Lexcalised Parsing of German V2

Yo Sato
Department of Computer Science
Queen Mary, University of London
Mile End Road, London E1 4NS, U.K.

Abstract

This paper presents a method and implementation of parsing German V2 word order by means of constraints that reside in lexical heads. It first describes the design of the underlying parsing engine: the head-corner chart parsing that incorporates a procedure that dynamically enforces word order constraints. While the parser could potentially generate all the permutations of terminal symbols, constraint checking is conducted *locally* in an efficient manner. The paper then shows how this parser can adequately cover a variety of V2 word order patterns with sets of lexically encoded constraints, including non-local preposing of an embedded argument or an adverbial.

1 Introduction

This paper presents a method of parsing V2 word order manifested in a variety of German matrix sentences in a lexicalised and locality-respecting manner: lexicalised, as the V2 pattern is licensed ultimately encoded in *verbs*, in the form of constraints that hold amongst its arguments and itself; locality-respecting, because (a) no constraint that operates on constituents from different subcategorisation frames is invoked and (b) the matrix verb and the preverbal constituent, however 'distant' its origin is, are ordered in the same projection via the slash-based mechanism.

The underlying grammar is loosely linearisation-based, in the sense that word order is dissociated from the syntactic structure in a discontinuity-allowing manner, as presented in Sato (2008). The main benefit of a linearisation approach is that syntactic constituency becomes independent (to a degree) of its surface realisation and hence discourages constituency manipulation for the sake of word order. In line of this spirit I will largely adopt the simple constituency construal that faithfully correspond to its semantics. However, I distance myself from the more or less standard version of linearisation grammar where potentially non-local LP conditions are permitted (Reape, 1993) or word order patterns are imposed at the clause level (as in 'topological field' model of Kathol (2000)).

The crux of the proposal consists in employing a head-corner parsing in which the set of word order constraints are incorporated into a VP's lexical head (i.e. common or auxiliary verb). For a V2 projection, its head verb contains the constraints to the effect that only one of its arguments can be fronted immediately before the verb itself. To enable this, potential discontinuity and obligatory adjacency in part of a phrase is included in the repertoire of word order constraints in addition to the standard LP (linear precedence) constraints.

2 The data

The V2 constructions to be dealt with in this paper are as follows (I will use as an example the tertiary verb gebengive or its past participle gegebengiven throughout):

1. The 'basic' case where dependency between the preverbal constituent and the matrix verb is strictly local, e.g:

Ein Buch geben die Eltern dem Sohn.
a book give the parents the son
'A book the parents give the son'

2. The case where an argument of the lower verb is fronted across the higher auxiliary verb:

   Ein Buch haben die Eltern dem Sohn gegeben.
   a book have the parents the son given
   'A book the parents have given the son'

3. The long-distance dependency case:

   Ein Buch, sagt ein Freund, dass er glaubt, dass die Eltern dem Sohn geben.

   'A book, a friend says that he thinks that the parents give the son'

4. Adjunct fronting

   Heimlich haben die Eltern dem Sohn ein Buch gegeben.
   secretly have the parents the son a book given
   'Secretly the parents have given the son a book.'

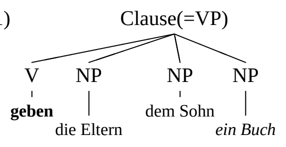
5. Partial VP fronting

   Ein Buch dem Sohn gegeben haben die Eltern.
   Ein Buch gegeben haben die Eltern dem Sohn.

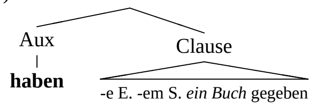
As stated, our approach adopts a linearisation approach in which constituency does not determine the surface realisation, which is handled instead by word order conditions encoded in lexical heads. My contention here is not so much plausibility as a grammar as *neutrality* to particular phrase structures, which linearisation promotes. Therefore I take a rather simplified position to use an entirely uniform phrase structure for the verb-argument structure for common verbs, namely the flat construal where all the arguments as well as the head project onto a clause ('VP') as mutual sisters, although I hasten to add our constraint enforcement could equally apply to configurational analyses. In fact we take an auxiliary verb to subcategorise for a clause rather than the complex verb analysis, and adopt the traditional binary iteration analysis for adjunct-head phrases, to see how our parser fares with configurational analyses.

I sum up the assumed constituency of the above examples graphically as trees (though this has little impact on word order):

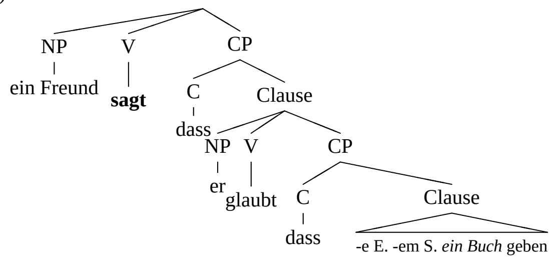
(1)



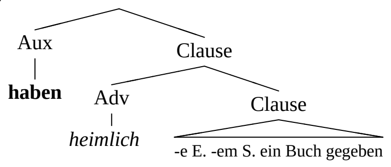
(2)&(5)



(3)



(4)



3 The parser

3.1 Core design

The design of the parser employed here can be called *constrained free word order parsing*. First, it allows for completely free word order at default. The core algorithm for the parse engine is what Reape (1991) presents as a generalised permutation-complete parser, which in turn is based on the preceding proposal of Johnson (1985). Details apart, while using context-free production rules (no multiple left-hand side non-terminal symbols), this algorithm only checks for the *presence* of all the right-hand side constituents, wherever in the string they occur, potentially discontinuously,[1] effectively licensing all the permutations of the given terminal symbols (e.g. $3! = 6$ permutations for the string consisting of *ring*, *up* and *John* including *up John ring* etc.). This 'directionless' parsing is rendered possible by Johnson's 'bitvector' representation of partial string coverage. In the above *up John ring* string, the coverage of the *ring* and *up* combina-

[1]More precisely, it searches for *non-overlapping* combinations, excluding the same word being counted more than once or more than one word counting towards the same rule in the same search path.

tion, which materially constitutes a complex verb, is represented as [1,0,1]. This is then then merged with the bitvector of *John*, [0,1,0] into [1,1,1]. Second, however, this rather promiscuous (and expensive) parsing is dynamically restricted by word order constraints that obtain in individual languages. With sufficient constraints applied during the parse, the above combinations with *ring*, *up* and *John* are restricted to *ring up John* and *ring John up*.

I do not claim for originality in this basic design. Daniels (2005) for example describes an implementation of an algorithm that falls precisely in such style of parsing.[2] The main points of the proposal lie in lexicalisation and localisation, which contrast with the general trend to introduce phrasal and non-local constraint processing for German processing, of which Daniels' work is an example. All the word order constraints are stored in lexicon, more specifically in lexical heads.

To adapt this design to a practical lexically driven parsing, the author implemented a rendering of *head-corner chart parsing*. It is head-corner in the sense described e.g. in van Noord (1991), where the parsing of a production rule always starts from its head. This is necessary for our design because the parser first retrieves the word order information from the head. Furthermore, it requires the words to be processed first by preterminal rules since without processing lexical heads the whole recognition process does not come off the ground. Therefore, a chart parsing algorithm that invokes lexical initialisation is utilised (as described in Gazdar & Mellish (1989) rather than the classical top-down parsing of Earley (1970)).

3.2 Constraint checking and propagation

Since no non-local word order constraints are introduced in our parsing, they can be fully enforced at each application of a production rule. More specifically, the checking of constraint compliance is carried out at the *completer* operation of chart parsing.[3] The data structure of an edge is suitably modified. In addition to the dotted production rule, it needs to carry the constraint set relevant to the corresponding production rule, retrievable from the head, which is always processed first in our head-corner algorithm.[4] Also, as we are adopting the bitvector representation of coverage, an edge contains its corresponding bitvector. The completer operation involves merger of two bitvectors, so the check can be conducted at this stage:

Completer in constrained parsing

Let A and B be symbols, α, β and γ be arbitrary strings, V_1 and V_2 be bitvectors and V^m be their merge, then:

If the chart contains an active edge $\langle V_1$, A$\rightarrow \alpha \bullet$ B $\beta\rangle$ and a passive edge $\langle V_2$, B$\rightarrow \gamma \bullet \rangle$, run the CHECK-ORDER procedure. If it succeeds, add edge $\langle V^m$, A$\rightarrow \alpha$B $\bullet\, \beta\rangle$ to the chart if V_1 and V_2 are mergeable. If it fails, do nothing.

The CHECK-ORDER procedure consists in a bitwise comparison of bitvectors. It picks out the bitvectors of the categories in question and checks the compliance of the newly found category with respect to the relevant constraints. If for example *A*, *B* and *C* had been found at [0,1,0,0,0], [0,0,1,0,1] and [1,0,0,1,0] respectively, this would validate $A \prec B$ but not $A \prec C$. Thus the edges for string combinations that violate the word order constraints would not be created, eliminating wasteful search paths.

As we will shortly see, the constraint type that checks continuity of a phrase is also introduced. Therefore the phrase (dis)continuity can also be ascertained *locally*, which is a major advantage over a parsing that relies largely on concatenation. Thus, the cost of constraint checking remains very small despite the capability of processing discontinuity.[5]

Note however that by locality is meant subcategorisation locality (or 'selection' locality as described in Sag (2007)): whatever is in the same subcategorisation frame of a lexical head is considered local. Depending on the adopted analysis, constituents 'local' in this sense may of course occur in different trees. Constraints on such 'non-local' —in the tree sense but not in the subcategorisation sense— constituents are still enforceable in the implemented parser. The unused constraints at a node,

[2]A foregoing implementation by Müller (2004) also employs bitvector-based linearisation approach.

[3]The equivalent operation is called the 'fundamental rule' in Gazdar & Mellish (1989).

[4]This retrieval of word order information is carried out at the *predictor* stage of chart parsing.

[5]It is worth mentioning that the bitvector checking is conducted over the whole string, the effect of applied constraints will be never lost.

for example some constraint applicable to the verb and its subject at the VP node in the configurational (subjectless-VP) analysis, is made to propagate up to the upper node. Thus it is no problem to enforce a constraint over 'different trees', as long as it is applied to 'local' constituents in our sense.[6]

4 Possible constraints and subtyping

It is crucial, if the computational properties of the parser is to be transparent in constrained free word order parsing, to identify the kind of word order constraints admitted into lexical heads. We will remain relatively conservative, in introducing only two operators for constraint encoding. We first invoke the binary LP operator ($\prec$) in a conventional sense: the whole (or, equivalently, right-periphery) of a string for category *A* needs to precede the whole (or left-periphery) of a string for category *B* to satisfy $A \prec B$ (I will use the shorthand $A \prec (B, C)$ to express $(A \prec B) \wedge (A \prec C)$. Crucially, the contiguity operator () is added. It takes a *set* of constituents as its operand and requires the constituents in it to be contiguous, regardless of their order. Thus, $\{A, B, C\}$ encodes the requirement for A, B and C *as a whole* forming a contiguous string. For example, the string *I ring John up* does not satisfy $\{ring, up\}$ but does satisfy $\{ring, John, up\}$.

Also important is how to succinctly generalise on the word order patterns now encoded in lexical items, as one would certainly want to avoid a tedious task of writing them all individually, if they allow for broader classification. For example the English transitive verb generally follows its subject argument and precedes its object argument, and one would naturally want to lump these verbs under one umbrella. For such a cluster of lexical heads, we will introduce a *word order (sub)type*. More pertinently, the German verbs may be classified into *v1-verb*, *v2-verb* and *vf-verb* according to the positions of their arguments in their projection. We will also allow *multiple inheritance* that becomes standard in the typed feature system (cf. Pollard and Sag (1987)).

[6]See Sato (2006) for details.

5 Constraints for V2

5.1 General setup

To enforce the V2 word order pattern lexically, I propose to use a combination of two word order subtypes: *dislocating-verb* (*disl-v*) and *matrix-v2-verb* (*mtrx-v2-v*). The former type represents a verb one of whose arguments is to be 'dislocated'. A verb of this type can thus be characterised as 'contributing' the dislocated (preverbal) element. The latter, on the other hand, is the type that is projected onto a matrix sentence. This type should be constrained such that one dislocated constituent must —and only one may— precede and be adjacent to the verb itself. It may be characterised as a verb that provides a locus —immediately before itself— of, or 'receives' the dislocated element.

Dislocation is handled by a constraint percolation mechanism. I assume the dislocated constituent is pushed into a storage that then participates in a slash style percolation, although the storage content would still need to be ordered by lexicalised constraints rather than by the percolation mechanism itself, as they are the sole resource for word order.[7] Thus the checking as regards the dislocated constituent is conducted at each projection in the percolation path, hence locally, while the percolation mechanism gives some 'global' control over dislocation. Not just the positioning of the dislocated constituent at the left-periphery of the whole sentence, but the assurance of a *global* singularity restriction of dislocation —not just one constituent per clause in multiple embeddings— becomes thus possible.

Let args be the set of the arguments of a *disl-v*, disl be that of the dislocated one and situ be that of the remaining arguments, i.e. $\mathsf{disl} \subset \mathsf{args}$ where $|\mathsf{disl}| = 1$ and $\mathsf{situ} = \{x | x \in \mathsf{args} \wedge x \notin \mathsf{disl}\}$. Then the type *disl-v* can be characterised as having the following constraint:

disl-v: $\mathsf{disl} \prec \mathsf{situ} \quad (\mathsf{disl} \rightarrow \mathsf{disl}_{st})$

Simply put, this says that the arguments are divided into two parts, the dislocated and in-situ parts, the former of which precedes the latter. We assume, as

[7]The adopted mechanism is close to Penn (1999), though he invokes potentially non-local topology-based constraints and removes the filler and gapped head entirely.

in the standard treatment, there is only one dislocated constituent, until we consider the VP fronting. The notation with an arrow on the right indicates this singleton set is pushed into the storage that is propagated upwards.

The *mtrx-v2-v* type is then characterised as follows:

mtrx-v2-v: $\mathsf{disl}_{st} \prec \mathsf{verb},\ \ \{\mathsf{disl}_{st}, \mathsf{verb}\}$

This simply says the dislocated constituent (stored in a lower node and percolated) immediately precedes the matrix verb. (For the following presentation, the storage-related notations will be omitted and implicitly assumed unless necessary. Also, the set variables disl and args will be used with the same meaning.)

Thus the combination of the two types gives, for example where $\mathsf{args} = \{A, B, C\}$, $\mathsf{disl} = \{A\}$ and the matrix verb is V, the following constraint set:

$$\{A \prec (B, C),\ \ A \prec V,\ \{A, V\}\}$$

which essentially says that the dislocated A immediately precedes the matrix verb V and precedes (not necessarily immediately) the in-situ B and C.

5.2 Local case

To begin with, let us see a case where dependency between the preverbal constituent and the matrix verb is strictly local, taking (1) as an example. Note first that there are six possible variants:

(1)

a. Die Eltern geben dem Sohn ein Buch.
b. Die Eltern geben ein Buch dem Sohn.
c. Dem Sohn geben die Eltern ein Buch.
d. Dem Sohn geben ein Buch die Eltern.
e. Ein Buch geben die Eltern dem Sohn.
f. Ein Buch geben dem Sohn die Eltern.

In this case, *geben* is both a matrix (argument-receiving) and dislocating (argument-contributing) verb. This means that the two subtypes should be overloaded. Let us call this overloaded sub-species *disl-mtrx-v2-v*: which is given the following specification:

disl-mtrx-v2-v:
$\mathsf{disl} \prec \mathsf{situ},\ \ \mathsf{disl} \prec \mathsf{verb},\ \ \{\mathsf{disl}, \mathsf{verb}\}$

To adapt this type to our verb, *geben*, where we represent its arguments as sNP (subject NP), ioNP (indirect object NP) and doNP (direct object NP), we obtain, for the case where sNP is preposed:

$$\{\mathsf{sNP} \prec (\mathsf{ioNP}, \mathsf{doNP}),\\ \mathsf{sNP} \prec \mathsf{geben},\ \ (\mathsf{sNP}, \mathsf{geben})\}$$

where the constraints on the first line is inherited from *disloc-v* while those on the second from *matrix-v2-v*. This corresponds to the sentences (a) and (b) above. The followings are the cases where ioNP and doNP are preposed, corresponding to (c,d) and (e,f), respectively.

$$\{\mathsf{ioNP} \prec (\mathsf{sNP}, \mathsf{doNP}),\ \ \mathsf{ioNP} \prec \mathsf{geben},\ \ (\mathsf{ioNP}, \mathsf{geben})\}$$
$$\{\mathsf{doNP} \prec (\mathsf{sNP}, \mathsf{ioNP}),\ \ \mathsf{doNP} \prec \mathsf{geben},\ \ (\mathsf{doNP}, \mathsf{geben})\}$$

These possible sets are enforced in the manner of exclusive disjunction, that is, only one of the above three sets actually obtains. This does not mean, however, each set must be explicitly stated in the verb and processed blindly. Only the abstract form of the constraint, as described under the type specification above, is written in the lexicon. During parsing, then, one of the sets, as dynamically found to match the input string, is computed and applied. In the subsequent discussion, therefore, only the direct-object fronting case is considered as a representative example for each construction.

5.3 Argument fronting across auxiliary

We now consider the cases where the dependency is not local, starting with an auxiliary-involving case. The dependency between an auxiliary and an argument of its lower verb is, according to the Aux-Clause construal adopted here, is not local. We can however succinctly specify such non-local V2 renderings as a case where the above two types are instantiated separately in two verbs. The example is reproduced below:

(2) Ein Buch haben die Eltern dem Sohn gegeben.

The argument-contributing gegebengiven is, as before, assigned the *disl-v* type, but is further subtyped and inherits the constraints also from *vf-v* (v-final verb), reflecting the fact that it occurs head-finally.

gegeben (type *disl-vf-v*):
$\{\mathsf{doNP} \prec (\mathsf{sNP}, \mathsf{ioNP}),$

$(\mathsf{sNP}, \mathsf{doNP}, \mathsf{ioNP}) \prec \mathsf{gegeben}\}$

The dislocated doNP climbs up the tree ((2) in Section 2) in the storage, which is then subject to the constraints of matrix *haben* at the top node. This argument-receiving auxiliary *haben* is, as before, given the *mtrx-v2-v* status.[8].

haben (type *mtrx-v2-v*):
$\{\mathsf{doNP}_{st} \prec \mathsf{haben}, (\mathsf{doNP}_{st}, \mathsf{haben})\}$
Thus the dislocated *ein Buch* is duly placed at the left-periphery in a manner that forbids intervention between itself and the matrix verb.

5.4 Long-Distance Dependency

Having dealt with an argument fronting of the auxiliary construction as a non-local case, we could now extend the same treatment to long-distance dependency. Our example is:

(3) Ein Buch, sagt ein Freund, dass er glaubt, dass die Eltern dem Sohn geben.

('A book, a friend says that he thinks that the parents give the son')

In fact, it suffices to endow exactly the same type as *gegeben*, i.e. *disl-vf-v*, to the occurrence of *geben* in a subordinate clause.[9]

geben (in subord. clause, type *disl-vf-v*):
$\{\mathsf{doNP} \prec (\mathsf{sNP}, \mathsf{ioNP}),$
$(\mathsf{sNP}, \mathsf{doNP}, \mathsf{ioNP}) \prec \mathsf{geben}\}$

This ensures that the dislocated argument goes progressively up towards the top node. To prevent this argument from being 'dropped' the half way through, however, the non-matrix CP-taking verbs 'in the middle' that should be bypassed, in our case *glaubt*, needs to possess the constraint that pushes the dislocated element to the left of itself:

glaubt (in subord. clause, type '*middle-v*'):[10]
$\{\mathsf{doNP}_{st} \prec \mathsf{glaubt}\}$

Finally, a *mtrx-v2-v*, in our case *sagt*, takes care of placing the dislocated constituent immediately before itself.

sagt (type *mtrx-v2-v*):[11]
$\{\mathsf{doNP}_{st} \prec \mathsf{sagt}, (\mathsf{doNP}_{st}, \mathsf{sagt})\}$

5.5 Adjunct fronting

I declared at the beginning to use the traditional binary adjunction analysis for adjunct-head phrases.[12] In order to achieve this, I first propose a fundamental conceptual shift, given the iterability and optionality of adjuncts. In the traditional concept of adjunct-head phrases, it is the adjunct that selects for the head it modifies rather than the other way round. Also semantically, the adjunct is considered the 'semantic head' that works as a functor. In light of this background, it is not implausible to take the adjunct as the 'parsing head' equipped with word order constraints. In fact, the opposite option — equipping the syntactic head with its relative word order with adjuncts— is not as feasible in our lexical head-corner parsing. The iterability of adjuncts means that the head would have to be equipped with an infinite number of adjuncts as its 'arguments', which would lead to various uninstantiation problems. Therefore, I swap the statuses and treat, in terms of parsing, the adjunct as a functor with word order constraints incorporated relative to its modifiee.

Thus, the word order constraints are now given to the lexical adjuncts also. I will take as an example adverbs.[13] Adverbs are now the potential locus of word order patterns relative to its modifiee (clause/VP), but are not given any specific constraint in German generally, because one can appear either after or inside a clause. Our focus is solely on the possibility of putting one *before* the clause it modifies, when it is subject to the V2 constraint. This is handled simply by saying, for such a type, which we call *disl-adverb*, it *dislocates itself*, in the manner of

[8]More precisely this also involves $\mathsf{haben} \prec \mathsf{VP}$(gapped)

[9]This means that, given the identical morphological form, *gegeben* is type-ambiguous between the matrix and subordinate occurrences. This does not add too much to parsing complexity, however, as this 'ambiguity' is quickly resolved when one of its argument is encountered.

[10]The constraints applicable to the usual finite verb is omitted, i.e. $\mathsf{sNP} \prec \mathsf{glaubt}$ and $\mathsf{glaubt} \prec \mathsf{CP}$(gapped).

[11]Likewise: $\mathsf{sagt} \prec \mathsf{CP}$(gapped) omitted.

[12]That is against the temptation for a constituency change that renders adjuncts sisters on par with arguments (cf. Bouma et al (2001)), in which case V2 would simply fall out from the foregoing word order types.

[13]The same treatment can be extended to prepositional adjuncts (remember the unused constraints will percolate up to the maximal projection).

‘head movement’ which is widely used in German syntax (Kiss and Wesche, 1991; Netter, 1992).

disl-adverb: adv (adv$\rightarrow$ disl_{st})

This specification ensures the adverb itself goes onto the extraction path, to be placed at the left-periphery, triggered by the *mtrx-v2-v* type. The singularity of the adverbials at the prerverbal position is ensured by means of percolation storage control.

6 Verbal Fronting

Our last challenge concerns fronting of verb or verbal projections. From the preceding discussion, an option that suggests itself is to treat the verb fronting as the case of verb dislocating itself. I will indeed propose a strategy along this line, but this avenue proves more difficult due to complications specific to verb-related fronting. Firstly, generally such fronting is limited to the environment of a lower VP governed by a higher verb such as an auxiliary, as can be seen from the following contrast:

(4)

a. Gegeben haben die Eltern dem Sohn ein Buch.

b. *Geben, sagt ein Freund, dass die Eltern dem Sohn ein Buch.

Second, the type we used for *gegeben* in Section 5.3, namely *disl-vf-v*, clearly does not work, as the verb does not occur phrase-finally (but in fact initially) relative to its sisters in (4a). Some relaxation of LP constraints seem to be in order.

Thirdly, German displays a variety of ways to front *part* of a VP:

(5)

Gegeben haben die Eltern dem Sohn ein Buch.
Dem Sohn gegeben haben die Eltern ein Buch.
Ein Buch gegeben haben die Eltern dem Sohn.
Dem Sohn ein Buch gegeben haben die Eltern.

This raises the question of whether this fits in the V2 pattern at all, coupled with the ongoing debate on the status of the preverbal string. Quite apart from the theoretical debate, however, how best to adequately generate these patterns is an acute parsing issue. We are assuming the flat clause=VP anaylsis, so relaxing the singularity condition seems unavoidable.

Fourthly, to make the matter worse, allowing multiple frontings and dropping LP requirements does not solve the problem, as ordering of the preverbal constituents is constrained, as shown in the following data:

(6)

*Gegeben dem Sohn haben die Eltern ein Buch.
*Dem Sohn gegeben ein Buch haben die Eltern.

It is a great challenge for any syntactician to provide a unified account for such complex behaviour, and I confine myself here to offering the ‘solution’ sets of constraints that adequately generate the desired string. What I offer is this: allowing multiple dislocations only for the verbal fronting cases via a new word order subtype, while retaining the verb-final LP conditions for these dislocated constituents.

For this new type we first relax the singularity condition for dislocation. To allow multiple dislocations, it would suffice to drop the $|\mathsf{disl}| = 1$ condition, but an unrestricted application of $\mathsf{disl} \subset \mathsf{args}$ would lead to overgeneration, due to two further constraints applicable: (1) not *all* arguments can and (2) the subject argument cannot be fronted along with the verb (as in (a) and (b) below, respectively):

(7)

a. *Die Eltern dem Sohn ein Buch gegeben haben.

b. *Die Eltern gegeben haben dem Sohn ein Buch.
*Die Eltern ein Buch gegeben haben dem Sohn.

Therefore we add the conditions to exlude the above, along with the the verb-final constraint applicable the dislocated constituents to exclude (6). Let us call this type *frontable-v*. The constraint specification is as follows:

gegeben (*frontable-v*):
$\mathsf{disl} = \{\mathsf{gegeben}\} \cup \mathsf{ptargs},\ \ \mathsf{ptargs} \prec \mathsf{gegeben}$
where $\mathsf{ptargs} \subset \mathsf{args}$ and $\mathsf{sNP} \notin \mathsf{ptargs}$

The proposed constraint set might strike as rather *ad hoc*. It would clearly be better to treat both the fronted and non-fronted occurrences of *gegeben* as sharing some common word order type, and what is meant by ‘applying the constraints amongst the dislocated constituents’ needs to be fleshed out. Thus this may not be an elegant solution, but nevertheless is an generatively adequate solution. More importantly it serves as a good example for the flexibility and adaptability of constrained free word order parsing, because it handles a rather complex word order pattern in a way neutral to grammatical construal, i.e. without invoking constituency manipulation.

7 Concluding Remarks

I conclude this paper by responding to a natural objection: why would one have to go through this convoluted route of lexical word order control, when the 'natural' way to constrain V2 —or V1 and VF, for that matter— would be to have some 'global' patterns pertinent to clause types? My responses are primarily engineering-oriented. First, lexicalised encoding gives the parser, through locality restriction, a certain control over computational complexity, as the search space for constraint enforcement is restricted.[14] However this not an entirely unique, if more amenable, feature to lexicalised parsing, as one could impose such a control in non-lexicalised parsing. The advantage truly unique to lexicalising word order lies in rendering the parser and grammar independent of surface realisation and hence re-usable across languages. In short, it promotes modularity. As we have seen, though the parser needs to conform to a certain strategy, the word order component is fairly independent, as a separate procedure which can be modified if for example more types of word order operators are needed. The grammar could also be kept more compact and cross-linguistically applicable, because word order is abstracted away from constituency. Therefore, paradoxically, an advantage of lexicalising German parsing is to enable the same parser/grammar to be used in other languages too, even if it is not naturally suited to the language.

References

Gosse Bouma, Robert Malouf, and Ivan Sag. 2001. Satisfying constraints on extraction and adjunction. *Natural Language and Linguistic Theory*, 19(1).

Mike Daniels. 2005. *Generalized ID/LP Grammar*. Ph.D. thesis, Ohio State University.

Jay Earley. 1970. An efficient context free parsing algorithm. *Communications of ACM*, 13:94–102.

Gerald Gazdar and Chris Mellish. 1989. *Natural Language Processing in Prolog*. Addison Wesley.

Mark Johnson. 1985. Parsing with discontinuous constituents. In *Proceedings of the 23rd Annual Meeting of the ACL*, pages 127–132.

Andreas Kathol. 2000. *Linear Syntax*. OUP.

Tibor Kiss and B Wesche. 1991. Verb order and head movement. In O Herzog, editor, *Text Understanding in LILOG*, pages 216–40. Springer.

Stefan Müller. 2004. Continuous or discontinuous constituents? a comparison between syntactic analyses for constituent order and their processing systems. *Research on Language and Computation 2(2)*.

Klaus Netter. 1992. On non-head non-movement. An HPSG treatment of finite verb position in German. In G. Görz, editor, *Proceedings of KONVENS 92*. Springer.

Gerald Penn. 1999. Linearization and Wh-extraction in HPSG: Evidence from Serbo-Croatian. In R. Borsely and A. Przepiorkowski, editors, *Slavic in HPSG*. CSLI.

Carl Pollard and Ivan Sag. 1987. *Information-Based Syntax and Semantics*. CSLI.

Mike Reape. 1991. Parsing bounded discontinuous constituents: Generalisation of some common algorithms. *DIANA Report, Edinburgh University*.

Mike Reape. 1993. *A Formal Theory of Word Order*. Ph.D. thesis, Edinburgh University.

Ivan Sag. 2007. Remarks on locality. In Stefan Müller, editor, *Proceedings of HPSG07*. CSLI.

Yo Sato. 2006. A proposed lexicalised linearisation grammar: a monostratal alternative. In Stefan Müller, editor, *Proceedings of HPSG06*. CSLI.

Yo Sato. 2008. *Implementing Head-Driven Linearisation Grammar*. Ph.D. thesis, King's College London.

Oliver Suhre. 1999. Computational Aspects of a Grammar Formalism for Languages with Freer Word Order. Diplomarbeit, Eberhard-Karls-Universität Tübingen.

Gertjan van Noord. 1991. Head corner parsing for discontinuous constituency. In *Proceedings of the 29th annual meeting on ACL*, pages 114–121.

[14] For a complexity analysis of such grammar, see Sato (2008) and Suhre (1999).