

# GIDL<sub>P</sub>: A Grammar Format for Linearization-Based HPSG

Michael W. Daniels and W. Detmar Meurers

Department of Linguistics  
The Ohio State University  
222 Oxley Hall  
1712 Neil Avenue  
Columbus, OH 43210  
daniels|dm@ling.osu.edu

## 1 Introduction

Within the framework of Head-Driven Phrase Structure Grammar (HPSG), the so-called linearization-based approaches have argued that constraints on word order are best captured within domains that extend beyond the local tree (see, for example, Reape 1993; Kathol 1995; Müller 1999; Donohue and Sag 1999; Bonami et al. 1999).<sup>1</sup> In the HPSG architecture, a theory can consist of any set of constraints on the data structures introduced in the signature; thus, such word order domains and the constraints thereon can be straightforwardly expressed.

On the computational side, most systems for implementing HPSG grammars (like LKB or ALE) employ a parser to efficiently process HPSG-based grammars. To make use of already-existing parsing algorithms, a phrase structure backbone is extracted from the set of constraints encoding an HPSG grammar and used to drive the parsing process. But phrase structure rules encode immediate dominance (ID) and linear precedence (LP) information in local trees, so they cannot directly encode linearization-based HPSG grammars. As a result, any attempt to directly encode linearization-based HPSG grammars must be based on some other grammar format.

The ID/LP grammar format (Gazdar et al. 1985) was introduced to separate immediate dominance from linear precedence, and several proposals

have been made for direct parsing of ID/LP grammars (see, for example, Shieber 1984). Separating LP from ID constraints makes it possible to express word order generalizations across rules. However, the domain in which a word order constraint applies is still the local tree licensed by an ID rule. The ID/LP format thus is not adequate for expressing the word order domains used in linearization-based HPSG approaches.

The LSL grammar format as defined by Suhre (1999) (based on Götz and Penn (1997)) allows elements to be ordered in domains which are larger than a local tree; as a result, categories are not required to cover contiguous strings. Linear precedence constraints, however, remain restricted to local trees: elements that are linearized in a word order domain larger than their local tree cannot be constrained. The approach thus provides valuable worst-case complexity results, but it is inadequate for encoding linearization-based HPSG theories, which crucially rely on the possibility to express linear precedence constraints on all elements within a word order domain.

In sum, no grammar format is currently available that adequately supports the encoding of a processing backbone for linearization-based HPSG grammars. As a result, implementations of linearization-based HPSG grammars have taken one of two options. Some simply do not use a parser, such as the work based on ConTroll (Götz and Meurers 1997); as a consequence, the efficiency and termination properties of parsers cannot be taken for granted in such approaches. The other approaches use a minimal parser that can only take advantage of a small subset of the requi-

---

<sup>1</sup>Apart from Reape's linearization-based HPSG tradition, there have also been proposals for a more complete separation of word order and syntactic structure (see, for example, Richter and Sailer 2001; Penn 1999). In this paper, we focus on the majority of linearization-based HPSG approaches, which follow Reape's architecture.

site constraints (see, for example, Johnson 1985; Reape 1991; van Noord 1991). Such parsers are typically limited to the general concept of resource sensitivity – every element in the input needs to be found exactly once – and the ability to require certain categories to dominate a contiguous segment of the input. The task of constructing a word order domain and enforcing word order constraints in that domain is left out of the parsing algorithm; as a result, constraints on word order either cannot be stated or are tested in a separate clean-up phase.

The purpose of this paper is to define a grammar format which generalizes ID/LP and LSL grammars to allow for a direct encoding of linearization-based HPSG theories in the tradition of Reape (1993). The definition of the Generalized ID/LP (GIDL) grammar format is an important first step towards defining efficient parsing algorithms that can make use of the dominance, precedence, and linearization domain information explicitly encoded in this grammar format.

The abstract first presents the relevant aspects of linearization-based HPSG and then defines the GIDL grammar formalism.<sup>2</sup> For presentation reasons, we will only use atomic categories in this abstract; nothing hinges on this, and when using the formalism to encode linearization-based HPSG grammars, one will naturally use the feature descriptions known from HPSG as categories.

## 2 Linearization-based HPSG

The idea of discontinuous constituency was first introduced into HPSG in a series of papers by Mike Reape (see Reape 1993, and references therein). The core idea is that word order is determined not at the level of the local tree, but at the newly introduced level of an *order domain*, which can include elements from several local trees. We interpret this in the following way: Each terminal has a corresponding order domain; just as constituents combine to form larger constituents, so do their order domains combine to form larger order domains.

Following Reape, a daughter’s order domain enters its mother’s order domain in one of two

<sup>2</sup>Due to space limitations, we focus on introducing the syntax of the grammar formalism and an example; we leave a rigid formal definition of the syntax and semantics for the full version of this paper.

ways. The first possibility, *domain union*, forms the mother’s order domain by shuffling together its daughters’ domains. The second option, *domain compaction*, inserts a daughter’s order domain into its mother’s. Compaction has two effects:

- **Contiguity:** The terminal yield of a compacted category contains all and only the terminal yield of the nodes it dominates; there are no holes or additional strings.
- **LP Locality:** Precedence statements only constrain the order among elements within the same compacted domain. In other words, precedence constraints cannot look into a compacted domain.

Note that these are two distinct functions of domain compaction: defining a domain as covering a contiguous stretch of terminals is in principle independent of defining a domain of elements for LP constraints to apply to. In linearization-based HPSG, domain compaction encodes both aspects.

Later work (Kathol and Pollard 1995; Kathol 1995; Yatabe 1996) introduced the notion of *partial compaction*, in which only a portion of the daughter’s order domain is compacted; the remaining elements are domain unioned.

## 3 Defining the GIDL Grammar Format

To develop a grammar format for linearization-based HPSG, we take the syntax of ID/LP rules and augment it with a means for specifying which daughters form compacted domains. A Generalized ID/LP (GIDL) grammar consists of four parts: a root declaration, a set of lexical entries, a set of grammar rules, and a set of global order constraints. We begin by describing the first three parts, which are reminiscent of context-free grammars (CFGs), and then address order constraints in section 3.1.

**The root declaration** has the form  $root(S, L)$  and states the start symbol  $S$  of the grammar and any linear precedence constraints  $L$  constraining the root domain.

**Lexical entries** have the form  $A \rightarrow t$  and link the pre-terminal  $A$  to the terminal  $t$ , just as in CFGs.

**Grammar rules** have the form  $A \rightarrow \alpha; C$ . They specify that a non-terminal  $A$  immediately dominates a list of non-terminals  $\alpha$  in a domain where a set of order constraints  $C$  holds.

Note that in contrast to CFG rules, the order of the elements in  $\alpha$  does not encode immediate precedence or otherwise contribute to the denotational meaning of the rule. Instead, the order can be used to generalize the head marking used in grammars for head-driven parsing (Kay 1990; van Noord 1991) by additionally ordering the non-head daughters.<sup>3</sup>

If the set of order constraints is empty, we obtain the simplest type of rule, exemplified in (1).

(1)  $S \rightarrow NP, VP$

This rule says that an **S** may immediately dominate an **NP** and a **VP**, with no constraints on the relative ordering of **NP** and **VP**. One may precede the other, the strings they cover may be interleaved, and material dominated by a node dominating **S** can equally be interleaved.

### 3.1 Order Constraints

GIDLp grammars include two types of order constraints: linear precedence constraints and compaction statements.

#### 3.1.1 Linear Precedence Constraints

Linear precedence constraints can be expressed in two contexts: on individual rules (as *rule-level* constraints) and in compaction statements (as *domain-level* constraints). Domain-level constraints can also be specified as *global* order constraints, which has the effect that they are specified for each single domain.

All precedence constraints enforce the following property: given a pair of appropriate elements in the same domain, one must completely precede the other for the resulting parse to be valid. Precedence constraints may optionally require that there

<sup>3</sup>By ordering the right-hand side of a rule so that those categories come first that most restrict the search space, it becomes possible to define a parsing algorithm that makes use of this information. For an example of a construction where ordering the non-head daughters is useful, consider sentences with AcI verbs like *I see him laugh*. Under the typical HPSG analysis (Pollard and Sag 1994), *see* combines in a ternary structure with *him* and *laugh*. Note that the constituent that is appropriate in the place occupied by *him* here can only be determined once one has looked at the other complement, *laugh*, from which it is raised.

be no intervening material between the two elements: this is referred to as immediate precedence. Precedence constraints are notated as follows:

- **Weak precedence:**  $A < B$ .
- **Immediate precedence:**  $A \ll B$ .

A pair of elements is considered appropriate when one element in a domain matches the symbol  $A$ , another matches  $B$ , and neither element dominates the other (it would otherwise be impossible to express an order constraint on a recursive rule).

The symbols  $A$  and  $B$  may be *descriptions* or *tokens*. A constraint involving descriptions applies in any domain in which the described categories occur; it thus can also apply more than once within a given rule or domain. Tokens, on the other hand, can only occur in rule-level constraints and refer to particular RHS members of a rule. In this paper, tokens are represented by numbers referring to the subscripted indices on the RHS categories. A constraint written exclusively with tokens is referred to as *token-based*, otherwise it is called *description-based*.

In (2) we see an example of a rule-level, description-based constraint.

(2)  $A \rightarrow NP_1, V_2, NP_3;$   
 $3 < V$

This constraint specifies that the token 3 in the rule's RHS (the second **NP**) must precede any constituents described as **V** occurring in the same domain.

In sum, LP constraints can be classified along two dimensions: location (rule- or domain-level) and composition (description- or token-based). Domain-level constraints are never token-based.

#### 3.1.2 Compaction Statements

As with LP constraints, compaction statements exist as rule-level and as global order constraints; they cannot, however, occur within other compaction statements. A rule-level compaction statement has the form  $\langle \alpha, A, L \rangle$ , where  $\alpha$  is a list of tokens,  $A$  is the category representing the compacted domain, and  $L$  is a list of domain-level precedence constraints. Such a statement specifies that the constituents referenced in  $\alpha$  form a compacted domain with category  $A$ , inside of which the order

constraints in  $L$  hold. As specified in section 3, a compacted domain must be contiguous (contain all and only the terminal yield of the elements in that domain), and it constitutes a local domain for LP statements.

It is because of partial compaction that the second component  $A$  in a compaction statement is needed. If only one constituent is compacted, the resulting domain will be of the same category; but when multiple categories are fused in partial compaction, the category of the resulting domain needs to be determined so that LP constraints can refer to that domain.

The rule in (3) illustrates compaction: each of the  $S$  categories forms its own domain. In (4) partial compaction is illustrated: the  $V$  and the first  $NP$  form a domain named  $VP$  to the exclusion of the second  $NP$ .

$$(3) S \rightarrow S_1, \text{Conj}_2, S_3; \\ 1 \ll 2, 2 \ll 3, \langle [1], S, [] \rangle, \langle [3], S, [] \rangle$$

$$(4) VP \rightarrow V_1, NP_2, NP_3; \\ \langle [1, 2], VP, [] \rangle$$

One will often compact only a single category without adding domain-specific LP constraints, so we introduce the abbreviatory notation of writing such a compacted category in square brackets. In this way (3) can be written as (5).

$$(5) S \rightarrow [S_1], \text{Conj}_2, [S_3]; \\ 1 \ll 2, 2 \ll 3$$

The formalism also supports *global compaction statements*. A global compaction statement has the form  $\langle A, L \rangle$ , where  $A$  is a description specifying a category that always forms a compacted domain, and  $L$  is a list of domain-level precedence constraints applying to the compacted domain.

## 4 Examples

We start with an example illustrating how a CFG rule is encoded in GIDL format. A CFG rule encodes the fact that each element of the RHS immediately precedes the next, and that the mother category dominates a contiguous string. The context-free rule in (6) is therefore equivalent to the GIDL rule shown in (7).

$$(6) S \rightarrow \text{Nom } V \text{ Acc}$$

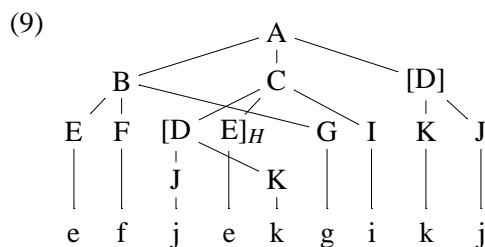
$$(7) [S] \rightarrow V_1, \text{Nom}_2, \text{Acc}_3; \\ 2 \ll 1, 1 \ll 3$$

In (8) we see a more interesting example of a GIDL grammar.

$$(8) \text{ a) root}(A, [] \\ \text{ b) } A \rightarrow B_1, C_2, [D_3]; \\ \quad 2 < 3 \\ \text{ c) } B \rightarrow F_1, G_2, E_3 \\ \text{ d) } C \rightarrow E_1, D_2, I_3; \\ \quad \langle [1,2], H, [] \rangle \\ \text{ e) } D \rightarrow J_1, K_2 \\ \text{ f) Lexical entries: } E \rightarrow e, \dots \\ \text{ g) } E < F$$

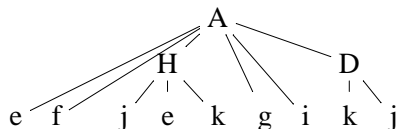
(8a) is the root declaration, stating that an input string must parse as an  $A$ ; the empty list shows that no LP constraints are specifically declared for this domain. (8b) is a grammar rule stating that an  $A$  may immediately dominate a  $B$ , a  $C$ , and a  $D$ ; it further states that the second constituent must precede the third and that the third is a compacted domain. (8c) gives a rule for  $B$ : it dominates an  $F$ , a  $G$ , and an  $E$ , in no particular order. (8d) is the rule for  $C$ , illustrating partial compaction: its first two constituents jointly form a compacted domain, which is given the name  $H$ . (8e) gives the rule for  $D$  and (8f) specifies the lexical entries (here, the preterminals just rewrite to the respective lowercase terminal). Finally, (8g) introduces a global LP constraint requiring an  $E$  to precede an  $F$  whenever both elements occur in the same domain.

Now consider licensing the string **efjkegikj** with the above grammar. The parse tree, recording which rules are applied, is shown in (9). Given that the domains in which word order is determined can be larger than the local trees, we see crossing branches where discontinuous constituents are licensed.



To obtain a representation in which the order domains are represented as local trees again, we can draw a tree with the compacted domains forming the nodes, as shown in (10).

(10)



There are three non-lexical compacted domains in the tree in (9): the root **A**, the compacted **D**, and the partial compaction of **D** and **E** forming the domain **H** within **C**. In each domain, the global LP constraint  $E < F$  must be obeyed. Note that the string is licensed by this grammar even though the second occurrence of **E** does not precede the **F**. This **E** is inside a compacted domain and therefore is not in the same domain as the **F**, so that the LP constraint does not apply to those two elements. This illustrates the property of LP locality: domain compaction acts as a ‘barrier’ to LP application.

The second aspect of domain compaction, contiguity, is also illustrated by the example, in connection with the difference between total and partial compaction. The compaction of **D** specified in (8b) requires that the material it dominates be a contiguous segment of the input. In contrast, the partial compaction of the first two RHS categories in rule (8d) requires that the material dominated by **D** and **E**, taken together, be a continuous segment. This allows the **E** to occur between the two categories dominated by **D**.

Finally, the two tree representations above illustrate the separation of the combinatorial potential of rules (9) from the flatter word order domains (10) which the GIDL format achieves. It would, of course, be possible to write phrase structure rules that license the word order domain tree in (10) directly, but this would amount to replacing a set of general rules with a much greater number of flatter rules corresponding to the set of all possible ways in which the original rules could be combined without introducing domain compaction. Müller (To appear) discusses the combinatorial explosion of rules that results for an analysis of German if one wants to flatten the trees in this way. If recursive rules such as adjunction are included – which is necessary since adjuncts and complements can be freely intermixed in the Ger-

man Mittelfeld – such flattening will not even lead to a finite number of rules.

## 5 Summary

In this abstract, we have introduced a grammar format that can be used as a processing backbone for linearization-based HPSG grammars. It supports the specification of discontinuous constituents and word order constraints on domains which extend beyond the local tree.

We hope that defining this grammar format will stimulate research on the efficient implementation of parsers for linearization-based HPSG grammars and facilitate discussion of the many avenues for optimization that this grammar format provides, given that word order domains and order constraints are made directly available to the parser. GIDL grammars could also be a valuable format for grammar extraction from treebanks or statistical parsing models.<sup>4</sup>

We have developed a prototype parser for the GIDL format, and work is currently progressing both on exploring order constraint compilation techniques to improve efficiency as well as on applying the parser to feature-based grammars.

## References

- Bonami, O., D. Godard, and J.-M. Marandin (1999). Constituency and word order in French subject inversion. In G. Bouma et al. (Ed.), *Constraints and Resources in Natural Language Syntax and Semantics*, pp. 21–40. CSLI.
- Collins, M. (2003). Head-driven statistical models for natural language parsing. *Computational Linguistics* 29(4).
- Donohue, C. and I. A. Sag (1999). Domains in Warlpiri. In *Abstracts of the Sixth Int. Conference on HPSG*, Edinburgh, pp. 101–106. University of Edinburgh.
- Gazdar, G., E. Klein, G. K. Pullum, and I. A. Sag (1985). *Generalized Phrase Structure Grammar*. Harvard UP.
- Götz, T. and W. D. Meurers (1997). The ConTroll system as large grammar development platform. In *Proceedings of the Workshop “Computational Environments for Grammar Development and Linguistic Engineering (EN-VGRAM)” held at ACL/EACL*, Madrid, pp. 38–45.
- Götz, T. and G. Penn (1997). A proposed linear specification language. Volume 134 in *Arbeitspapiere des SFB 340*.
- Johnson, M. (1985). Parsing with discontinuous constituents. In *Proceedings ACL*, Chicago, pp. 127–132.
- Kathol, A. (1995). *Linearization-Based German Syntax*. Ph. D. thesis, The Ohio State University.
- Kathol, A. and C. Pollard (1995). Extraposition via complex domain formation. In *Proceedings of ACL*, pp. 174–180.

<sup>4</sup>Collins (2003, sec. 7.3 and 7.4.1), for example, discusses the excessive number of rules that flat tree annotations such as in the PennTreebank lead to.

- Kay, M. (1990). Head-driven parsing. In M. Tomita (Ed.), *Current Issues in Parsing Technology*. Dordrecht: Kluwer.
- Müller, S. (1999). *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*. Niemeyer.
- Müller, S. (To appear). Continuous or discontinuous constituents? A comparison between syntactic analyses for constituent order and their processing systems. *Research on Language and Computation*.
- Penn, G. (1999). Linearization and WH-extraction in HPSG: Evidence from Serbo-Croatian. In R. D. Borsley and A. Przepiórkowski (Eds.), *Slavic in HPSG*. CSLI.
- Pollard, C. and I. A. Sag (1994). *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press.
- Reape, M. (1991). Parsing bounded discontinuous constituents: Generalisations of some common algorithms. In M. Reape (Ed.), *Word Order in Germanic and Parsing*, pp. 41–70. DYANA R1.1.C, ESPRIT BR 3175.
- Reape, M. (1993). *A Formal Theory of Word Order: A Case Study in West Germanic*. PhD thesis., Univ. of Edinburgh.
- Richter, F. and M. Sailer (2001). On the left periphery of German finite sentences. In D. Meurers and T. Kiss (Eds.), *Constraint-Based Approaches to Germanic Syntax*. CSLI.
- Shieber, S. M. (1984). Direct parsing of ID/LP grammars. *Linguistics and Philosophy* 7, 135–154.
- Suhre, O. (1999). Computational aspects of a grammar formalism for languages with freer word order. Diplomarbeit. (= Volume 154 in Arbeitspapiere des SFB 340, 2000).
- van Noord, G. (1991). Head corner parsing for discontinuous constituency. In *Proceedings of ACL*, pp. 114–121.
- Yatabe, S. (1996). Long-distance scrambling via partial compaction. In M. Koizumi, M. Oishi, and U. Sauerland (Eds.), *Formal Approaches to Japanese Linguistics 2*, pp. 303–317. Cambridge, MA: MITWPL.