A polynomial parser for the topological model
Illustration by a topological grammar for German

Abstract

This paper describes a parser for German, which treats major phenomena of word order including scrambling, (partial) VP fronting and extraposition. The outputs of the parser are dependency trees and topological phrase structures. We use the CKY parsing algorithm, which is polynomial with some bounds on the number of emancipations (that is, constituents that are positioned outside of the domain of their governor). Our approach will show the procedural role of tools such as the slash feature of HPSG.

1 Introduction

The aim of this article is to describe a parsing algorithm for topological grammars.

For a simple description of word order phenomena of so-called "free word order languages" the topological model postulates the existence of template-like structures, i.e. a series of places, called *fields*. For German, this tradition dates back to the 19th (Herling 1821) and mid-20th century (Drach 1937 for the "field" metaphor and Bech 1955 for verbal topology).

The topological model has been implemented in HPSG (Kathol 1995) and DGs (Bröker 1998, Duchier & Debusman 2001, Gerdes & Kahane 2001). The DG-based topological grammars build two types of structures: The syntactic dependency trees (i.e. unordered trees whose nodes are labeled with the words of the sentence, and whose branches are labeled with syntactic relations among the words (*subject, direct object.…*)) and a topological structure.

In Duchier & Debusman 2001, this topological structure is a projective dependency tree; Gerdes & Kahane 2001 propose a topological phrase structure, i.e. a hierarchy of *topological phrases* that are defined as fixed series of fields.

Contrarily to most phrase structure grammars (based on X-bar principles), a topological phrase structure does not attempt to express subcategorization directly. (It is only linked by powerful grammar rules to the corresponding dependency trees with the information on subcategorization.) For instance, in the topological grammar for German, a non-finite verb can head two kinds of topological phrases, either a phrase, called *domain*, with positions for all of its dependents, or a restricted phrase, which forms the verb cluster, with no positions for dependents other than predicative elements. These two kinds of phrases must be placed in very different topological positions: A domain is placed in any *major field* (Vorfeld, Mittelfeld, Nachfeld) and the verb cluster only in a field at the end of the proposition called the *right bracket*.

The algorithm we present is based on the topological phrase structure and follows the traditional CKY algorithm for context-free grammar. The mismatches between the topological structure and the dependency tree force us to store some information in a way which is similar to the *slash feature* of G/HPSG (Gazdar *et al.* 1985). This brings to the fore the procedural role of the slash feature. Moreover, we combine the slash feature with another feature we call the *visitor feature*, which plays a complementary role.

In Section 2 we briefly recall some word order phenomena of German covered by the topological grammar which cause difficulties for parsing. Section 3 presents the grammar formalism and Section 4 the parsing algorithm we propose.

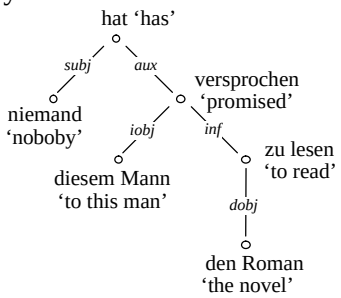
2 Some word order phenomena of German

The topological grammar allows analyzing uniformly important free word order phenomena of German such as *(partial) VP-fronting*, *intraposition*, *extraposition*, *auxiliary flip* and *pied-piping*. In the following, we present only the main phenomena (see Kathol 1995 and Gerdes & Kahane 2001 for a larger grammar).

German is a V2 language, that is, the main finite verb occupies always the second position of the declarative sentence. Moreover, all the other verbs of the proposition tend to form a verb cluster at the end of the proposition, cutting the sentence in five areas: *Vorfeld, left bracket, Mittelfeld, right bracket, Nachfeld*. The left bracket is filled by the V2 and the right bracket by the verb cluster. The three other fields, the *major fields*, accommodate the other constituents, whose relative order is rather free.[1] The fact that the complements of different verbs can be placed in contradiction to an X-bar phrase structure is called *scrambling* (1b). Some complications arise because a subordinated verb, rather than going in the verb cluster, can open an *embedded domain* in one of the major fields. This phenomenon is covered by the terms of *VP-fronting* (1c), *intraposition* (1e), and *extraposition* (1d). Moreover, a dependent of a verb in an embedded domain can *emancipate* and go in a major field of a higher domain (1f).

(1) **a.** *Niemand hat diesem Mann den Roman zu lesen versprochen*
Nobody (nom.) has this man (dat.) the novel (acc.) to read promised
b. *Diesem Mann hat den Roman niemand zu lesen versprochen*
c. *Den Roman zu lesen hat diesem Mann niemand versprochen*
d. *Diesem Mann hat niemand versprochen, den Roman zu lesen*
e. *Diesem Mann hat, den Roman zu lesen, niemand versprochen*
f. *Zu lesen hat diesem Mann den Roman niemand versprochen*
g. *Den Roman hat niemand diesem Mann versprochen zu lesen*
'Nobody promised this man to read the novel.'

All of these sentences correspond to the same dependency tree:



In order to describe these phenomena, Gerdes & Kahane 2001 consider that each word opens one (or several) *topological phrases* in which can go its dependents and eventually some words emancipated from embedded phrases. A topological phrase resembles a *box*, whose ordered compartments, called *fields*, can themselves accommodate new boxes. In addition to the rules that list the fields of each type of box, the topological grammar has two further types of rules:

- rules that indicate into which field a word can go, depending on the position of its governor;
- rules that indicate which type of box a word can create when it is placed into a given field.

The hierarchy of boxes forms the *topological structure*. The word that opens a box is called the *topological head* of this box; the topological heads of the other phrases in the box are called its *topological dependents*. A node that is not positioned in a constituent opened by its governor is said to be *emancipated*: Its topological governor is then different from its syntactic governor (see Kahane *et al.* 1998 and Duchier & Debusmann 2001 for similar notions).

We can now describe German word order in the following terms.

- The main finite verb of the sentence is placed at first (in the initial field).[2] It opens the *main*

[1] The placement depends on the communicative (or information) structure of the sentence. In particular, placement in the Nachfeld underlies communicative and heaviness constraints. Direct objects are the most rare to find in the Nachfeld.

[2] We consider that in a compound verb form such as *hat gelesen* 'has read' the past participle depends syntactically on the auxiliary, which is the finite verb form. The V2 is thus always the root of the syntactic dependency tree and the head of the main domain.

domain, which is the underlying pattern of a declarative sentence, and consists of the following sequence of five fields: [Vorfeld, left bracket, Mittelfeld, right bracket, Nachfeld]. It occupies the second position of the main domain, the *left bracket*.

- A non-finite verb depending on V2 can go into the *right bracket*. As a result, it opens a reduced constituent, a *verb cluster*, with only one position on its left for a verbal dependent.[3] If a subsequent third verb joins the verbs already in the right bracket, it will again open a phrase with a position to its left, and so on.
- Dependents (verbal or non-verbal) of any of the verbs of the main domain (V2, any verb in the right bracket or even an embedded verb) can go in any of the three major fields, leaving aside some constraints we will not discuss here. Exactly one element has to occupy the Vorfeld, which is the first field.
- When a verb is placed in any of the major fields, it opens an *embedded domain*, which consists of three fields: *Mittelfeld*, *right bracket*, *Nachfeld*. It occupies the right bracket where the same rules as in the main domain apply. A dependent of a verb in an embedded domain can go in a major field of a higher domain if the embedded domain allows this emancipation.

3 The Formalism

We extend the topological linearization formalism for dependency trees of Gerdes & Kahane 2001 to account for analysis and to include a syntactic module.

A grammar in this formalism is called a *Topological Dependency Grammar*. It has two modules: a *syntactic module*, which controls the well-formedness of the syntactic representation by the projection of the lexicon, and a *topological module*, which ensures the correspondence between the syntactic structure and the topological structure.

[3] The situation is different for verbs that allow auxiliary flip (Oberfeldumstellung). See Gerdes & Kahane 2001 for details.

3.1 The syntactic module

The parameters to instantiate are the vocabulary V, the set of (lexical) categories C, the set of syntactic roles R, and the set of lexical rules. A lexical rule assigns a category and a valency bag (multiset) to a word. An element of the valency is a couple (r,A) where r is a syntactic role and A a category.[4]

Example

V = the German words

C = { V, Vfin, Vzu, Vinf, Vpp, N, D}
(V = verb, Vfin = finite verb, Vzu = infinitive with *zu*, Vinf = base infinitive, Vpp = past participle, N = noun, D = determiner)[5]

R = { subj, dobj, iobj, aux, inf, det }

Lexical rules :

hat 'has': Vfin, val:{(subj,Nnom),(aux,Vpp)}

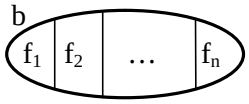
gelesen 'read': Vpp, val:{(dobj,Nacc)}

3.2 The topological module

The parameters to instantiate are the vocabulary V, the set of (lexical) categories C, the set of syntactic relations R, the set of box names B, the set of field names F, the initial field i, the order of permeability of the boxes, which is a partial ordering on B (used for emancipation) and four sets of rules:

1. Box description rules:

The rule $b \rightarrow f_1\ f_2 \ldots f_n$ indicates that the box b consists of the list of fields $f_1, f_2, \ldots, f_n$.

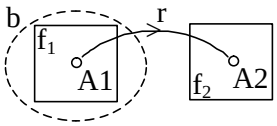


[4] We do not present the treatment of modifiers, when the governor is selected by the dependent. It does not pose any technical problem but it necessitates particular rules we cannot present here (for the treatment of modifier in a DG see for example Nasr 1995; various propositions in HPSG can also be adapted here). We do not explicit either the optionality of a syntactic argument.

[5] For the clarity of our exposure we give a very rough presentation of the category. For the nouns, cases are added in their names (nom, gen, dat, and acc).

2. Field description rules:
The pair (f,ε) in F×{!,?,+,∗} indicates that the field f has to contain exactly one element (!), at most one element (?), at least one element (+) or any number of elements (∗). This is the *satiation value* of the field. A box is *saturated* iff all fields of type (f,!) contain exactly one constituent and all fields of type (f,+) contain at least one constituent.

3. Correspondence rules (between the dependency and the topological structure):
The rule (r,A1,A2,f_2,b) indicates that a word w2 of category A2, that exhibits a dependency of type r on a word w1 of category A1, can go into field f_2 of a box containing w1, if this box is separated from w1 by borders of type = b (in other words, the parameter b controls the emancipation).



(In our figures, boxes are represented by ovals, fields by rectangles or sections of an oval.)

4. Box creation rules:
The rule (A,f,b,f') indicates that a word of category A, placed into a field f, can create a box b and go into the field f' of this box. The word that creates a box is called its *topological governor*.

Box creation rules are applied recursively until a lexical rule of type (c,f,b,•) is encountered where b is a lexical box with a unique lexical field, into which the word has to be placed.

Phrase structure derivation starting from a dependency tree

The word labeling the root node of the syntactic dependency tree is placed into the initial field i. Box creation rules are then activated until the word is placed in a lexical field (•). A correspondence rule is activated for one of the dependents of the root node, placing it in an accessible field. Just as for the root node, box creation rules are activated until the word is assigned to a lexical field. This procedure continues until the whole tree is used up. Each time a box creation rule is triggered, a box is created and a description rule for this box has to be activated. Finally, all boxes have to be saturated (e.g. a field requiring at least one element cannot remain empty).

3.3 Example of a grammar

We will now instantiate our formalism for the German grammar fragment we have informally presented in Section 2.

Parameters

B = { md, ed, vc, v, xp }
(md = main domain, ed = embedded domain, vc = verbal cluster, v = verb, xp = non-verbal phrase)

F = { i, vf, lb, mf, rb, nf, cf, hf, of, • }
(i = initial field, vf = Vorfeld, lb = left bracket, mf = Mittelfeld, rb = right bracket, nf = Nachfeld, of = Oberfeld, hf = head field, • = lexical field, f = vf/mf/nf = major field)

i is the initial field

Permeability order

0 < vc < xp = ed < md

Box description

md	->	vf lb mf rb nf
ed	->	mf rb nf
vc	->	of hf
v	->	•
xp	->	undescribed

Field description

(i,!), (vf,!), (mf,∗), (nf,∗), (lb,!), (rb,?), (of,?), (hf,!), (•,!).

Correspondence rules[6]

Positioning of the first verb in the right bracket:[7]
(-, -, V, rb, 0)

Positioning of a verb to the left of the preceding verb in the right bracket:
(-, V, V¬fin, of, 0)

[6] Some parameters can be left underspecified. In particular the syntactic role is not instantiated. Contrary to English, in German, the placement does not really depend on the syntactic role of the dependent.

[7] The last parameter (•) indicates that the right bracket of a given domain is not accessible when emancipating an element from an embedded domain.

Positioning of an element in a major field:[8]
(-, V, -, f, ed)

Box creation rules

Creation of the main domain in the initial field: (Vfin, i, md, lb)

Creation of an embedded domain in a major field: (V¬fin, f, ed, rb)

Creation of a verbal cluster in the right bracket or in the Oberfeld: (V,rb/of,vc, hf)

Positioning of a verb: (V, lb/hf, v, •)

Creation of a non-verbal phrase: (X, f, xp, ?)

4. The parsing algorithm

The parsing algorithm is driven by the topological structure, which is built bottom-up: A topological phrase is placed (in the field of a higher phrase) only when it is saturated.

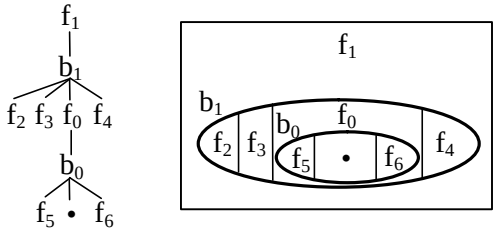
We begin with a presentation of the algorithm when there is no emancipation. In this case, we can ensure that the dependency structure is also built bottom-up, that is, a node combines with its governor only when it has all its dependents. This condition can no longer hold when emancipations are allowed.

4.1 The algorithm without emancipation

The philosophy of a CKY algorithm is to be-gin to parse one word segments of the sentence, to note the results in a parse matrix, and to parse bigger and bigger segments by concatenation of segments previously parsed. The entries of our parse matrix are of the form [i,j,hd:A,val:X,top:C] where i and j delimit the segment, A is the category of the governor, X is its valency, and C is a topological configuration.

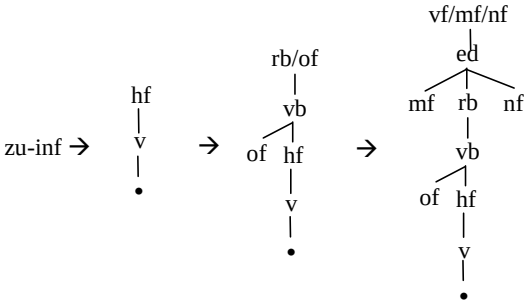
A *topological configuration* (TC) is an ordered tree of fishbone type whose internal nodes are fields or boxes, where daughters of a field are boxes, where daughters of a box are fields, where the root is a field, and where leaves are labeled by a field. Each field label is accompanied by a satiation value (?, !, *, or +). The fishbone type implies that a TC has exactly one spine, which terminates in a lexical field • corresponding to the position the head of the segment, all other branches being of length 1.

Example of a topological configuration in tree (left) and box representation (right):



The first step of the algorithm is to associate a category, a valency and a TC to each one word segment [i,i] of the sentence. These initial TCs are built by using the box creation rules, the box description rules and the field description rules. It consists more or less to lexicalize this part of the grammar.

Given a word a of category A, for each lexical rule ($A,f_0,b_0,\bullet$), we consider the TC f_0-b_0-•. Then we can trigger box creation rules (A,f_1,b_1,f_0), a box description rule for b_1 and field description rules for each field of b_1, giving us new TCs. More complex configurations can be obtained by triggering rules (A,f_2,b_2,f_1) and needed box and field description rules. Our German grammar verifies that this process necessarily stops at this step and that no rule (A,f_3,b_3,f_2) exists and can apply.[9] The following figure shows the TCs associated with a *zu*-infinitive:



[8] The last parameter indicates that it is possible to emancipate out of any type of box inferior to 'ed' in the order of permeability, i.e. ed, xp, or vc, but in German, emancipation will not in general be possible out of phrasal complements.

[9] A similar assumption is made in X-bar Syntax, where an item can only be the head of a limited number of phrases (X, X', and X" = XP in the most common version). In our framework, this number is limited but not fixed: A word can head two or three constituents according to its place. For instance, a non-finite verb that opens a new domain heads three constituents (a lexical box, a verb cluster, and an embedded domain), whereas it heads only two constituents when it joins in the right bracket of its governor's domain.

Note that these TCs can be compiled off-lined, creating lexicalized tree chunks, similar to elementary trees of Tree Adjoining Grammars. During the analysis, each TC, not only maximal TCs, has to be considered separately, which causes the maximum number of TCs per word to show up in the complexity of the algorithm.

Let us examine the recursivity step of our algorithm. Suppose that we have the parsed segments [i,j,hd:A1,val:X1,top:C1] and [j+1,k,hd:A2,val:X2,top:C2]. The two segments can be combined if one of them fill requirements of the other. Suppose that A1 will be the head. Then A2 must fill one of the valency slots of A1 and C2 must fill one of the fields of C1. Moreover we impose that the valency and the TC of A2 are saturated, that is, X2=∅ and no field of C2 needs to be filled (none shows a satiation value equal to ! or +). The leaves of C1 between • and the place filled by C2 must also be saturated. Finally the combination of A1 and A2 must be licensed by a correspondence rule (r,A1,A2,f2,b), where r is the syntactic role assigned to A2 and f2 is both the root of C2 and the field of C1 will be filled by C2 (the variable b, which controls emancipation, is not considered here). This give us the new parsed segment [i,k,hd:A1,val:X,top:C], where X is X1 deprived of (r,A2) and C is a copy of C1 where

1) the field filled by C2 is suppressed if it had a satiation value equal to ? or ! and it receives the value * elsewhere and

2) the leaves between the spine and the place filled by C2 are suppressed (they cannot be filled afterwards).

The parsing succeeds if the whole sentence corresponds to a segment [1,n,hd:A,val:∅, top:C], where C is saturated and rooted by the initial field i. If we keep backpointers at each step in the algorithm, we have a compact representation of the parse forest.

With this limited algorithm, we can analyze for example: *Den Roman gelesen hat Maria.*

4.2 The algorithm with emancipation

Example: *Den Roman hat Maria gelesen.*

An emancipated constituent is not in the maximal projection of its governor. In our example, *den Roman*, which depends on the past participle *gelesen*, is placed in a field of the main domain headed by the auxiliary *hat*.

Suppose we want to apply our previous algorithm (the CKY parsing without emancipation). We can easily parse the segments *den Roman, hat Maria,* and *gelesen*, but neither *den Roman* and *hat Maria*, nor *hat Maria* and *gelesen* can be combined. Our parsing will still be driven by the topological structure and we will maintain the condition of saturation of the TC of the topological dependent. But we need conditions on the syntactic combinations of the words.

Two solutions are possible.

The first one consists in combining *hat Maria* and *gelesen* but we must bear in mind that *gelesen* still expects a dependent. Therefore we do not require the valency of the topological phrase to be saturated and we must percolate it in a special feature similar to the slash feature of G/HPSG (Gazdar et al. 1985, Pollard & Sag 1994)

The second possibility consists in combining *den Roman* and *hat Maria*. In this case we do not trigger a correspondence rule because no dependency must be built. We must store *den Roman* in a special feature we call visitor (see Hudson 2001 for a similar device), which is the converse of the slash feature. The slash feature allows us to lift up a need (a valency slot to be filled), while the visitor feature allows to hand down a resource (that will fill a valency slot).

Nevertheless, with our conditions on the saturation of TCs, the two strategies are not equivalent and there are both necessary. Let us consider two examples.

Example 1: *Maria hat den Roman gelesen*

Although the sentence is projective, *den Roman* must be analyzed as an emancipated constituent. Indeed, *gelesen* is in the left bracket of the main domain and the maximal projection of *gelesen,* the verb cluster, does not contain its dependent *den Roman*, which is in the Mittelfeld of the main domain headed by *hat*. From the topological point of view, *den Roman* can only combine with *hat* but it is not in the valency of *hat* and it must be considered as a visitor.

Example 2: *Ich glaube, dass Maria den Roman gelesen hat*.

In this example *den Roman* is still emancipated and, from the topological viewpoint, it cannot combine with its governor *gelesen*. It also cannot combine with *hat* because they are separated by

gelesen. The smallest topological phrase containing *den Roman* and *gelesen* also contains *hat*. Therefore the slash strategy is needed and *gelesen* and *hat* must combine together before combining with *den Roman*.

We can now present the complete algorithm. The entries of our parse matrix are of the form [i,j,hd:A,slash:X,visit:Y,top:C]. The slash value X is a bag of 4-tuples (r,B1,B2,b) where r is a syntactic role, B1 and B2 two categories and b a box. The visitor value Y is a bag of pairs (B,f) where B is category and f a field. We do not use a Valency feature; valencies are put expressed directly in the slash feature of the lexical constituent. Our initial segments are of the form [i,i,had:A,slash:X,visit:∅,top:C] where the slash elements are obtained by translating a valency element (r,B) of A into (r,A,B,0) (0 indicates that no emancipation has been done).

Suppose now that we want to combine the parsed segments [i,j,hd:A1,slash:X1,visit:Y1, top:C1] and [j+1,k,hd:A2,slash:X2,visit:Y2, top:C2]. One of them must fill a place in the TC of the other. Suppose that C2 takes the field f2 of C1. It supposes that C2 is saturated and Y2=∅ (the governor of a visitor cannot be in a higher domain). The combination of the two segments give us the parsed segment [i,k,hd:A1,slash:X,visit:Y,top:C] where C is built as in the algorithm without emancipation[10], X is the union of X1 and X'2 and Y is the union of Y1 and {(A2,f2)}. The element of X'2 are obtained by replacing the last value b (the box label controlling the emancipation) by the max (for the permeability order) of b and the top box label of C2

Now such a segment can be reduced by correspondences rules, each one combining a slash element (= a need) with a visitor (= a resource). More precisely, the slash element (r,A,B,b) can merge with the visitor (B,f) if there exist a correspondence rule (r,A,B,f,b') with b=b'. Both elements are suppressed from the bags. This reduction can be made at any moment. It will be necessary to combine the segment with its governor in order to empty the visitor bag. The parsing succeeds if the whole sentence corresponds to a segment [1,n,hd:A,slash:∅,visit:∅, top:C], where C is saturated and rooted by the initial field i.

Example 3: den Roman hat Maria zu lesen versprochen

den Roman hat Maria: [1, 4, hd:Vfin, slash:{(aux,Vfin,Vpp,0)}, visit:{(Nacc,vf)}, top:i-md-lb-v-•-mf*-rb?-nf*]

zu lesen versprochen: [5, 6, hd:Vpp, slash:{(obj,Vzu,Nacc,vc)}, visit:∅, top:rb-vc-h-v-•]

s: [1, 6, hd:Vfin, slash:{(aux,Vfin,Vpp,0), (obj,Vzu,Nacc,vc)}, visit:{(Nacc,vf),(Vpp,rb)}, top:i-md-lb-v-•-nf*]

s can be reduced to [1, 6, hd:Vfin, slash:∅, visit:∅, top:i-md-lb-v-•-nf*] by merging (aux,Vfin,Vpp,0) and (Vpp,rb) with the rule (aux,Vfin,Vpp,rb,0) (particularization of (-,-,V,rb,0)), as well as (obj,Vzu,Nacc,vc) and (Nacc,vf) with the rule (obj,Vzu,Nacc,vf,ed) (particularization of (-,V,-,f,ed)).

4.3 Complexity

Just as for regular CKY algorithms, we can suppose that the parameter sets including the rule sets are bounded.

Accordingly, the results for a grammar without emancipation are similar: The number T of TCs is bounded by the number of box creation rules, because we suppose that the grammar does not use box creation rules recursively. Thus the complexity remains of order $RT^2C^2n^3$ with R being the number of correspondence rules and C the number of categories.

The slash and visitor features are more expensive: We assume the slash and visitor sets to be bounded by K, i.e. we suppose that we do not need to keep more than K entries in the slash and visitor sets at a time.

One slash quadruple allows a finite number S of different configurations. Equally, the number V bounds the number of different configurations of an entry in the visitor bag. A segment carrying slash and visitor sets can appear in S^KV^KTC states. The maximum number of entries in each square of the parse matrix remains $O(R(TC)^2(SV)^Kn^3)$.

We avoid exponential growth only because we restrict the number of slash and visitor entries of

[10] Additionally we need to suppress all the fields of C1 that are lower in C1 than f2 in order to avoid a visitor to find its governor in a lower constituent. Of course we require that these fields were saturated.

each configuration.[11] Our algorithm only verifies the saturation of the subcategorization frame of each predicate, i.e. the possibility of constructing a correct dependency tree for a topological phrase structure, but this does not allow us to construct the dependency tree.

If we kept backpointers at each step of the algorithm, i.e. if we noted down the segments corresponding to the lexical elements that enter into a dependency relation, we would obtain a compact representation of the parse forest and we could reconstruct the dependency tree, making our algorithm grow by $O(n^{4K})$.

4 Conclusion

We have proposed a parsing algorithm for a grammar which allows us to handle very complex phenomena of word order. Moreover this is a rather rich grammar which builds both dependency structures and topological phrase structures. It must be underlined that we do not use our phrase structure to represent the syntactic structure of the sentence, but only for linearization, i.e. as an intermediate step between the text and the syntactic structure proper, which is encoded by a non ordered dependency tree.

Our bottom-up strategy driven by the topological structure forces us to introduce tools equivalent to the slash feature of G/HPSG. We hope that this presentation shed light on the procedural role of the slash feature, and on the complementary possibility of a linguistic analysis using a visitor feature.

Work is in progress to choose useful input and output formats and to implement the presented algorithm. Real values on efficiency will not be available as long as the grammar does not surpass experimental size, as the complexity depends heavily on the number of slash and visitor places, of categories, of box creation rules, and of correspondence rules needed in the linguistic description.

It seems to be an interesting task to compare the performance of this algorithm with the constraint-based approach of Duchier & Debusman 2001 and with efficiency considerations for HPSG as in Nishida et alii 2001.

[11] It is difficult to give an upper bound for the case of the German grammar presented here, because the number corresponds to the maximum number of elements that can depend on the verbs of the right bracket or that are emancipated from an embedded domain. This depends heavily on choices in the linguistic analysis.

References

Bech Gunnar, 1955, *Studien über das deutsche Verbum infinitum*, 2nd edition 1983, Linguistische Arbeiten 139, Niemeyer, Tübingen.

Bröker Norbert, 1998, "Separating Surface Order and Syntactic Relations in a Dependency Grammars", *COLING-ACL'98*, 174-180.

Drach, Erich, 1937, *Grundgedanken der deutschen Satzlehre*, Diesterweg, Frankfurt/M..

Duchier Denys, Ralph Debusmann, 2001, "Topological Dependency Trees: A Constraint-Based Account of Linear Precedence", *ACL 2001*, 180-87.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullum and Ivan Sag *Generalized Phrase structure grammar*, Harvard University Press, Cambridge MA, 1985.

Gerdes Kim, Sylvain Kahane, 2001, "Word Order in German: A Formal Dependency Grammar Using a Topological Hierarchy", *ACL 2001*, 220-27.

Herling, S. H. A., 1821 "Über die Topik der deutschen Sprache". In: *Abhandlungen des frankfurterischen Gelehrtenvereins für deutsche Sprache*. Frankfurt/ M., Drittes Stück, S. 296-362, 394

Hudson Richard, 2000, "Discontinuity", in S. Kahane (ed.), *Dependency Grammars*, *T.A.L.*, 41(1): 15-56, Hermès, Paris.

Kahane Sylvain, Alexis Nasr, Owen Rambow, 1998, "Pseudo-Projectivity: a Polynomially Parsable Non-Projective Dependency Grammar", *COLING-ACL'98*, Montreal, 646-52.

Kathol Andreas, 1995, *Linearization-based German Syntax*, PhD thesis, Ohio State University.

Bernd Kiefer, Hans-Ulrich Krieger, John Carroll, and Rob Malouf. 1999. A bag of useful techniques for e? cient and robust parsing. In *Proceedings of ACL 99*, pages 473–480.

Lenerz Jürgen, 1977, *Zur Abfolge nominaler Satzglieder im Deutschen*, TBL Verlag Günter Narr, Tübingen.

Nishida, Kenji, Kentaro Torisawa and Jun'ichi Tsujii. (2001). "Compiling an HPSG-based grammar into more than one CFG". In *Proceedings of PACLING 2001*. pp. 199--206.

Pollard, C. and I. Sag., 1994, *Head-Driven Phrase Structure Grammar*, Chicago: University of Chicago Press, and Stanford: CSLI Publications.