

Constrained Free Word Order Parsing with Lexicalised Linearisation Grammar

Yo Sato

Department of Computer Science
King's College London
Drury Lane, London WC2B 5RL
yo.sato@kcl.ac.uk

Abstract

This paper presents a non-CFG parsing algorithm that works with a lexicalised HPSG grammar, in which the word order related constraints (WOCs) originate from *lexical heads*. Our parser can handle cross-serial scrambling in an efficient manner, referring dynamically to the WOCs of the lexicon and restricting its search space.

1 Backgrounds

1.1 Case for Non-CFG Parsing

After Shieber (1985) demonstrated that there is a natural language that defies CFG parseability, linguists generally came to agree that for adequate description of some languages at least ‘mildly’ context-sensitive grammars are required (Joshi, 1985), but a non-CFG parser is yet to be adopted for a large-scale implementation of computational grammar. There are at least two reasons behind this situation. First, there is a perception that non-CFG constructions are marginal in natural languages. Secondly, a context-sensitive parsing is known to be costly. Thus, many NLP practitioners decide that non-CFG parsing backbones are just not worth the price.

One of the main obstacles for CFG parsing is the discontinuity caused by ‘interleaving’ or ‘crossing’ of elements from different phrases in natural languages with ‘freer’ word order, where

discontinuity and scrambling is frequently observed. To illustrate, I take up the ‘incoherent’ variety of the object control verb construction in Japanese and German, as this is where the contrast in CFG parseability emerges most clearly. First, a typical Japanese example:

(1J) Taro-ga Hanako-ni sono hon-o
Taro(Nom) Hanako(Dat) the-book(Acc)

yomu-no-o yurusu.
to-read allow

‘Taro allows Hanako to read the book’

(1J') Taro-ga [sono hon-o] hanako-ni [yomu-no-o]
yurusu.
Hanako-ni [sono hon-o] Taro-ga [yomu-no-o]
yurusu.
[Sono hon-o] Hanako-ni Taro-ga [yomu-no-o]
yurusu.
...

Here the (1J') instances are scrambled variants of the ‘canonical’ (1J). In the traditional ‘bi-clausal’ analysis according to which the object control verb subcategorises for a VP complement (‘to read the book’) as well as nominal complements (including the subject), this embedded VP, *sono hon-o yomu-no-o*, becomes discontinuous, being intervened by a complement (or two) of the main verb, in (1J') (in square brackets).

One can observe the exact parallel in German, except that free scrambling manifests itself only in restricted environments like subordinate clauses: sure enough, you get exactly the same discontinuity-causing scramblability as in (1G'):

- (1G) Ich glaube, dass der Fritz der Sabine
I believe Comp Fritz(Nom) Sabine(Dat)
- das Buch zu lesen erlaubt.
the book(Acc) to read allow
- ‘I think that Fritz allows Sabine to read the book’
- (1G’) Ich glaube, dass der Fritz [das Buch] der Sabine
[zu lesen] erlaubt
Ich glaube, dass der Sabine [das Buch] der Fritz
[zu lesen] erlaubt
Ich glaube, dass [das Buch] der Sabine der Fritz
[zu lesen] erlaubt
...

One well-known CFG response is to use ‘mono-clausal’ analysis or *argument composition* (Hinrichs and Nakazawa, 1990; Manning et al., 1999), according to which the higher verb and lower verb are combined to form a single *verbal complex*, which in turn subcategorises for nominal complements. Under this treatment both the verbal complex (*allow to read*) and its ‘complements’ (the allowee, allowee and what is read) are rendered continuous, and hence CFG-parseable, in all the above examples. However, this does not quite save the CFG parseability, in the face of the fact that you could extrapose the lower V + NP for the German example:

- (2G) Ich glaube, dass der Fritz der Sabine [erlaubt], das Buch [zu lesen].

Now we have a discontinuity of ‘verbal complex’ instead of complements (in square brackets). In Japanese, though rightward extraposition is not allowed due to the head-final constraint, one can separate the lower verb towards the left:

- (2J) Sono hon-o [yomu-no-o] Taro-ga Hanako-ni [yurusu].

Thus either way, some discontinuity or other is inevitable. Object control verbs are certainly by no means rare, both in German and in Japanese. Furthermore non-configurational languages show discontinuity of other sorts that do not lend itself to a straightforward CFG parsing (Kathol, 2000; Chung, 1998; Yatabe, 1996; Saito, 1985). The CFG-recalcitrant constructions exist in abundance, pointing to an acute need for non-CFG parsing.

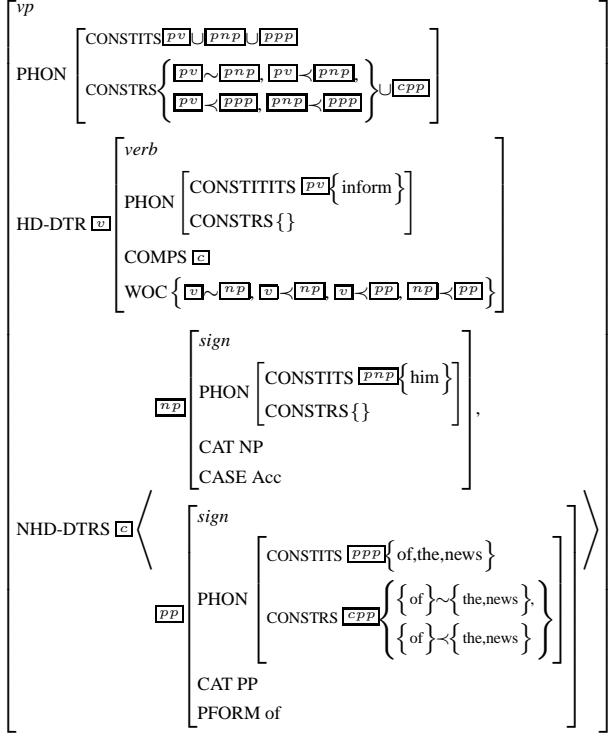


Figure 1: Example of feature structure with WOC

1.2 Incorporating LP into Words

As a grammar formalism, we choose to work within HPSG, but break away from the influential Word Order Domain theory proposed by Reape (1993; 1994), as its DOM feature constitutes a layer dedicated to linearisation and effectively entails a departure from a lexicalist framework. Instead we try to retain the lexicalist contour of HPSG, by incorporating the Word Order Constraints (WOC) feature into *lexical heads*.¹

Here I present a brief overview of the formalism with an example. For full details the reader is referred to Sato (2006; forthcoming a). Figure 1 is the AVM for the English VP *inform him of the news*, where one finds the WOC feature in its head verb, *inform*: here the binary ADJ (adjacency) and LP relations, represented with the infix notation ~ and < respectively, encode the constraints. We are assuming, somewhat simplisti-

¹Proposed discontinuity-capable computational grammars are generally also non-lexicalist (Götz and Penn, 1997; Daniels and Meurers, 2004). Notable exceptions are Covington (1994) and Manandhar (1995).

cally, the following: first, the general head-initial constraint of English VPs; secondly, the verb and its indirect object must be adjacent (*inform yesterday him of the news* is not allowed) and the indirect object must precede the of-PP (*inform of the news him* is not allowed). Notice that there is no ADJ requirement between the indirect NP and of-PP. This allows an intervention between the two constituents, as in *inform him yesterday of the news*.

The job of constraining the phonological string is borne by the CONSTRS subfeature of the now compound PHON, into which the WOC values emanating from lexical heads are percolated. In this example the PHON feature of the VP encodes the legitimate word order patterns which obey the specified constraints in an underspecified manner. Also notice that the WOCs are passed up *all the way through* to the upper nodes into PHON|CONSTR (notice the NHD-DTR's PHON|CONSTR values, in this example only \overline{PPP} , are unioned), to ensure that all the WOCs can be enforced at any point of successive projections. For example, it is not at the stage of the head's projection but at some upper node that an ADJ constraint may be violated: the ungrammatical *inform yesterday him of the news* should be blocked in the application of VP→VP adv, one step after the projection of *inform*. Yet, due to the cumulative inheritance of WOCs, the relevant WOC, \overline{PVP} , is also found in this upper node, blocking the above sequence there.

2 Constrained non-CFG Parsing

2.1 Existent Algorithms

Before delving into the parsing algorithm for the lexicalist grammar outlined above, let us first look at the previously proposed algorithms for discontinuity parsing, some key ideas of which are crucially used for our new algorithm.

Now, the most general form of parsing algorithms capable of crossing discontinuity would be such that a string of words are accepted as a phrase whatever linear position each of the constituent occupies, i.e. all the permutations of constituents, as well as the discontinuity, are ac-

cepted. So, if I take a German infinitiv VP example, *das Buch ... zu lesen*, it can be realised under this parsing (counterfactually) at least in 4! ways, [das, Buch, zu, lesen], [das, zu, Buch, lesen], [Buch, lesen, das, zu] etc., as well as the intervened versions of these, such as [das, Buch, -, zu, lesen], [Buch, lesen, -, das, -, zu], etc., where '-' represents some intervening constituent from another phrase.

Reape (1991) proposes an algorithm for such 'permutation complete' language in tabular parsing, i.e. one that records intermediate parses in 'edges'.² By using Johnson's (1985) idea of using *bitmasks*, he modifies the working of edges such that an active edge is created (either by finding a rule or by moving the dot) *in whatever position* the corresponding passive edge is found. Bitmasks are used as the indication of the parse progress in an edge, instead of the 'vertices' of the traditional CFG tabular parser. Let us see how the parser works with the German VP *das Buch ... zu lesen*, as realised discontinuously in, say, in the higher projection [das, Buch, der, Fritz, dem, Sabine, zu, lesen, erlaubt].

Starting from where the parsing of the NP and V-Inf are completed, their (passive) edges look like the following. The bitmasks, to the right of the rules, encode that the constituents have been found at the first and second positions and the fifth and sixth positions respectively in the seven-word long string.

- 1 NP → Det N • [das,Buch, -, -, -, -, -]
- 2 V-Inf → [zu] V • [-, -, -, -, zu,lesen, -]

Edge 2 above triggers the creation of the following active edge, given the rule VP→ V-Inf NP:

- 3 VP → • V-Inf NP [-, -, -, -, -, -]

and then we proceed to another edge (dot move):

- 4 VP → V-Inf • NP [-, -, -, -, zu,lesen, -]

and then to:

- 5 VP → V-Inf NP • [das,Buch, -, -, zu,lesen, -]

at which point the parse of the VP is completed.

Notice that in Edges 4 and 5 'merging' the two bitmasks has taken place, in synchronisation with

²His algorithm follows the CYK algorithm, although we will be using the bottom-up chart parser by Gazdar and Mellish (1989).

a dot move: the bitmask of Edge 4 is the result of merging (vacuously) those of Edges 2 and 3, while Edge 5 merges the bitmasks of 1 and 4. As can be seen, the parse proceeds by filling in vacant positions, wherever they are, rather than going from right to left. It should be easy to see this parsing scheme accepts discontinuous and cross-serially permuted phrases.

Clearly, however, the problem with such a parser is lack of efficiency, as in this form the worst-case complexity will be exponential ($O(n! \cdot 2^n)$, where n is the length of string). It is desirable, or even essential, to restrain it for practical parsing of real data. However, the fact that natural languages usually do *not* allow such free permutation presents itself as a means to restrict the search space. In a recent development, Daniels and Meurers (2004) exactly do this: using the word order constraints specific to particular languages dynamically during parse, to block creation of edges known to be wasteful.

In their implementation, Daniels and Meurers incorporate into the edge two additional bitmasks, what they call *positive* and *negative masks*, which encode the bits that have to be occupied, and those that cannot be occupied, respectively. Continuing with the same German example, you might for instance like the head-final constraint to restrict further search, when you have reached the stage where V-Inf has been found in the edge of the embedded VP, since the moment the parser has found *zu lesen*, it becomes known that the complements of the infinitival verb have to appear before it. This information can be expressed in the negative mask [-, -, -, -, zu, lesen, x], where x indicates the position that cannot be occupied by its complement. Thus no further edge is created even if an NP is found in this position. The positive mask is the complementary version of the negative one. You might like to add, at an earlier stage of the parse when *lesen* has been found for the *zu-inf-v* edge, this positive mask: [x, x, x, x, -, lesen, x] (‘the remaining string, namely *zu*, has to occupy the ‘-’ marked position’). Thus, you can stop the parser from searching the positions the categories yet to be found cannot occupy, or force it to search only the positions they have to

occupy.

An important job remains to be done, however: one needs to state these word order constraints in the grammar, preferably in such a way that the parser easily ‘translate’ them into the positive/negative masks. Daniels and Meurers’ answer to this task is a rather traditional one: stating them as LP attachments to phrase structure rules and the parser picks them in rule applications. They modify the standard HPSG rather extensively in a way similar to GPSG, in what he calls ‘Generalised ID/LP Grammar’.

As we have seen, however, it is possible to stick to the lexicalist spirit of HPSG while incorporating the word-order constraints successfully even for cross-serial instances. Let me finish this section by alluding to another advantage over the grammar with LP-attached phrase structure rules. Essentially, our lexicalist architecture dispenses with any separate phrase structure rule component as an intermediary between grammar and parser. Thanks to this simpler structure, it allows for reusability of commonly shared parts of specific grammars. Let us suppose two languages have a type of verb with almost exactly the same feature structures including subcategorisation frame, the only difference being the word order that holds of the verb itself and its complements. In our architecture it is easy for the two grammars to share the common part, with only the WOC feature differing.³ In contrast in Generalised ID/LP Grammar, even though its phrase structure rules allow for considerable freedom in word order, one has to re-state rules for each language after all, since they must be accompanied by language-specific LP constraints.

2.2 New Algorithm for Word Order Checking

It is relatively straightforward to devise an algorithm that uses the word-order information stored in the feature structure of lexical heads. While the idea of maintaining efficiency by dynamically ap-

³This will also lend itself, eventually, to a complementary modularisation of universal/specific components for grammar building, where one only needs to write the reusable ‘universal’ grammar once.

plying the word order constraints during the parse remains the same as in Daniels and Meurers, now we directly refer to the WOC features in the head instead of relying on LP-attached phrase structure rules or media carrying the WOC information like positive/negative masks.

The new algorithm is similar to the tabular parsing described in Section 2.1, with some modifications I will now describe. While we retain the bitmask for occupied word positions, the WOC constraints picked up from the feature structure of the head are carried in an edge instead of negative/positive masks. When an active edge is about to embrace the next category, these constraints are checked and enforced, limiting the search space thereby. Such parsing dictates particular search strategies, however: it needs to be a head-corner (van Noord, 1997) and bottom up parser, unless some delayed unification technique is used (Morawietz, 1995). Now that the WOC information is stated in the lexical heads, they need to be instantiated first if they were to be used in parsing. The non-head daughters also need to be instantiated before the constraint checks, so the strategy must be bottom-up.

Now that there is no phrase structure rule in the grammar, we are also in need of their counterparts to be fed into the edges for parsing. This can be straightforwardly achieved at the same time when the WOCs are picked up from the lexical head, as it also contains the subcategorisation information as well. Its potential daughters are found in the COMPS feature and the parser constructs the projection rule on the fly. Thus an active edge is created consisting of this ‘rule’ dynamically generated from the head and of its WOCs. This initial edge looks like the following:

$\langle \text{Mum} \rightarrow \text{Hd-Dtr} \bullet \text{Nhd}_1 \text{ Nhd}_2 \dots \text{Nhd}_n; \text{WOCs} \rangle$

where WOCs is the set of ADJ and LP constraints picked up, if any (e.g. $\{\text{Hd-Dtr} \prec \text{Nhd}_2, \text{Nhd}_1 \sim \text{Nhd}_2, \dots\}$). This now tries to find the rest – non-head daughters, and whenever a non-head daughter is found and is about to be embraced (before dot move), the relevant word-order constraints, if any, are checked. This checking is conducted in the following way. The following is the

representation of an edge when the parsing proceeds to the stage where some non-head daughter, in this representation Dtr_j , has been parsed, and Dtr_j is to be searched for.

$\langle \text{Mum} \rightarrow \text{Dtr}_1 \text{ Dtr}_2 \dots \text{Dtr}_j \bullet \text{Dtr}_{j+1} \dots \text{Dtr}_n; \text{WOCs} \rangle$

Then when Dtr_j is found, the parser does *not* immediately move the dot. At this point the following procedure, which checks the WOC constraints, is called.

```

procedure CHECK-WOCS
for each WOC such that WOC ∈ WOCS
  if either one of WOC's right-hand side and
  left-hand side is  $\text{Dtr}_j$  then
    if WOC's other operand is included in
    the found categories  $\{\text{Dtr}_1, \dots, \text{Dtr}_i\}$ 
    ( $\text{Call it Dtr}_k$ ), then
      ENFORCE CHECK for  $\text{Dtr}_j$  and  $\text{Dtr}_k$ 
    else do nothing
  else do nothing

```

where ENFORCE CHECK is a simple list operation. This picks the bitmasks of the daughters in question and enforce either ADJ or LP constraint. For example, if the daughters' bitmasks are $[-, 1, 1, -, -, -]$, $[-, -, -, -, 1, 1]$ (where 1 represents occupied positions), the first daughter precedes the second, therefore an LP check will succeed (naturally, if the bitmasks are $[-, -, -, -, 1, 1]$ and $[-, 1, 1, -, -, -]$, conversely, it fails). If it was an adjacency test for the same bitmasks, it fails, while it would succeed for $[-, -, 1, 1, -, -]$, $[-, -, -, -, 1, 1]$, and so on. The failure here would prevent the dot move – merging of bitmasks – from taking place.

Thus, edges that would violate the word order constraints would not be created, thereby preventing wasteful search. This is the same feature as Daniels and Meurers' algorithm, and therefore the efficiency would also be near identical.⁴ The main difference is that we use the information in

⁴The number of edges required is identical. I say ‘near’ because the way of checking WOCs is different. Given the fact that my procedure searches the list of WOCs and found categories in the edge dynamically it will probably take a little bit more space and time, though the difference should be practically negligible.

the feature structure of the daughter categories without resorting to an intermediary rule component.

2.3 Implementation Particulars

I have coded the HPSG feature structure in the way described in Section 1.2, using ProFIT (Erbach, 1995). Specifically this involves specifying the type and sub-features for our new feature, WOC, incorporating the feature structure into the lexical head type. The PHON feature is modified into a compound structure and the WOC-percolation mechanism is incorporated into the Head-Complement Schema. I created small grammars for Japanese and German, with particular focus on linearisation-heavy constructions such as control verbs, into the WOC of which appropriate constraints are written.

The algorithm outlined above has been implemented in Prolog. It is a head-corner, bottom-up chart parser, the codes of which are based upon those by Gazdar and Mellish (1989). The modification consists firstly of reflecting Reape's permutation complete parsing, namely introducing bitmasks for edges, then of the addition of the word order checking procedure described above. The procedure is called every time a dot move is about to take place for an active edge.

I put the grammars to the parser and confirmed the constraints enforced. I also compared the number of edges this parser generates with that of the parser I wrote earlier that employs Daniels and Meurers' algorithm, and the numbers they generated were exactly the same.

3 Conclusion and Future Evaluation

In the above I have been outlining the parsing algorithm for lexicalist linearisation grammar with a brief description of the grammar formalism employed. While based upon the overgenerating, discontinuity-capable non-CFG parsing at default, the algorithm involves a procedure that dynamically checks, each time the dot moves in an edge, the compliance in terms of word order, limiting the search space thereby. The word order information is transferred from the WOC feature of lexical heads into edges: as the parser is head-

corner, this information, if any, is always found when an edge is formed.

Though some initial tests on generative capacity have been carried out on the implementation of our algorithm, full evaluation has yet to be conducted on its computational properties and performance either in theoretical and practical terms.

In practical terms, with a view to using our parser for a large-scale system, an evaluation of speed on an actual machine would be crucial. Particularly important parameters are the kinds and amount of constraints and the length and complexity of the sentences to be parsed. Systematic testing is required by changing these parameters. The size of the grammar, in our case mainly lexicon, is another factor that can affect processing. We need to test whether an increase in the size of the grammar will lead to significant slowing of speed.

Theoretically, we need to identify the computational complexity of parsing with the grammar we have. Though it is context-sensitive and requires exponential complexity *without* WOC constraints, it is now required to ascertain how much we can restrict the search space by means of WOC constraints, or in other words, how 'mild' the context sensitivity required can be. It is only then that we could identify the exact efficiency of our parsing is. Suhre's (2000) result suggests that it can be NP complete, but this certainly depends on the kinds and amount of constraints we incorporate into the grammar.

References

- C. Chung. 1998. Argument composition and long distance scrambling in Korean. In A. Hinrichs, T. Nakazawa, and A. Kathol, editors, *Complex Predicates in Nonderivational Syntax*. Academic Press.
- M. Covington. 1994. Discontinuous dependency parsing of fixed and free word order. Technical report, Artificial Intelligence Program, University of Georgia.
- M. Daniels and D. Meurers. 2004. GIDL: A grammar format for linearization-based HPSG. In *Proceedings of the HPSG04 Conference*.

- G. Erbach. 1995. ProFIT: Prolog with features, inheritance and templates. *Proceedings of the Seventh Conference of the European Association for Computational Linguistics*.
- G. Gazdar and C. Mellish. 1989. *Natural Language Processing in Prolog*. Addison Wesley.
- T. Götz and G. Penn. 1997. A proposed linear specification language. Technical report, University of Tübingen. Arbeitspapiere des SFB 340.
- E. Hinrichs and T. Nakazawa. 1990. Subcategorization and VP structure in German. In S. Hughes et al., editor, *Proceedings of the Third Symposium on Germanic Linguistics*.
- M. Johnson. 1985. Parsing with discontinuous constituents. In *Proceedings of the 23rd Annual Meeting of ACL*, pages 127–132.
- A.K. Joshi. 1985. How much context-sensitivity is necessary for characterizing structural descriptions? In Karttunen L. Dowty, D. and A. Zwicky, editors, *Natural Language Processing – Theoretical, Computational and Psychological Perspectives*. CUP.
- A. Kathol. 2000. *Linear Syntax*. OUP.
- S. Manandhar. 1995. Deterministic consistency checking of LP constraints. In *Seventh Conference of the European Chapter of the Association for Computational Linguistics*, pages 165–172.
- C. Manning, I. Sag, and M. Iida. 1999. The lexical integrity of Japanese causatives. In R. Levine and G. Green, editors, *Studies in Contemporary Phrase Structure Grammar*. CUP.
- F. Morawietz. 1995. Formalization and parsing of typed unification-based ID/LP grammars. Technical Report 68, Universität Tübingen.
- M. Reape. 1991. Parsing bounded discontinuous constituents: Generalisation of some common algorithms. *DIANA Report, Edinburgh University*.
- M. Reape. 1993. *A Formal Theory of Word Order*. Ph.D. thesis, Edinburgh University.
- M. Reape. 1994. Domain union and word order variation in German. In J. Nerbonne et al., editor, *German in Head-Driven Phrase Structure Grammar*.
- M. Saito. 1985. *Some Asymmetries in Japanese and their Theoretical Consequences*. Ph.D. thesis, MIT.
- Y. Sato. 2006. Lexicalising word order constraints for implemented linearisation grammar. In *Proceedings of European Association of Computational Linguistics 2006 Student Session*.
- Y. Sato. forthcoming a. Two alternatives for lexicalist linearisation grammar: Locality Principle revisited.
- S. Shieber. 1985. Evidence against the context freeness of natural languages. *Linguistics and Philosophy*, 8(3):333–43.
- O. Suhre. 2000. Computational aspects of a grammar formalism for languages with freer word order. Diplomarbeit, Tübingen Univ.
- G. van Noord. 1997. An efficient implementation of the head-corner parser. *Computational Linguistics*.
- S. Yatabe. 1996. Long-distance scrambling via partial compaction. In M. Koizumi et al., editor, *Formal Approaches to Japanese Linguistics 2*. MIT Press, Cambridge, Mass.