Workshop TAG+5, Paris, 24-27 May 2000

Complexity of Linear Order Computation in Performance Grammar, TAG and HPSG

Karin Harbusch
Computer Science Dept., Univ. Koblenz
Rheinau 1, D-56075 Koblenz/DE
harbusch@informatik.uni-koblenz.de

Gerard Kempen
Dept. of Psychology, Leiden Univ.
PO Box 9555, NL-2300 RB Leiden/NL
kempen@fsw.leidenuniv.nl

Abstract *This paper investigates the time and space complexity of word order computation in the psycholinguistically motivated grammar formalism of* Performance Grammar *(PG). In PG, the first stage of syntax assembly yields an unordered tree ('mobile') consisting of a hierarchy of lexical frames (lexically anchored elementary trees). Associated with each lexical frame is a* linearizer*—a Finite-State Automaton that locally computes the left-to-right order of the branches of the frame. Linearization takes place after the* promotion *component may have raised certain constituents (e.g. Wh- or focused phrases) into the domain of lexical frames higher up in the syntactic mobile. We show that the worst-case time and space complexity of analyzing input strings of length* n *is O(*n^5*) and O(*n^4*), respectively. This result compares favorably with the time complexity of word-order computations in Tree Adjoining Grammar (TAG). A comparison with Head-Driven Phrase Structure Grammar (HPSG) reveals that PG yields a more declarative linearization method, provided that the FSA is rewritten as an equivalent regular expression.*

1. Performance Grammar

Performance Grammar (PG; Kempen, 1999) is a psycholinguistically motivated grammar formalism for analysis and generation. Somewhat simplified, and in the terminology of TAGs (cf. Joshi & Schabes, 1997), PG defines lexically anchored initial trees and generates derived trees synchronously linked to conceptual structures described in the same formalism (as in Synchronous TAGs; Shieber & Schabes, 1990) and it factors dominance relationships and linear precedence in surface structure trees (Joshi, 1987). PG differs from recent TAG versions in that there are no auxiliary trees, and that adjunction is replaced by a combination of substitution—the only composition operation—and finite-state linearizers that take care of vertical movement ('*promotion*') of phrases and of the linear order of branches of derived trees.

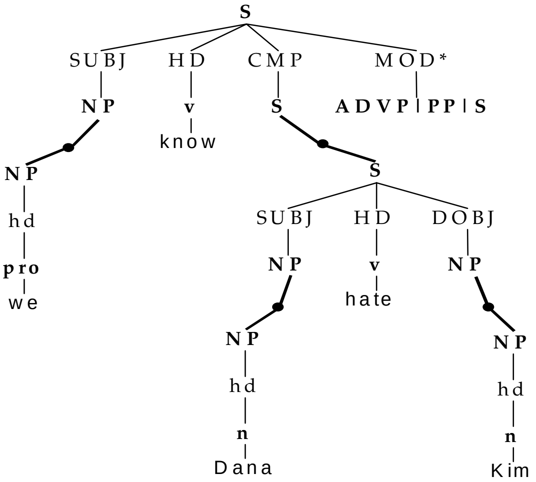

Fig. 1. Simplified lexical frames underlying the sentences We know Dana hates Kim *and* Kim we know Dana hates *(example from Sag & Wasow, 1999). Order of branches is arbitrary. The lines containing filled circles denote substitution (feature unification).*

More precisely, PG's initial trees, called *lexical frames*, are 4-tiered mobiles. The top layer of a frame consists of a single *phrasal node* (called the 'root'; e.g. S, NP, ADJP, PP), which is connected to one or more *functional nodes* in the second layer (e.g., SUBJect, HeaD, Direct OBJect, CoMPlement, MODifier). At most one exemplar of a functional node is allowed in the same frame, except for MOD nodes, which may occur several times (cf. the Kleene star: MOD*). Every functional node dominates exactly one phrasal node ('foot') in the third layer, except for HD which immediately dominates a lexical (part-of-speech) node. Each lexical frame is 'anchored' to exactly one lexical item: a 'lemma' printed in the fourth layer below the lexical node serving as the frame's HeaD (Fig. 1).

Associated with nodes in the first and the third layer are *feature matrices* (not discussed any further here), which can be *unified* with other matrices as part of the substitution process. The unification operation is non-recursive and always involves one root and one foot node of two different lexical frames (see the filled circles in Fig. 1). Only local information can prevent a substitution. No feature information is percolated through the derived tree.

Left-to-right order of the branches of a lexical frame is determined by the *'linearizer'* associated with a lexical frame. We assume that every lexical frame has a one-dimensional array specifying a fixed number of positions (slots, ‘landing sites’) for constituents. For instance, verb frames (i.e., frames anchored to a verb) have an array whose positions can be occupied by a Subject NP, a Direct Object NP, the Head verb, etc. Fig. 2 shows the 12 slots where constituents of English verb frames can go. The positions numbered F1 through F3 make up the Forefield (from Ger. *Vorfeld*) M1 through M7 belong to the Midfield (*Mittelfeld*); B1 and B2 are the Backfield (*Nachfeld*). The annotations at the arcs denote possible fillers of the slots. For example, slot F1 can be occupied by one constituent: either a focus carrying constituents (in Main clauses only), a subordinating conjunction (in an adverbial MODifier clause), a Wh-phrase 'promoted' out of a lower lexical frame (see below), or a non-promoted Wh-phrase. The HeaD verb of a clause is assigned the first Midfield slot (M1), possibly preceded by the complementizer *to* and followed by a particle. Lexical frames anchored to other parts of speech than verbs (e.g. NP- or PP-frames) have their own specialized linearization arrays.

A key property of linearization in PG is that certain constituents may move out of their 'own' array and get 'promoted' to a position in an array located at a higher level in the hierarchy of lexical frames. *Promotion* takes place when, due to subcategorization constraints, a linearization array is ‘truncated’, that is, instantiated incompletely. For instance, if a verb takes a non-finite complement clause, the whole Forefield (slots F1 through F3) will be missing from the complement's array. Due to incomplete instantiation of the linearization array of a lexical frame, one or more constituents of that lexical frame may be deprived of its landing site. In that case, these constituents move up the hierarchy of lexical frames, looking for an instantiation of their landing site in a higher array. The first (i.e. lowest) landing site is always chosen as the final destination.

Truncation of linearization arrays only affects *lateral* (i.e. left- or right-peripheral) slots. The slot occupied by the head of the phrase is never truncated away, which implies that the head of a lexical frame is never promoted. How many slots at either side of the head are actually instantiated, is determined strictly locally, i.e. depends only on information contained by the lexical frame the array belongs to, and its parent frame (its unification partner).

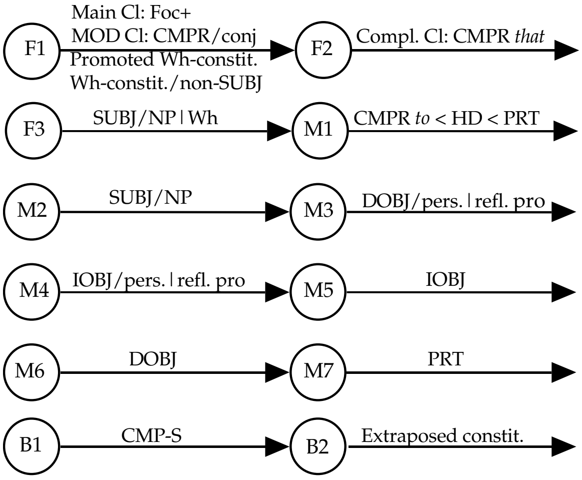

Fig. 2. Linearization array for constituents of S-frames. Placement conditions are annotated on the arcs. E.g., „SUBJ/NP|Wh“ at slot F3 means: SUBJect, provided it is an NP or a Wh-phrase; „<“ indicates the precedence relation between constituents sharing a slot. MODifiers have not been depicted.

The mechanism controlling the distribution of constituents over the instantiated slots of a linearization array, is modeled as a *Finite-State Automaton* (FSA). The FSA associated with a lexical frame traverses the instantiated slots of its array from left to right. At each slot, it inspects the set of constituents that are waiting for placement in the array, and inserts there any constituents meeting the placement conditions (arc labels in Fig. 2).

Fig. 3 illustrates promotion of a focused Direct Object. Examples (1)-(4), taken from Haegeman (1994), demonstrate some subtle consequences of PG's word ordering scheme for *Wh*-questions[1].

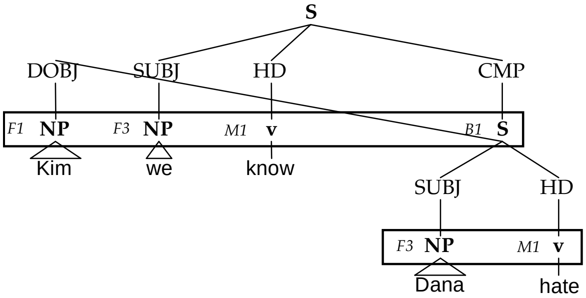

Figure 3. Promotion/linearization at work for the sentence Kim we know Dana hates. *The Direct OBJect of hates carries focus and therefore needs an F1 slot as landing site. Because the linearization array of* hates *has been instantiated incompletely,* Kim *is promoted into the array of the main clause.*

(1) Who do you think left?
[**S**[*F1* Who *M1* do *M2* you *B1*
[**S**[*M1* think *B1*
[**S**[~~*F1 F2 F3*~~ *M1* left]]]]]]

(2) *Who do you think that left?
[**S**[*F1* Who *M1* do *M2* you *B1*
[**S**[*M1* think *B1*
[**S**[~~*F1*~~ *F2* that *F3 M1* left]]]]]]

(3) Who do you think Bill saw?
[**S**[*F1* Who *M1* do *M2* you *B1*
[**S**[*M1* think *B1*
[**S**[~~*F1 F2*~~ *F3* Bill *M1* saw]]]]]]

(4) Who do you think that Bill saw?
[**S**[*F1* Who *M1* do *M2* you *B1*
[**S**[*M1* think *B1*
[**S**[~~*F1*~~ *F2* that *F3* Bill *M1* saw]]]]]]

As outlined in Kempen & Harbusch (1998) and Kempen (1999), the PG's word ordering scheme enables generating the mildly context-sensitive language $a^nb^nc^n$, as well as to account for the movement and word order patterns in English, German and Dutch, including certain rather complicated scrambling phenomena in German. The complexity of these phenomena in contrast with the relative simplicity of this scheme suggests that PG may give rise to very efficient methods of analyzing linear order. Below we show that the worst-case time and space complexity is $O(n^5)$ and $O(n^4)$, respectively.

2. Time and Space Complexity

Consider input string w=$w_1,...,w_n$ of length n. The overall analysis is divided into two steps:

1. Enumerating the complete set of lexical frame hierarchies dominating all permutations of *w* (henceforth called the *set of dominance structures*), and
2. Checking linear order on the basis of the FSA, taking into account the possibility of promotion of phrases in valid dominance structures.

Step 1. Any lexical frame is rewritable in terms of a context-free rule because the functional nodes in the second layer of a lexical frame can be viewed simply as annotations on edges descending from the root node. Every word in *w* is associated with one or (in case of word-class ambiguity) several (O(1)) lexical frames, and every lexical frame has exactly one lexical anchor.

Since a lexical frame is an unordered tree, it can be viewed as an Immediate Dominance rule with an empty set of Linear Precedence rules (ID/LP); and parsing with lexical frames could proceed as outlined in Shieber (1984). However, this method would not take the full set of valid dominance structures into account. For instance, the sentence *Kim we know Dana hates* cannot be analyzed by an ID/LP grammar because *Kim* has moved outside the locality scope of *hate*.

Therefore we follow an indirect course. We interpret the input string as a multiset, i.e. as the set of all permutations of input words, so

[1] Our promotion scheme differs from the 'lifting' scheme recently proposed by Kahane, Nasr & Rambow (1998) in that we allow promotion exclusively along *lateral* (i.e. truncated) regions of a linearization array (thus ruling out, e.g., the promotion pattern in example (2) above). Lifting does not seem to embody an non-ad-hoc equivalent restriction.

that any scope of locality is included. Moreover, we 'freeze' the lexical frames into an arbitrary but fixed left-to-right order of branches, which gives a context-free grammar[2]. This guarantees, for instance, that the valid dominance structure is built for the example in Fig. 3 (as one of the permutations of *We know Dana hates Kim).* Hence, the first step enumerates all locality domains[3].

In order to deal efficiently with multisets in the input, we use a slightly extended version of Earley parsing which overgenerates with respect to repetitions of the same input symbol. The reason is that we do not check here whether any symbol occurs more than once.

First, a subgrammar G' is constructed which only provides the lexical frames of any input symbol w_i, $i=1,...,n$. The only modification of the Earley algorithm concerns the *scanning* step. Instead of exploiting only the items (X, $\alpha\bullet t\beta$) where $t=w_{i+1}$ in the original input string, the parser scans all items and produces (X, $\alpha t\bullet\beta$) according to subgrammar G'. Obviously, this modification performs as bad as ordinary scanning does in the worst case, without introducing additional time and space requirements are introduced. Moreover, the modified scanning method implies that all permutations of the input string are explored. Consequently, given the extended Earley algorithm for subgrammar G', the time complexity and the space complexity for the construction of all dominance structures of the multiset of w remains O(n^3) time and O(n^2) space units[4].

Step 2 is based on the linearizer FSAs and linearization arrays associated with the phrases ('items') in the dominance structures. An array represents a hypothetical order of the input elements $w_i...w_n$ under the assumption that the input elements $w_1...w_{i-1}$ have been ordered successfully. These orderings are licensed by the finite number of slots in the FSA. As the grammar is in Greibach normal form, one ordered symbol must equal the terminal in the rule. All other symbols may go to the finite set of *promotion sites* provided by the FSA. Therefore, the task of step 2 can be reformulated as follows: For any derivation, compute all bijective functions from the terminals in the context-free rules to the input symbols.

In order to deal efficiently with the O(n^2) items that are provided as input to step 2, ordinary Earley processing is assumed along the backpointers inserted in step 1. Initially, this yields all items of the form (S, $tX_1...X_k\bullet$) in I_n. These O(n) items have successfully passed step 1. Now, each of these items is associated with arrays each representing one of the following hypotheses:

$t=w_1$ and no landing site is selected, or

$t=w_k$ and the sequence $w_1,...,w_{k-1}$ of promoted symbols is licensed by a sequence of landing sites to the left of w_k according to the item's FSA ($k=2, ...,n$).

Exploring the number of resulting items, we have to consider O(n) context-free rules in I_n. Moreover, the order in the original input string determines a finite sequence of landing sites according to the currently considered FSA (w_1, w_1w_2, $w_1w_2w_3$, ..., $w_1w_2w_3...w_n$). Hence, the space is O(n^2). With each array we associate a pair containing (1) the '*list of promoted symbols' LPS* and (2) the '*fixed-order marker' FOM* which provides the index in w that t takes. Notice the length of LPS≤n; |FOM|=1.

Now, all substructures of these items (completions) are evaluated, taking the already analyzed input symbols into account (this index with respect to w is provided by the FOM). Hence, the context-free rules applied here can only order O(n-1) times O(n-1) symbols. In general, assuming FOM=i, there exist O(n-i+1) times O(n-i+1) potential orders for the remaining elements $w_{i+1},...,w_n$.

[2] Without loss of generality, we assume that the leftmost branch contains the head of the frame. Hence we deploy a context-free grammar in *Greibach normal form:* (X, t Y_1 ... Y_k), with X and Y_1 ...Y_k non-terminal, and t terminal.

[3] Throughout the paper we assume a condensed representation of the set of potential dominance structures; cf. 'items' in Earley parsing (Earley, 1970).

[4] Since the unification operation in PG is non-recursive, it only involves testing a finite list of constraints. Hence, it does not increase time complexity.

Hence, the overall space complexity is O(n^4).

Now consider the general case for an item (X,$t_j\beta\bullet\mathbf{Y}\gamma$) in I_j with LPS=$a_1,...,a_p$, and FOM=i ($p<i$; $j\leq i$; β, γ, δ possibly empty sequences of non-terminals; $a_1,...,a_p=w_1...w_{i-1}$ with missing elements):

For any item (Y,$t_{j+1}\delta\bullet$) in I_{j+1} the following hypotheses are generated: $w_{i+1}...w_{i+k-1}$ is licensed by a sequence of landing sites to the left of w_{i+k} according to the local FSA. Furthermore, one of the following situations holds: $t_{j+1}=w_{i+k}$ or $t_{j+1} \in$ LPS. Consequently, FOM=$i+k$, LPS=LPS+$w_{i+1},...,w_{i+k-1}$. If $t_{j+1} \in$ LPS, the rightmost w_{i+1} in LPS is erased without loss of generality[5].

If we assume that all items are revisited according to their backpointers, an ordinary Earley parser is capable of performing step 2. (Initially, LPS=nil and FOM=0; finally, an item (S,$\alpha\bullet$) with LPS=nil, FOM=n must exist.) Hence, the input of size O(n^2)—the output of step 1—leads to an overall time complexity of O(n^2) times O(n^3), i.e. O(n^5).

Because this result compares favorably with other grammar formalisms (see below), we conclude that PG provides an efficient method for linear order computation. This advantage derives basically from the deployment of the promotion/linearization scheme, which allows for non-local ordering effects of local ordering decisions, in particular the partial instantiation of linearizers.

3. PG, TAG, and HPSG

For reasons of space we only address the two broadly applied formalisms of *Tree Adjoining Grammar* (TAG, cf. Joshi & Schabes, 1997; and *Head-Driven Phrase Structure Grammar* (HPSG, cf. Sag & Wasow, 1999).

For TAGs, various definitions of dominance and linear order have been proposed in the literature (cf. Joshi, 1987; Vijay-Shanker, 1992 for the definition of quasi-trees; Rambow, 1994 for V-TAGs). They all have in common that long-distance movements are structurally realized by adjoining, thus yielding the extended domains of locality characteristic of all TAGs.

Linear ordering in *Local Dominance/(Tree) Linear Precedence (LD/(T)LP) TAGs* proceeds very much like the ID/LP framework defined for context-free grammars. Since local dominance structures are provided where ‘moved’ constituents feature at the structural level (i.e. adjoining stretches the distance between nodes of the same elementary tree), the cost of linear ordering is at least O(n^6) time units—as for ordinary TAGs (cf. Joshi & Schabes, 1997).

As is well-known, scrambling cannot be described by a simple (LD/(T)LP) TAG. *Quasi-trees* represent partial descriptions of trees. This definition allows for underspecified ordering of moved elements. Loosely speaking, in this framework the spine for promotion is specified declaratively. Similarly, *V-TAGs* (a specific kind of Multi-component TAG) provide a method for manipulating different portions of the same overall derivation tree. Both formalisms are able to handle scrambling phenomena. However, the individual readings are spelled out as different derived trees which are computed on the basis of adjoining in an ordinary TAG parser; hence, this costs at least O(n^6) time units.

The essential difference between the PG and TAG formalisms can be summarized as follows. In both PG and TAG, dominance structures—consisting of lexical frames and elementary trees, respectively—describe linguistically motivated domains of locality. In TAG, the adjoining operation which moves constituents apart, affects the dominance structure. In PG, the linearization component leaves the dominance structure intact. The linearizer FSA associated with lexical frames can accommodate constituents originating from other constituents—a behavior that is less costly, as shown above.

In HPSG (Sag & Wasow, 1999), the *PHON* and *GAP* features, the *GAP principle* and the *argument realization principle* are basically responsible for word ordering and long-distance movement. The *PHON* feature of phrasal types enumerates the linear order on

[5] This reflects the linguistic observation that a promoted phrase chooses the lowest possible landing site.

the basis of list addition (⊕, i.e. a non-commutative sum). Furthermore, movement phenomena are handled by the *GAP feature*, the *GAP principle* and the *argument realization principle*. The *GAP feature* contains a list of elements to be moved. The *argument realization principle*, which says that a word structure tree is well-formed only if the valence lists (SPR and COMPS) add up to the argument structure (ARG-ST), is extended to instantiate gaps freely; i.e. some elements of ARG-ST are neither on the SPR nor on the COMPS list, but on the GAP list instead. The *GAP principle* tests whether the GAP values of all daughters add up to be the GAP value of the mother, unless the rule sanctioning the structure is the Head-Filler Rule. In order to ultimately get all gaps filled, the initial symbol must have an empty GAP list.

This method, like PG's linearization scheme, computes linear order without manipulating the dominance structure (i.e., the daughters' feature descriptions). Loosely speaking, the specification in the PHON feature can be interpreted as a regular expression equivalent to a FSA (although the PHON feature does not provide the definition of the Kleene Star; the infinity of licensed orderings is provided by the recursive application of schemata, i.e. A⊕B, where B has the PHON feature C⊕D—cf. Sag & Wasow, *o.c.*, p. 374). Furthermore, the realization of movement phenomena corresponds directly to promotion, i.e., the gap is percolated along the spine. The definition of landing sites is defined differently, however. In PG, landing sites are enumerated declaratively whereas HPSG terminates the percolation procedurally in terms of the GAP principle. As computation of feature specifications is, in general, NP-complete (Hegner, 1995), the cost of linear order computation is of no particular interest to HPSG. However, HPSG aims at describing linguistic phenomena *declarativel*y. Our description, we claim, is more declarative than the current HPSG realization. The linearizer FSA of a lexical frame can be rewritten as an equivalent regular expression and becomes associated with the referring phrasal type in HPSG.

4. Conclusions

We have described an approach to linear ordering that involves a non-local precedence mechanism which does not rely on a definition and scope of movement as in terms of the GAP feature. In comparison to TAG's structural representations based on adjoining, PG's promotion/linearization yields a more efficient analysis. Compared to HPSG, it can give rise to more declarative word ordering.

References

EARLEY, J. (1970). An Efficient Context-free Parsing Algorithm. *Comm. of the ACM, 13:2.*

HAEGEMAN, L. (1994). *Introduction to Government and Binding Theory (2nd ed.).* Oxford: Blackwell.

HEGNER, S. (1995). *Distributivity in Incompletely Specified Type Hierarchies: Theory and Computational Complexity*. University Tübingen, Linguistics Dept., Tech. Report 4.

JOSHI, A.K. (1987). The Relevance of Tree Adjoining Grammar to Generation. In: Kempen, G. (Ed.), *Natural language generation.* Dordrecht: Kluwer.

JOSHI, A.K., SCHABES, Y. (1997). Tree Adjoining Grammars. In: G. Rozenberg, A. Salomaa (Eds.), *Handbook of Formal Languages (Vol. 3).* Berlin: Springer.

KAHANE, S., NASR, A. & RAMBOW, O. (1998). Pseudo-projectivity: a polynomially parsable non-projective dependency grammar. *Procs. of COLING-ACL*, Montreal.

KEMPEN, G. (1999). Human Grammatical Coding. Ms. Leiden University.

KEMPEN, G., HARBUSCH, K. (1998). A 'Tree Adjoining' Grammar without Adjoining. In: *Procs. of TAG+4*, Philadelphia PA.

RAMBOW, O. (1994). *Formal and Computational Aspects of Natural Language Syntax*. Ph.D. Thesis, University of Pennsylvania.

SAG, I.A., WASOW, T. (1999). *Syntactic Theory: A Formal Introduction*. Stanford: CSLI.

SHIEBER, S.M. (1984). Direct Parsing of ID/LP-Grammars. *Linguistics&Philosophy 7.*

SHIEBER, S.M., SCHABES, Y. (1990). Synchronous Tree-Adjoining Grammars. *Procs. of COLING-90*, Helsinki.

VIJAY-SHANKER, K. (1992). Using Descriptions of Trees in Tree Adjoining Grammars. *Computational Linguistics, 18:4.*