint allows together with requirement a precise description of what categories can constitute the described phrase. In our example, constraints indicate an impossible realization of a noun with a pronoun or a proper noun with a determiner. One can notice the use of sub-typing: as it is usually the case in linguistic theories, a category has several properties that can be inherited when the description of the category is refined (in our example, the type *noun* has two sub-types, *proper* and *common* represented in feature based notation). All constraints involving a noun also hold for its sub-types.

The dependency relation is a semantic one. It indicates that the dependent must combine its semantic features with the governor. In the same way as HPSG does now with the DEPS feature (cf. [6]), this relation concerns any category, not necessarily the governed ones. In this way, the difference between a complement and an adjunct is that only the complement is selected by a requirement constraint, both of them being constrained with a dependency relation. This also means that a difference can be done between the syntactic head (indicated by the *oblig* constraint and the semantic one (the governor of the dependency relation), even if in most of the cases, these categories are the same. Moreover, one can imagine the specification of dependencies within a phrase between two categories other than the head. More generally, it is always possible to specify relations directly between any categories of the phrase.

### 3 Constraint graphs

A constraint graph is a set of constraints describing a phrase. Each constraint is represented by a set of nodes (categories) and an edge. A constraint graph is not necessarily connected. The set of constraints for the *NP* presented in the previous section can be represented by the following constraint graph:



In this graph, each edge has a label indicating the kind of constraint ( $d$  stands for dependency,  $x$  for exclusion,  $p$  for precedence,  $o$  for oblig,  $r$  for requirement<sup>1</sup>). For example, the edge  $\langle Det \leftarrow_p AP \rangle$  represents the linear precedence constraint  $Det \prec AP$ . The obligation constraint (as the uniqueness one, not represented here) is particular in the sense that it stipulates a constraint over one category. It is then represented as a reflexive relation. The sub-typing relation between the noun and its subcategories is indicated with a dotted line which does not correspond to a constraint edge. This schema illustrates the fact that relations can be specified indifferently over a category or its specifications which inherit the constraints of the root.

A constraint graph, as represented in the example (1), has a label corresponding to the phrasal category it describes. A specific relation, in the HPSG’s head feature principle way, exists between the head of the graph (indicated by the obligation relation) and this label (its root).

A grammar is thus formed by a set of constraint graphs, each one corresponding to the description of a phrase. The basic idea consists then, given a set of categories, to verify its satisfiability according to the constraints. Let us call such a set, using a constraint programming terminology, an *assignment*. Each constraint of the grammar, independently from its membership to a constraint graph, is verified for an assignment. In this case, constraints are activated when the constrained variables correspond to some elements of the assignment.

When a constraint graph  $\mathcal{G}$  is activated (i.e. the assignment contains categories constrained by  $\mathcal{G}$ ), its label is instantiated and the corresponding category is considered as realized. If an assignment satisfies all the constraints of a constraint graph, the corresponding category is well-formed. But each assignment, whatever its form, can receive a characterization constituted by the state of the constraint graph after evaluation (formed by satisfied and non satisfied constraints). This means that it is possible to give a characterization of any object, even ill-formed.

One can notice an interesting characteristics: no constituency information belongs to the set of constraints. This constitutes an important difference with other phrase-structure approaches in which such information as to be encoded more or less implicitly. This comes from the fact that these methods define grammaticality in terms of constraints over a hierarchical structure. It is then necessary to build such a structure before verifying its properties. Insofar as building the structure is done following rules, only “well-formed” structures can be built. It is even the case in HPSG in which all the elements of a phrase has to be selected by a head or its complements.

---

<sup>1</sup> In this approach, constraints can be stipulated over sets of categories. We propose to represent this information by introducing a virtual node noted  $\bowtie$ . The set of categories is represented by the conjunction of nodes at the origin of a relation  $\bowtie$ .

The constraint  $\{a, b\} \mathcal{R} c$  is then represented by the graph :

$$\begin{array}{c}
 a \text{ ---} \\
 b \text{ ---} \\
 \quad \quad \quad \searrow \quad \swarrow \\
 \quad \quad \quad \quad \quad \quad \bowtie \text{ ---} \mathcal{R} \rightarrow c
 \end{array}$$

In the approach described here, a phrase is instantiated only after constraint verification. Such a verification relies on relations between categories, as described above, and doesn't necessitate any hierarchical information: the projection notion doesn't play a role in this process and the government information is a relation among others. Moreover, we can notice at this point that an assignment can contain more categories than that actually constrained.

## 4 Description Graphs

In the same way as constraints in HPSG play the role of feature structure descriptions, constraint graphs allow to build the graph representing the syntactic structure of a given input. We call this graph a *description graph*. The mechanism consists, starting from the set of lexical categories corresponding to the words of the input, in building the subsets of categories (corresponding to assignments) that can be characterized by a constraint graph. When a subset is characterized, the corresponding phrase-level category is instantiated and completes the initial set of categories. New assignments can then be built, the process ends when no assignment can be characterized.

Characterizing an assignment  $\mathcal{A}$  comes to evaluate the constraint system (i.e. the entire grammar) for  $\mathcal{A}$ . At the end of this process, it is possible to identify a subset of constraint graphs relevant<sup>2</sup> for  $\mathcal{A}$ . The constraint system after evaluation is formed by three different kind of constraints: *satisfied*, *violated* and *unspecified* constraints. This last state concerns constraints which variables don't belong to  $\mathcal{A}$ . A description graph is then formed with a constraint graph specifying these different states for each constraint.

The following example illustrates this process. The input here is formed with a determiner and a noun, the considered assignment is  $\mathcal{A} = \{Det, N[com]\}$ . As described in the previous section, the satisfiability of  $\mathcal{A}$  is calculated over the entire constraint system (the grammar). In this example, we can suppose that all activated constraints belong to the *NP* constraint graph described in (1). More precisely, the following constraints can be evaluated for  $\mathcal{A}$ :

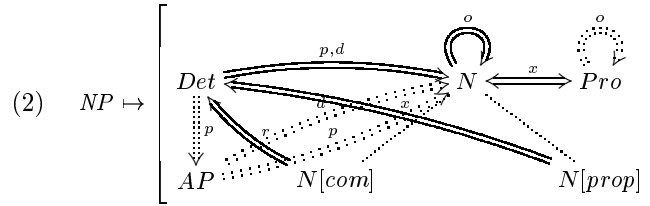
- |   |  |
|---|--|
| <p>(1) <math>Det \prec N</math><br/> (2) <math>N[com] \Rightarrow Det</math><br/> (3) <math>N \not\Leftarrow Pro</math></p> | <p>(4) <math>N[prop] \not\Leftarrow Det</math><br/> (5) <math>Det \rightsquigarrow N</math><br/> (6) <math>Oblig(NP) = \{N, Pro\}</math></p> |
|---|--|

All these constraint are satisfied, which means that  $\mathcal{A}$  is well-formed. The category corresponding to the constraint graph label (here *NP*) can then be instantiated and a new edge (labeled with *NP*) covering the categories of the

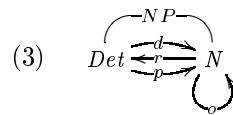
---

<sup>2</sup> Note that a classical constituency-driven process would consist in first identifying the relevant constraint graphs and then evaluating them.

assignment is created. The description graph built from (1) has the following form<sup>3</sup>:

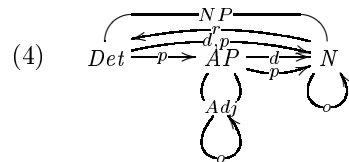


In the remaining, for clarity, we only indicate in the description graphs the satisfied and violated constraints for realized categories. Moreover, the category instantiated by the description graph is specified as a label of an edge covering the assignment. The description graph in (2) is then represented as follows:



The instantiated category can be used to its turn into another assignment. We can notice that two different edge types coexist in this graph: edges representing constraints (which are labeled with the constraint name) and edges representing the coverage of a phrase. All categories immediately dominated by this edge are considered as constituents of the phrase.

Let's complete the previous example with an adjective phrase composed with a single adjective realized between the determiner and the noun. The new assignment is  $\mathcal{A} = \{Det, AP, N[com]\}$ . Following the same mechanism, the resulting description graph is of the form:



In this graph, a description graph has been built for the  $AP$  (a single relevant constraint is satisfied, the obligation one, no other constraint is violated). This category has then been instantiated and can be used, as any other category, for the rest of the process. The description graph of  $AP$  is thus integrated to the general description graph of the  $NP$ . What is interesting in this approach is that a relation can be expressed between any categories (for example directly between the adjective and the determiner), not necessarily into a government framework.

<sup>3</sup> In some graphs, for clarity, we can indicate as label of the same arrow several constraint types. This notation factorizes different constraints provided that they have the same source and target.

This means that, contrarily to most of other approaches, a relation between two categories does not need a specific role for the head. Such a phenomenon can be for example illustrated by the anaphoric reference between a pronoun and a noun phrase (typically in dislocated constructions): in this case, one can stipulate a relation between the anaphoric clitic and the determiner requiring the determiner to be definite.

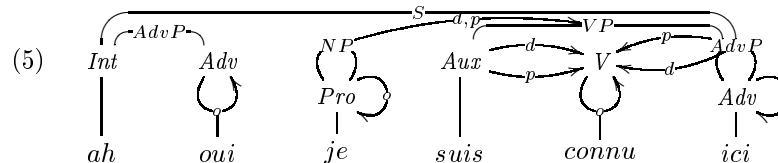
## 5 Parsing ill-formed inputs

This approach is flexible in the sense that nothing is said more than the information explicitly stipulated by constraints. This is an open conception of syntactic information: what is explicit (the constraints) has to be verified, the rest simply cohabits with the structure. The point is that constraint verification doesn't mean imperative satisfiability (in other words, constraints are relaxed). As explained above, the characterization of an assignment can contain satisfied or non-satisfied constraints. Both are important for analyzing the properties of the corresponding phrase. In a robust parsing perspective, this means that it is possible to characterize incomplete or ill-formed phrases. Moreover, and perhaps more importantly, an assignment can contain unconstrained categories. Such categories don't have any relation with other categories without any consequence over the characterization itself. Such phenomenon occurs in particular in spoken languages with so-called *associated* elements (see [5]).

This property is possible because constituency is no more considered as a constraint. The set of categories belonging to a constraint graph only indicates some possible constituents, not necessarily the complete set of them. The following example illustrated this property.

- (a) ah oui je suis connu ici  
*oh yes I am known here*

This input, taken from a spoken French corpus<sup>4</sup>, contains an interjection *ah* followed by an adverb *oui*. The problem consists in finding a status for the interjection and the adverb which is not in this case a sentence modifier.



We consider here that, in spite of the fact that no direct relation exists between the interjection and the adverb, these two categories can form an assignment characterizing an *AdvP*. More precisely, no constraint is violated by the assignment  $\mathcal{A} = \{Int, Adv\}$  when verifying the *AdvP* constraint graph. It is then a possibility to consider *Int* as a possible element of *AdvP*. In the same way,

<sup>4</sup> Corpus "Bouliste", GARS, Université de Provence.

the *AdvP* is not directly connected to the rest of the input. This representation shows that associated elements are part of the input (they are covered by an edge *S*) without having exact relation with the rest of the sentence.

The second example<sup>5</sup> illustrates the capacity of integrating other kind of non-connected elements such as phatic or interpolated clauses.

- (b) la bijoutière a pris sa retraite et donc c'était une amie d'enfance de ma belle-mère d'ailleurs et donc euh elle a posé elle la question à ma belle mère  
*the jeweler has taken her retirement and then it was a friend of childhood of my mother-in-law moreover and then uh she has asked the question to my mother-in-law*

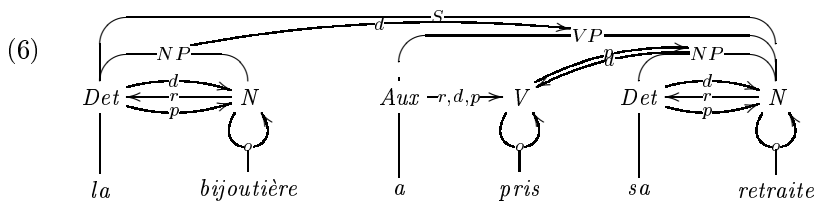
We isolate, for clarity, three sub-components of the input:

*S1: la bijoutière a pris sa retraite*

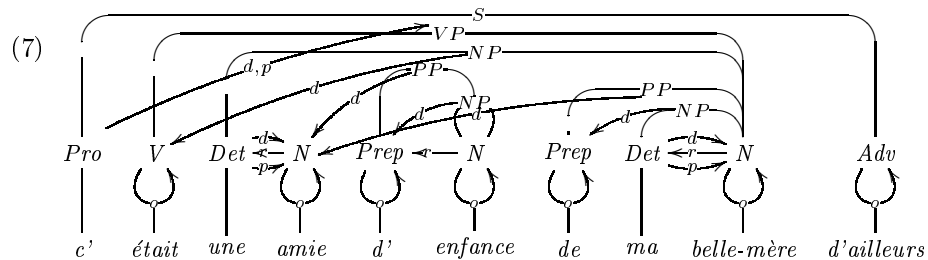
*S2: c'était une amie d'enfance de ma belle-mère d'ailleurs*

*S3: elle a posé elle la question à ma belle mère*

The first component *S1* doesn't present any difficulty. Its description graph is presented in (6). This graph is connected: all its categories have some relation with the others. In this analysis, the auxiliary depends from the main verb. This analysis could be changed easily: if we consider that the dependency direction is the opposite (from the verb to the auxiliary) the edge direction simply has to be reversed.



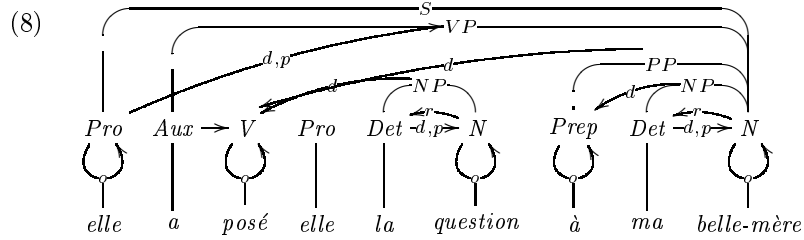
The second component *S2* can be described with the description graph (7). Note that, for readability, some relations are not mentioned. The final adverb can be considered, as for the previous example, as an associated element, its role being more enunciative than syntactic. This category thus belongs to the *S* edge, but is not connected with any other category.



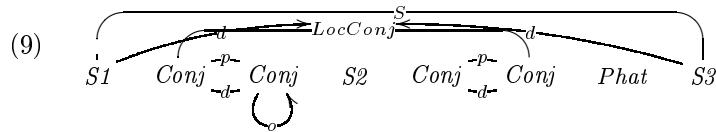
The last subcomponent *S3* contains one problematic element: the repetition of the clitic *elle* in an unexpected position. This phenomenon can be interpreted

<sup>5</sup> Corpus "La bijoutière", GARS, Université de Provence.

as an hesitation or an insistence construction. In all cases, it cannot be analyzed as having the same kind of relation with the structure as the first clitic. We think then preferable to consider it as a non-connected element of the structure.



The characterization of the entire input can be done in the same manner. It contains several elements that cannot be directly connected to the structure: the phatic *eah* and the interpolated clause *S2* (a repetition of the conjunctive locution together with some specific prosodic events not described here reinforce the interpolation interpretation). The description graph of the input can be presented as follows:



In this analysis, we consider that one of the conjunction is the head, the repetition doesn't play a specific role here. The conjunctive locution is formed by the two repetitions, each conjunct (*S1* and *S3*) has a dependency relation with it (the locution plays the role of semantic head). In this analysis, the interpolated, the repetition and the phatic belong to the structure, but don't have any specific relation with other categories.

## 6 Conclusion

This technique has been tested over a written text corpus<sup>6</sup> of 80,000 words annotated with disambiguated POS tags. Two different systems were under evaluation: a shallow parser and a deep one. The result in both cases is a graph (or a set of graphs in the case of the deep parsing technique). The first system takes about 40 seconds to parse the entire corpus, the second about 3 minutes. It generates around 60 edges per sentence.

The approach described in this paper proposes a flexible account of ill-formed inputs and more generally for robust parsing. Unexpected elements, hesitations, phatics, repetitions, interpolated, etc. can easily be integrated to the syntactic description of an input. This makes this method well adapted for parsing spoken languages.

<sup>6</sup> A corrected version of the corpus CLIF built by Talana, see <http://www.talana.linguist.jussieu.fr>.

One of the interesting consequences of such a fully constraint-based account of syntax is that it calls into question the classical notion of ill-formedness (and then well-formedness). In this approach, any kind of input can be characterized. Some of them can satisfy all the constraints, some other not. In the end, the interesting point when parsing a language (especially in the perspective of human-machine communication) is to give a description (i.e. analyze) of an input (whatever its form) more than calculating whether or not it belongs to a language. We have shown in this paper that a constraint-based approach (withdrawing the constituency notion) constitutes an efficient parsing technique.

## References

1. Abney S. (1996) "Partial Parsing via Finite-State Calculus", in proceedings of *ESSLLI'96 Robust Parsing Workshop*.
2. Bès G. & P. Blache (1999) "Propriétés et analyse d'un langage", in proceedings of *TALN'99*.
3. Blache P. (2000) "Constraints, Linguistic Theories and Natural Language Processing", in *Natural Language Processing*, D. Christodoulakis (ed.), Lecture Notes in Artificial Intelligence, Springer.
4. Blache P. & J.-M. Balfourier (2001) "Property Grammars: a Flexible Constraint-Based Approach to Parsing", in proceedings of *IWPT-2001*.
5. Blanche-Benveniste C. (1997) *Approches de la langue parlée en français*, Ophrys.
6. Bouma G., R. Malouf & I. Sag (2001) "Satisfying Constraints on Extraction and Adjunction", in *Natural Language and Linguistic Theory*, 19:1, Kluwer.
7. Chanod J.-P. (2000) "Robust Parsing and Beyond", in *Robustness in Language Technology*, Kluwer.
8. Duchier D. & R. Debusmann (2001) "Topological Dependency Trees: A Constraint-Based Account of Linear Precedence", in proceedings of *ACL*.
9. Gazdar G., E. Klein, G. Pullum, I. Sag (1985), *Generalized Phrase Structure Grammar*, Blackwell.
10. Grinberg D., J. Lafferty & D. Sleator (1995), A robust parsing algorithm for link grammars, *CMU-CS-95-125*, Carnegie Mellon University.
11. Kübler S. & E. Hinrichs (2001) "From Chunks to Function-Argument Structure: A similarity-Based Approach", in proceedings of *ACL*.
12. Maruyama H. (1990), "Structural Disambiguation with Constraint Propagation", in proceedings of *ACL'90*.
13. Pollard C. & I. Sag (1994), *Head-driven Phrase Structure Grammars*, CSLI, Chicago University Press.
14. Sag I. & T. Wasow (1999), *Syntactic Theory. A Formal Introduction*, CSLI.
15. Tesnière L. (1959) *Éléments de syntaxe structurale*, Klincksieck.
16. van Noord G., G. Bouma, R. Koeling & M.-J. Nederhof (1998), in *Natural Language Engineering*, 4:1.