

# Proceedings of Formal Grammar 2004

Gerhard Jäger,  
Paola Monachesi, Gerald Penn  
and Shuly Wintner  
(eds.)

August 7/8, 2004  
Nancy



---

# Contents

Preface      v

- 1    **On Learning Discontinuous Dependencies from  
Positive Data**      1  
DENIS BÉCHET, ALEXANDER DIKOVSKY, ANNIE FORET  
AND ERWAN MOREAU
- 2    **Learning Dependency Languages from a Teacher**      17  
JÉRÔME BESOMBES AND JEAN-YVES MARION
- 3    **An integrated approach to French liaison**      29  
OLIVIER BONAMI, GILLES BOYÉ, JESSE TSENG
- 4    **On Induction of Morphology Grammars and its  
Role in Bootstrapping**      47  
DAMIR ČAVAR, JOSHUA HERRING, TOSHIKAZU IKUTA,  
PAUL RODRIGUES, GIANCARLO SCHREMENTI
- 5    **Bidirectional Optimality for Regular Tree  
Languages**      63  
STEPHAN KEPSEK
- 6    **Grammatical Framework and Multiple  
Context-Free Grammars**      77  
PETER LJUNGLÖF
- 7    **Elliptical Constructions and Surface-Based  
Syntax**      91  
STEFAN MÜLLER

- 8 Type-Logical HPSG 107**  
CARL POLLARD
- 9 About Spilled Beans and Shot Breezes 125**  
JAN-PHILIPP SOEHN
- 10 Resumption in Persian Relative Clauses 141**  
MEHRAN TAGHVAIPOUR
- 11 A Hierarchy of Mildly Context-Sensitive  
Dependency Grammars 151**  
ANSSI YLI-JYRÄ AND MATTI NYKÄNEN

---

# Preface

On behalf of the Program Committee for Formal Grammar 2004 we are very pleased to present you with this volume containing the papers accepted for presentation at the Conference on Formal Grammar, held August 7/8, 2004, in Nancy, France. As in previous years, the conference is held in conjunction with the European Summer School in Logic, Language, and Information (ESSLLI2004).

This is the ninth in a series of such conferences, following the initial FG conference in August 1995 in Barcelona. The objective of these meetings is to provide a forum for the presentation of new and original research on formal grammar, with particular regard to the application of formal methods to natural language analysis. Themes of interest include, but are not limited to, formal and computational phonology, morphology, syntax, semantics and pragmatics; model-theoretic and proof-theoretic methods in linguistics; constraint-based and resource-sensitive approaches to grammar; learnability of formal grammar; integration of stochastic and symbolic models of grammar; foundational, methodological and architectural issues in grammar.

As in previous years, the call for papers elicited a fair number of very good submissions. We would like to thank the members of the program committee, listed below, for their time and effort in evaluating these papers.

- Chris Brew (Ohio State)
- Miriam Butt (Konstanz)
- David Chiang (UPenn)
- Tim Fernando (Dublin)
- Philippe de Groote (LORIA, Nancy)
- Mark Hepple (Sheffield)
- Makoto Kanazawa (Tokyo)
- Jonas Kuhn (UTexas at Austin)

- Shalom Lappin (King's College, London)
- Jens Michaelis (Potsdam)
- Guido Minnen (DaimlerChrysler AG)
- Uwe Mönnich (Tübingen)
- Stefan Müller (Bremen)
- Mark-Jan Nederhof (Groningen)
- James Rogers (Earlham College)
- Anoop Sarkar (Simon Fraser University)
- Giorgio Satta (Padua)

We are glad to announce three invited talks by distinguished researchers. Ed Keenan from the University of California at Los Angeles agreed to give a lecture on “Bare Grammar: A New Approach to Language Universals”. Glyn Morrill from the Universitat Politècnica de Catalunya is going to present joint work with Anna Gavarró “On aphasic comprehension and working memory load”. Last but not least, Giorgio Satta from the University of Padua will give a presentation.

We are grateful to CSLI Publications for agreeing to publish a volume with the full papers after the conference. They will appear as Online Proceedings. Nevertheless we decided to make the extended abstracts available during the conference as ESSLLI readers as in previous years.

Last but not least, we are indebted to the local organizers of ESSLLI2004 in Nancy, in particular to Patrick Blackburn and Carlos Areces, for providing us with the necessary logistics, ranging from handling the registration over organizing accommodation to printing these proceedings. We are looking forward to inspiring and enjoyable days in Nancy.

*June 2004*

*Gerhard Jäger, Paola Monachesi, Gerald Penn and Shuly Wintner*

---

# On Learning Discontinuous Dependencies from Positive Data

DENIS BÉCHET, ALEXANDER DIKOVSKY, ANNIE FORET  
AND ERWAN MOREAU

**Abstract** This paper is concerned with learning in the model of Gold the *Categorical Dependency Grammars* (CDG), which express discontinuous (non-projective) dependencies. We show that rigid and  $k$ -valued CDG (without optional and iterative types) are learnable from strings. In fact, we prove that the languages of *dependency nets* coding rigid CDGs have finite elasticity, and we show a learning algorithm. As a standard corollary, this result leads to the learnability of rigid or  $k$ -valued CDGs (without optional and iterative types) from strings.

## 1.1 Introduction

*Dependency grammars* (DGs) are formal grammars assigning *dependency trees* (DT) to generated sentences. A DT is a tree with words as nodes and *dependencies* - i.e. named syntactic relations between words - as arrows. Being very promising from the linguistic point of view (see Mel'cuk (1988)), the DGs have various advantages as formal grammars. Most important is that in terms of dependencies one can naturally encode word order (WO) constraints *independently of syntagmatic relations*. One of the most important WO constraints is that of DT *projectivity*: projections of all words fill continuous<sup>1</sup> intervals. It is widely believed that DTs are a by-product of head selection in

---

<sup>1</sup>So “discontinuous” corresponds to “non-projective” in dependency terms.

constituents. This is evidently false for non-projective DTs because the projections of the heads are always continuous. But even in the projective case the difference between the two syntactic structures is deep. Even if sometimes the DTs defined from heads are isomorphous to the DTs defined directly, the syntactic functions corresponding to individual dependencies (i.e. the corresponding primitive dependency types) are different. The types of dependencies determine the distributivity of a word in a given lexico-grammatical class in a specific role (as the governor, as a subordinate, as the subject, as a direct object in the post-position, pronominalized or not, etc.). This is why, the number of primitive types is noticeably greater for dependency relations than for syntagmatic relations. The lexical ambiguity is greater for DGs, which is compensated by high self-descriptiveness of individual dependency types. This explains the differences in traditional analyses of the same constructions in syntagmatic terms (which are more or less based on X-bar syntax) and in dependency terms.

Many DGs are projective, i.e. define only projective DTs (cf. Gaifman (1961), Sleator and Temperley (1993), Lombardo and Lesmo (1996)). This property drastically lowers complexity of DGs. As it concerns symbolic learning, positive results are known exclusively for projective DGs: Moreau (2001), Besombes and Marion (2001), Bechet (2003). In this paper we obtain the first result of symbolic learnability of non-projective DGs from positive data. It holds for a reach class of DGs introduced recently in Dikovsky (2004). At the same time, we describe some cases of unlearnability from strings of such grammars with optional or iterative types.

## 1.2 Categorical Dependency Grammars

### 1.2.1 Syntactic types

We extend the definition of types in Dikovsky (2004) to *repetitive* and *optional* types as follows.  $\mathbf{C}$  will denote a finite set of elementary categories. Elementary categories may be *iterated* and become *optional*.  $\mathbf{C}^* =_{af} \{C^* \mid C \in \mathbf{C}\}$ ,  $\mathbf{C}^+ =_{af} \{C^+ \mid C \in \mathbf{C}\}$ ,  $\mathbf{C}^? =_{af} \{C^? \mid C \in \mathbf{C}\}$ , will denote respectively the sets of *iterative*, *repetitive* and *optional* categories.  $\mathbf{C}^\omega =_{af} \mathbf{C}^* \cup \mathbf{C}^+ \cup \mathbf{C}^?$ . All categories in  $\mathbf{C}^\omega$  are *neutral*. Besides them there are *polarized* categories of one of four oriented polarities: left and right positive  $\swarrow, \nearrow$  and left and right negative  $\searrow, \swarrow$ . For each polarity  $v$ , there is the unique “dual” polarity  $\tilde{v}$ :  $\swarrow = \tilde{\searrow}$ ,  $\nwarrow = \tilde{\nearrow}$ ,  $\searrow = \tilde{\swarrow}$ ,  $\swarrow = \tilde{\nwarrow}$ . Intuitively, the positive categories can be seen as valencies of the outgoing distant dependencies of governors, and the negative categories as those of the incoming distant dependencies of

subordinate words. So they correspond respectively to the beginnings and the ends of distant dependencies. For instance, the positive valency ( $\searrow pre-UPON-obj$ ) marks the beginning of the distant dependency  $pre-UPON-obj$  of a transitive verb governing a left-dislocated object headed by the preposition ‘UPON’, whereas the end of this dependency: UPON is marked by the dual negative valency ( $\swarrow pre-UPON-obj$ ) (cf. *upon what dependency theory we rely*).

$\nearrow \mathbf{C}$ ,  $\nwarrow \mathbf{C}$ ,  $\searrow \mathbf{C}$  and  $\swarrow \mathbf{C}$  denote the corresponding sets of polarized distant dependency categories. For instance,  $\nearrow \mathbf{C} = \{(\nearrow C) \mid C \in \mathbf{C}\}$  is the set of *right positive* categories.  $V^+(\mathbf{C}) = \nearrow \mathbf{C} \cup \nwarrow \mathbf{C}$  is the set of positive distant dependency categories,  $V^-(\mathbf{C}) = \searrow \mathbf{C} \cup \swarrow \mathbf{C}$  is the set of those negative.

Defining distant dependencies, it is sometimes necessary to express that the subordinate word is the first (last) in the sentence, in the clause, etc., or it immediately precedes (follows) some word. E.g., in French the negative dependency category  $\swarrow clit-dobj$  of a cliticized direct object must be anchored to the auxiliary verb or to the verb in a non-analytic form. For that we will use specially marked *anchored* negative categories:  $Anc(\mathbf{C}) =_{af} \{\#(\alpha) \mid \alpha \in V^-(\mathbf{C})\}$  - our name for negative categories whose position is determined relative to some other category - whereas the negative categories in  $V^-(\mathbf{C})$  will be called *loose*.

**Definition 1** *The set  $Cat(\mathbf{C})$  of categories is the least set verifying the conditions:*

1.  $\mathbf{C} \cup V^-(\mathbf{C}) \cup Anc(\mathbf{C}) \subset Cat(\mathbf{C})$ .
2. For  $C \in Cat(\mathbf{C})$ ,  $A_1 \in (\mathbf{C} \cup \mathbf{C}^\omega \cup Anc(\mathbf{C}) \cup \nwarrow \mathbf{C})$  and  $A_2 \in (\mathbf{C} \cup \mathbf{C}^\omega \cup Anc(\mathbf{C}) \cup \nearrow \mathbf{C})$ , the categories  $[A_1 \setminus C]$  and  $[C / A_2]$  also belong to  $Cat(\mathbf{C})$ .

We suppose that the constructors  $\setminus$ ,  $/$  are associative. So every complex category  $\alpha$  can be presented in the form:

$$\alpha = [L_k \setminus \dots L_1 \setminus C / R_1 \dots / R_m].$$

For instance,  $[\#(\swarrow clit\_dobj) \setminus subj \setminus S / auxPP]$  is one of possible categories of an auxiliary verb, which defines it as the host word for a cliticized direct object, requires the local subject dependency on its left and, on its right, the local dependency *auxPP* with a subordinate.

### 1.2.2 Grammar definition

**Definition 2** *A categorial dependency grammar (CDG) is a system  $G = (W, \mathbf{C}, S, \delta)$ , where  $W$  is a finite set of words,  $\mathbf{C}$  is a finite set of elementary categories containing the selected root category  $S$ , and  $\delta$  - called lexicon - is a finite substitution on  $W$  such that  $\delta(a) \subset Cat(\mathbf{C})$  for each word  $a \in W$ .*

Now we will extend to new types the definitions of the language and DT language generated by a CDG. The language and DT language generated by a CDG are defined using a provability relation  $\vdash$  on strings of categories. The core part of this definition are the rules of polarized dependency valencies control. The idea behind these rules is that in order to establish a distant dependency between two words with dual dependency valencies, the negative valency must be *loose*. The anchored negative valencies can serve only to anchor a distant subordinate to a host word. As soon as the correct position of the subordinate is identified, its valency becomes loose and so available to the governor.

In the definition below we suppose that to each occurrence of a category  $\Gamma_1 C \Gamma_2$  corresponds a DT  $D$  of category  $C$ .  $r(D)$  denotes the root of  $D$ . For space reasons, we present only the rules for left constructors. The rules for right constructors are similar.

**Definition 3** *Provability relation*  $\vdash$ :

**Local dependency rule:**

**L.**  $\Gamma_1 C [C \setminus \alpha] \Gamma_2 \vdash \Gamma_1 \alpha \Gamma_2$ . If  $C$  is the category of  $D_1$  and  $[C \setminus \alpha]$  is that of  $D_2$ , then  $\alpha$  becomes the category of the new DT :  $D_1 \cup D_2 \cup \{r(D_1) \xleftarrow{C} r(D_2)\}$ .

**$\omega$ -dependency rules:**

**I.**  $\Gamma_1 C [C^* \setminus \alpha] \Gamma_2 \vdash \Gamma_1 [C^* \setminus \alpha] \Gamma_2$ . If  $C$  is the category of  $D_1$  and  $[C^* \setminus \alpha]$  is that of  $D_2$ , then  $[C^* \setminus \alpha]$  in the consequence becomes the category of the new DT :  $D_1 \cup D_2 \cup \{r(D_1) \xleftarrow{C} r(D_2)\}$ .

**R.**  $\Gamma_1 C [C^+ \setminus \alpha] \Gamma_2 \vdash \Gamma_1 [C^* \setminus \alpha] \Gamma_2$ . If  $C$  is the category of  $D_1$  and  $[C^+ \setminus \alpha]$  is that of  $D_2$ , then  $[C^* \setminus \alpha]$  becomes the category of the new DT :  $D_1 \cup D_2 \cup \{r(D_1) \xleftarrow{C} r(D_2)\}$ .

**O.**  $\Gamma_1 C [C^? \setminus \alpha] \Gamma_2 \vdash \Gamma_1 \alpha \Gamma_2$ . If  $C$  is the category of  $D_1$  and  $[C^? \setminus \alpha]$  is that of  $D_2$ , then  $\alpha$  in the consequence becomes the category of the new DT :  $D_1 \cup D_2 \cup \{r(D_1) \xleftarrow{C} r(D_2)\}$ .

**$\Omega$ .**  $\Gamma_1 [C \setminus \alpha] \Gamma_2 \vdash \Gamma_1 \alpha \Gamma_2$  for all  $C \in \mathbf{C}^* \cup \mathbf{C}^?^2$ .

**Anchored dependency rule:**

**A.**  $\Gamma_1 \#(\alpha) [\#(\alpha) \setminus \beta] \Gamma_2 \vdash \Gamma_1 \alpha \beta \Gamma_2$ ,  $\#(\alpha) \in \text{Anc}(\mathbf{C})$ .

**Distant dependency rule:**

**D.**  $\Gamma_1 (\swarrow C) \Gamma_2 [(\searrow C) \setminus \alpha] \Gamma_3 \vdash \Gamma_1 \Gamma_2 \alpha \Gamma_3$ .

The rule applies if there are no occurrences of subcategories  $\swarrow C$ ,  $\#(\swarrow C)$  and  $\searrow C$  in  $\Gamma_2$ . If  $\swarrow C$  is the category of  $D_1$  and  $[(\searrow C) \setminus \alpha]$  is that of  $D_2$ , then  $\alpha$  becomes the category of the new DT :  $D_1 \cup D_2 \cup \{r(D_1) \xleftarrow{C} r(D_2)\}$ .

---

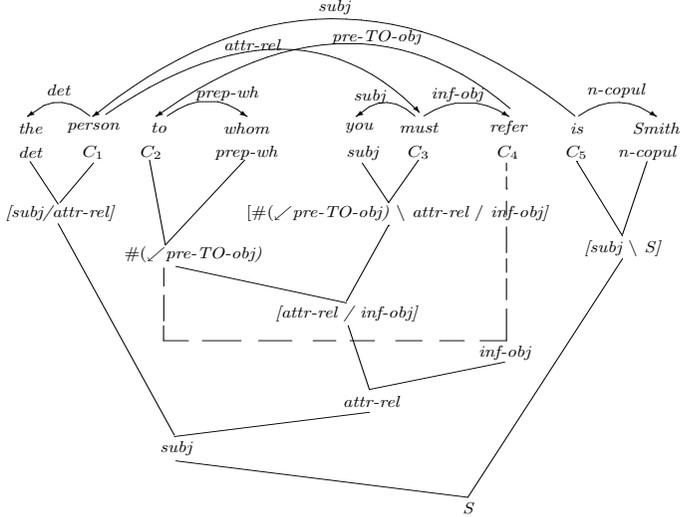
<sup>2</sup>The DTs rest unchanged when no instruction.

$\vdash^*$  denotes the reflexive-transitive closure of  $\vdash$ .

**Definition 4** A DT  $D$  is assigned by a CDG  $G = (W, \mathbf{C}, S, \delta)$  to a sentence  $w$  (denoted  $G(D, w)$ ) if  $D$  is defined as DT of category  $S$  in a proof  $\Gamma \vdash^* S$  for some  $\Gamma \in \delta(w)$ .

The DT-language generated by  $G$  is the set of DTs  $\Delta(G) = \{D \mid \exists w \in W^+ G(D, w)\}$ . The language generated by  $G$  is the set of sentences  $L(G) = \{w \in W^+ \mid \exists D G(D, w)\}$ .

**Example 5 (PP-movement in English from Dikovsky (2004))**



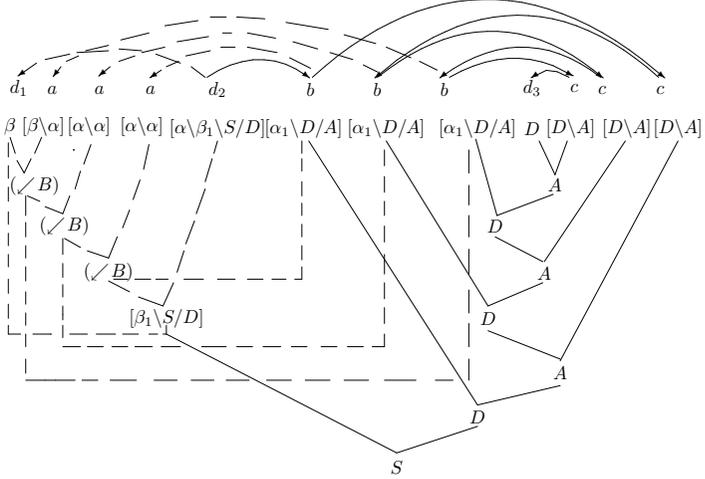
This movement is expressed using the following categories:

- $C_1 = [det \setminus subj / attr-rel] \in \delta(person)$ ,
- $C_2 = \#[\setminus pre-TO-obj] / prep-wh \in \delta(to)$ ,
- $C_3 = [subj \setminus \#[\setminus pre-TO-obj] \setminus attr-rel / inf-obj] \in \delta(must)$ ,
- $C_4 = [\setminus pre-TO-obj \setminus inf-obj] \in \delta(refer)$ ,
- $C_5 = [subj \setminus S / n-copul] \in \delta(is)$ ,
- $det \in \delta(the)$ ,  $prep-wh \in \delta(whom)$ ,  $subj \in \delta(you)$ , and  $n-copul \in \delta(Smith)$ .

The following example cited from Dikovsky (2004) shows that CDGs can generate non-CF languages and are more expressive than dependency grammars generating projective DTs.

**Example 6** Let  $G_0 = (\{a, b, c, d_1, d_2, d_3\}, \mathbf{C}_0, S, \delta_0)$ , where  $\delta_0$  is defined by:

$a \mapsto [\beta \setminus \alpha], [\alpha \setminus \alpha],$   $d_1 \mapsto \alpha,$   
 $b \mapsto [\alpha_1 \setminus D/A],$   $d_2 \mapsto [\alpha \setminus \beta_1 \setminus S/D],$   
 $c \mapsto [D \setminus A],$   $d_3 \mapsto D,$   
 where  $\alpha = \#(\sphericalangle B)$ ,  $\alpha_1 = (\sphericalleftarrow B)$ ,  $\beta = \#(\sphericalangle C)$  and  $\beta_1 = (\sphericalleftarrow C)$ .  
 $L(G_0) = \{d_1 a^n d_2 b^n d_3 c^n \mid n > 0\}$ .  
 A proof of  $d_1 a^3 d_2 b^3 d_3 c^3 \in L(G_0)$ , in which  $\alpha = \#(\sphericalangle B)$ ,  $\alpha_1 = (\sphericalleftarrow B)$ ,  
 $\beta = \#(\sphericalangle C)$  and  $\beta_1 = (\sphericalleftarrow C)$ :

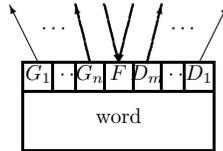


### 1.2.3 Language of (untyped) dependency nets

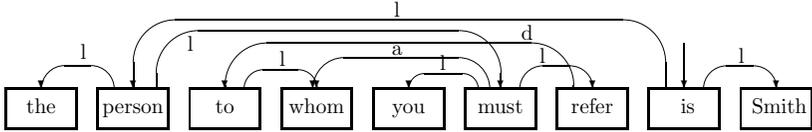
A CDG connects the words of a sentence by oriented edges that correspond to: local dependencies, anchored dependencies, discontinuous dependencies. Thus, the parsing of a sentence can be summarized by a dependency net built from nodes connected by dependencies.

**Definition 7** An untyped node is a list of vertices called **slots** associated to a word. A node is an untyped node where each slot corresponds to the elementary categories of one category of the word.

Here is the node corresponding to  $[G_1 \setminus \dots \setminus G_n \setminus F/D_m / \dots / D_1]$ :



**Definition 8** An (untyped) dependency net is a list of (untyped) nodes connected by local (l), anchored (a) and distant (d) dependencies that correspond to a parsing of the sentence.



One of the slots is not connected and serves as the main conclusion of the dependency net (i.e. the elementary category  $S$ ). It appears on the figure as an arrow without origin that ends on “is”. The dependency tree (DT) corresponding to a dependency net is obtained by erasing anchored dependencies and adding categories on local and distant dependencies (categories appear on nodes in dependency nets and on edges in dependency trees).

**Definition 9** An (untyped) dependency net  $N$  is assigned by a CDG  $G$  if there exists a parsing of the list of words of  $N$  that corresponds to the (untyped) nodes and the dependencies of  $N$ .

The language of dependency nets of  $G$ , denoted  $\mathcal{NL}(G)$  is the set of dependency nets assigned by  $G$ .

The language of untyped dependency nets of  $G$ , denoted  $UN\mathcal{L}(G)$  is the set of untyped dependency nets assigned by  $G$ .

**Definition 10** CDGs that associate at most  $k$  nodes to each symbol are called  $k$ -valued. 1-valued grammars are also called rigid.

### 1.3 Learnability, finite elasticity and limit points

A class of grammars  $\mathcal{G}$  is learnable iff there exists a learning algorithm  $\phi$  from finite sets of words to  $\mathcal{G}$  that converges, for any  $G \in \mathcal{G}$  and for any growing partial enumeration of  $\mathcal{L}(G)$ , to a grammar  $G' \in \mathcal{G}$  such that  $\mathcal{L}(G') = \mathcal{L}(G)$ .

Learnability and unlearnability properties have been widely studied from a theoretical point of view. A very useful property for our purpose is the finite elasticity property of a class of languages. This term was first introduced in Wright (1989), Motoki et al. (1991) and, in fact, it implies learnability. A very nice presentation of this notion can be found in Kanazawa (1998).

**Definition 11 (Finite Elasticity)** A class  $\mathcal{C}$  of languages has infinite elasticity iff  $\exists (e_i)_{i \in \mathbb{N}}$  an infinite sequence of sentences,  $\exists (L_i)_{i \in \mathbb{N}}$  an infinite sequence of languages of  $\mathcal{C}$  such that  $\forall i \in \mathbb{N} : e_i \notin L_i$  and  $\{e_0, \dots, e_{i-1}\} \subseteq L_i$ . A class has finite elasticity iff it has not infinite elasticity.

**Theorem 12 [Wright 1989]** A class that is not learnable has infinite elasticity.

**Corollary 13** *A class that has finite elasticity is learnable.*

Finite elasticity is a very nice property because it can be extended from a class to every class obtained by a *finite-valued relation*<sup>3</sup>. We use here a version of the theorem that has been proved in Kanazawa (1998) and is useful for various kinds of languages (strings, structures, nets) that can be described by lists of elements over some alphabets.

**Theorem 14 [Kanazawa 1998]** *Let  $\mathcal{M}$  be a class of languages over  $\Gamma$  that has finite elasticity, and let  $R \subseteq \Sigma^* \times \Gamma^*$  be a finite-valued relation. Then the class of languages  $\{R^{-1}[M] = \{s \in \Sigma^* \mid \exists u \in M \wedge (s, u) \in R\} \mid M \in \mathcal{M}\}$  has finite elasticity.*

**Definition 15 (Limit Points)** *A class  $\mathcal{C}\mathcal{L}$  of languages has a limit point iff there exists an infinite sequence  $(L_n)_{n \in \mathbb{N}}$  of languages in  $\mathcal{C}\mathcal{L}$  and a language  $L \in \mathcal{C}\mathcal{L}$  such that:  $L_0 \subsetneq L_1 \subsetneq \dots \subsetneq L_n \subsetneq \dots$  and  $L = \bigcup_{n \in \mathbb{N}} L_n$  ( $L$  is a limit point of  $\mathcal{C}\mathcal{L}$ ).*

If the languages of the grammars in a class  $\mathcal{G}$  have a limit point then the class  $\mathcal{G}$  is *unlearnable*.<sup>4</sup>

## 1.4 Limit points for CDGs with optional or iterative categories

**Limit point constructions.**

**Definition 16** ( $G_n, G'_n, G_*, G'_*$ ) *Let  $S, A, B$  be three elementary categories. We define by induction:*

$$\begin{array}{ll} C_0 = S & C'_0 = S \\ C_{n+1} = (C_n/A^?) & C'_{n+1} = (C'_n/A^*)/B^* \\ G_0 = \{a \mapsto A, c \mapsto C_0\} & G'_0 = \{a \mapsto A, b \mapsto B, c \mapsto C'_0\} \\ G_n = \{a \mapsto A, c \mapsto [C_n]\} & G'_n = \{a \mapsto A, b \mapsto B, c \mapsto [C'_n]\} \\ G_* = \{a \mapsto [A/A^?], c \mapsto [S/A^?]\} & G'_* = \{a \mapsto A, b \mapsto A, c \mapsto [S/A^*]\} \end{array}$$

These constructions yield two limit points as follows.

**Theorem 17** *We have:*

$$\begin{array}{l} \mathcal{L}(G_n) = \{ca^k \mid k \leq n\} \text{ and } \mathcal{L}(G_*) = c\{a\}^* \\ \mathcal{L}(G'_n) = \{c(b^*a^*)^k \mid k \leq n\} \text{ and } \mathcal{L}(G'_*) = c\{b, a\}^* \end{array}$$

**Corollary 18** *They establish the non-learnability from strings for the underlying classes of (rigid) grammars : those allowing optional categories ( $A^?$ ) and those allowing iterative categories ( $A^*$ ).*

<sup>3</sup>A relation  $R \subseteq \Sigma^* \times \Gamma^*$  is finite-valued iff for every  $s \in \Sigma^*$ , there are at most finitely many  $u \in \Gamma^*$  such that  $(s, u) \in R$ .

<sup>4</sup>This implies that the class has infinite elasticity.

**Proof for optional categories:** Only three rules apply to  $G_n, G_*$  in this case : **L.** (local dependency rule), **O.** and  **$\Omega$ .** ( $\omega$ -dependency rules). These rules enjoy the subformula property.

- $\mathcal{L}(G_0)$  : (1) it clearly contains  $c$  (of category  $S$ ) ; (2) since no rule applies to the elementary categories  $\{A, S\}$ , it contains only  $c$ .
- $\mathcal{L}(G_n)$  ( $n > 0$ ) : we consider  $ca^k$  and denote by  $\Delta_{n,k}$  its category assigned by  $G_n$ . We have :

$$\Delta_{n,0} = [C_n] = \underbrace{[S/A^?/\dots/A^?]}_{n} \vdash^* S \text{ (by } \mathbf{\Omega} \text{ rule, } n \text{ times)}$$

$$\Delta_{n,k} = \underbrace{[S/A^?/\dots/A^?]}_n \underbrace{A\dots A}_k \vdash \underbrace{[S/A^?/\dots/A^?]}_{n-1} \underbrace{A\dots A}_{k-1} \text{ (if } k > 0,$$

by **O.** rule)

(1) By induction, for all  $k \leq n$ ,  $\Delta_{n,k} \vdash^* S$ , that is  $ca^k \in \mathcal{L}(G_n)$  when  $k \leq n$ . (2) Let us consider  $w \in \mathcal{L}(G_n)$  of a category  $\Delta$ .  $w$  cannot start with an  $a$  (a category  $A$  on the left of  $\Delta$  could not disappear, due to the use of right constructors only);  $w$  cannot contain several  $c$ , (no cancelation of  $S$  is possible since none occurs under a constructor) ; thus  $w = ca^k$  for some  $k \geq 0$ . If  $k > n$ , then after one step  $\Delta_{n,k}$  necessarily leads to  $\Delta_{n-1,k-1}$  or  $\Delta_{n-1,k}$  as above : by induction starting from the case  $\mathcal{L}(G_0)$   $k > n$  is thus not possible. Therefore,  $w = ca^k$  with  $k \leq n$ .

- $\mathcal{L}(G_*)$  : (1) it contains  $ca^k$  because of  $[S/A^?] \vdash S$ ,  $[S/A^?][A/A^?] \vdash [S/A^?]A \vdash S$  and  $[S/A^?]\dots[A/A^?][A/A^?] \vdash [S/A^?]\dots[A/A^?]A \vdash [S/A^?]\dots A$   
(2)  $w \in \mathcal{L}(G_*)$  has exactly one  $c$  (at least one to provide  $S$ , and no more as explained above); it cannot start with an  $a$  (otherwise a type part would rest before  $S$ ) ; therefore  $w = ca^k$ . ■

**Proof for iterative categories:** Only three rules apply to  $G'_n, G'_*$  : **L.** (local dependency rule), **I.** and  **$\Omega$ .** ( $\omega$ -dependency rules), all of them enjoying the subformula property.

- $\mathcal{L}(G'_0)$  : (1) it clearly contains  $c$  (category  $S$ ) and (2) only  $c$  since no rule applies to  $\{A, B, S\}$ .
- $\mathcal{L}(G'_n)$  ( $n > 0$ ). We have  $D'_n = C'_{n-1}/A^*$  and  $C'_n = D'_n/B^*$ .  
(1) For  $w \in \{c(b^*a^*)^k \mid k \leq n\}$ , we have  $w \in \mathcal{L}(G'_n)$  by :  $[C'_n]B\Delta \vdash [C'_n]\Delta$  and  $[D'_n]A\Delta \vdash [D'_n]\Delta$  (by **I.** rule) and  $[C'_n] \vdash [D'_n]$  (by  **$\Omega$ .** rule)  
(2) Let  $w' \in \mathcal{L}(G'_n)$ . As above for  $G_n$ ,  $w'$  cannot start with an  $a$  or a  $b$  (right constructors only); and  $w'$  cannot contain several  $c$  (no  $S$  under a constructor) ; thus  $w' = cw''$ , where  $w'' \in \{b, a\}^*$ .

$w' \in \{c(b^*a^*)^k \mid k \leq n\}$  follows by induction on  $n$  and on the length of types  $\Gamma$  words  $w \in \{b, a\}^*$  from the following assertion :

(i) if  $[C'_n]\Gamma \vdash S$ , then  $w \in \{(b^*a^*)^k \mid k \leq n\}$  and (ii) if  $[D'_{n+1}]\Gamma \vdash S$  then  $w \in \{a^*(b^*a^*)^k \mid k \leq n\}$ .

For  $n = 0$ , (i) is clear from  $\mathcal{L}(G'_0) = \{c\}$ . For (ii), if  $\Gamma = B\Gamma'$ , we get the first step with the only possibility of  $[D'_{n+1}]B\Gamma' \vdash [C'_n]B\Gamma'$ . For (ii), if  $\Gamma = A\Gamma'$ , we have two possibilities  $[D'_{n+1}]A\Gamma' \vdash [C'_n]A\Gamma'$  or  $[D'_{n+1}]A\Gamma' \vdash [D'_{n+1}]\Gamma'$ . For (i), if  $\Gamma = A\Gamma'$ , we get the first step with the only possibility of  $[C'_n]A\Gamma' \vdash [D'_n]A\Gamma'$ . For (i), if  $\Gamma = B\Gamma'$ , we have two possibilities  $[C'_n]B\Gamma' \vdash [D'_n]B\Gamma'$  or  $[C'_n]B\Gamma' \vdash [C'_n]\Gamma'$ . This implies (i), (ii) by induction on  $n$  or a shorter type.

- $\mathcal{L}(G'_*)$  : (1) it clearly contains  $c\{b, a\}^*$  using  $[S/A^*]A\Delta \vdash S\Delta$  ( $\mathbf{I}$ . rule) and  $[S/A^*] \vdash S$  ( $\mathbf{\Omega}$ . rule)
- (2)  $w' \in \mathcal{L}(G'_*)$  has exactly one  $c$  (at least one to provide  $S$ , and no more, as explained above for  $G_n$ ); it cannot start with  $a$  (otherwise a type part would rest before  $S$ ). Therefore,  $w' \in c\{b, a\}^*$ . ■

## 1.5 Finite elasticity of rigid $UN\mathcal{L}$

This section is concerned with languages of untyped dependency nets rather than grammars of strings. The following theorem is essential because it implies that the corresponding class of rigid CDG (without optional and iterative categories) has finite elasticity and thus is learnable from strings. This result can also be extended to the class of  $k$ -valued CDG for every  $k$ .

**Theorem 19** *Rigid CDGs define a class of languages of untyped dependency nets that has finite elasticity.*

**Proof:** We use a result of Shinohara (1990, 1991) that proves that *formal systems* that have *finite thickness* have also finite elasticity. In Shinohara (1991) this result is applied to *length-bounded elementary formal system with at most  $k$  rules* and also to *context sensitive languages* that are definable by at most  $k$  rules. Formal systems in Shinohara (1991) describe not only languages of strings but also languages of terms. They can be applied to typed or untyped dependency nets which can be seen as well-bracketed strings (each dependency is associated to an opening and a closing (typed) bracket). For the class of rigid untyped dependency net grammars, a sketch of proof is as follows:

1. **Definition.** A CDG  $G_1 = (W_1, \mathbf{C}, S, \delta_1)$  is *included* in a CDG  $G_2 = (W_2, \mathbf{C}, S, \delta_2)$  (notation  $G_1 \subseteq G_2$ ) iff  $W_1 \subseteq W_2$  and  $\forall x \in W_1, \delta_1(x) \subseteq \delta_2(x)$ .
2. **Definition and lemma.** The mapping  $UN\mathcal{L}$  from CDG to untyped dependency net grammars is monotonic: if  $G_1 \subseteq G_2$  then

$$\text{UNL}(G_1) \subseteq \text{UNL}(G_2).$$

3. **Definition.** A grammar  $G$  is *reduced with respect to a set  $X$  of untyped dependency nets* iff  $X \subseteq \text{UNL}(G)$  and for each grammar  $G' \subseteq G$ ,  $X \not\subseteq \text{UNL}(G')$ . Intuitively, a grammar that is reduced with respect to  $X$  covers all the structures of  $X$  and has no redundant expressions.
4. **Lemma.** For each finite set  $X \subseteq \text{UNL}(G)$ , there is a finite set of rigid untyped dependency net languages that correspond to the grammars that are reduced with respect to  $X$ . This is the main part of the proof. In fact, if a rigid untyped dependency net grammar  $G = (W, \mathbf{C}, S, \delta)$  using types  $Tp$  is reduced with respect to  $X$  then each word that does not appear in one of the untyped dependency net of  $X$  must be associated through  $\delta$  to the empty set. All other words must be associated to exactly one type of  $Tp$  (the grammar is rigid). The left and right numbers of slots are given by the occurrences of the word in the untyped dependency nets and they must be the same for all the occurrences because the language we try to learn corresponds to a *rigid* untyped dependency net grammar. If the sum of the left and right arities of each word in  $X$  is bound by  $m$ , and if  $n$  is the number of words that appear in  $X$ , the number of equivalent grammars<sup>5</sup> is bound by the number of partitions of a set of  $n \times m$  elements.
5. **Definition.** Monotonicity and the previous property define a system that has *bounded finite thickness*.
6. **Theorem.** Shinohara proves in Shinohara (1991) that a formal system that has bounded finite thickness has finite elasticity.
7. **Corollary.** Rigid untyped dependency net languages have finite elasticity.

### A learning algorithm for rigid untyped dependency net grammars

The learning algorithm is based on Buszkowski's original algorithm for rigid (classical) categorial grammars Buszkowski and Penn (1989). Since a rigid grammar  $G$  assigns only one type to each word,  $X \subseteq \text{UNL}(G)$  implies that all occurrences of a word  $w$  appearing in  $X$  must be used with the same type  $t \in Tp$ . Thus  $G$  is reduced with respect to  $X$  if it simply contains no useless word.

---

<sup>5</sup>Equivalent grammars are grammars that are associated to the same language. A sufficient condition is the existence of a bijective relation between the primitive types of both grammars.

The algorithm  $\phi_1$  takes an input sequence  $\mathbf{s} = N_1, \dots, N_l$  of untyped dependency nets and returns a CDG that corresponds to the smallest

rigid untyped dependency net language compatible with  $\mathbf{s}$ .

It returns a failure if the sequence corresponds to no rigid untyped dependency net language<sup>6</sup>. Here is the algorithm  $\phi_1$  :

1. For each word  $w$ , collect in  $\mathbf{s}$  its occurrences together with its left and right arities. Fail if a word is used with different arities.
2. With each word  $w$  in  $\mathbf{s}$ , associate  $n+m+1$  variables corresponding to its  $n$  left argument categories, its  $m$  right argument categories and its head  
type:  
 $X_{-n}^w, \dots, X_{-1}^w, X_0^w, X_1^w, \dots, X_m^w$   
( $X_0^w$  corresponding to the head type).
3. Infer from  $\mathbf{s}$  the equality constraints for the variables corresponding to the beginnings and the ends of the same dependencies.  
Respect the orientation: for each word  $w$  and each of its argument category, the corresponding dependency must be oriented from  $w$  to a word with the corresponding head type. Return a failure if this condition is not fulfilled.
4. Resolve the resulting equality system and associate an elementary category with each variables' equivalence class  $\overline{X_i^w}$ .
5. Return the CDG  $G_{\mathbf{s}}$  such that for each word  $w$  in  $\mathbf{s}$  with associated variables  $X_{-n}^w, \dots, X_{-1}^w, X_0^w, X_1^w, \dots, X_m^w$ , the lexicon of  $G_{\mathbf{s}}$  assigns to  $w$  the category  $[\overline{Y_{-n}^w} \setminus_{-n} \dots \setminus_{-2} \overline{Y_{-1}^w}, \setminus_{-1} \overline{Y_0^w} /_1 \overline{Y_1^w} /_2 \dots /_m \overline{Y_m^w}]$ , in which:
  - $Y_i^w$  is  $X_i^w$  if the corresponding dependency is local and everywhere
 in  $\mathbf{s}$  there is exactly one incoming / outgoing dependency in this slot.

---

<sup>6</sup>I.e. when this sequence is not included in at least one of the rigid untyped dependency net languages.

Besides this,  $\setminus_i$  is  $\setminus$  if  $i \neq 0$  (respectively,  $/_i$  is  $/$  in the case of right argument).

- Otherwise,  $Y_i^w = (X_i^w)^?$  if there is at most one outgoing dependency in this slot and at least one net with no outgoing dependency in this slot in  $\mathbf{s}$ ,  
or  $Y_i^w = (X_i^w)^*$  if there is a net with several outgoing dependencies  
and at least one net with no outgoing dependency in this slot,  
or finally,  $Y_i^w = (X_i^w)^+$  if there is always more than one outgoing dependency and there is at least one net with several outgoing dependencies

in this slot. Besides this,  $\setminus_i$  is  $\setminus$  (respectively,  $/_i$  is  $/$ ).

- $Y_i^w$  is  $\#(\swarrow X_i^w)$  (resp.  $\#(\searrow X_i^w)$ ) if the left (respectively, right) slot is the end of an anchored dependency.
- $Y_i^w$  is  $\nwarrow X_i^w$  (respectively,  $\nearrow X_i^w$ ) if the left (respectively, right) slot is the beginning of a distant dependency.
- $Y_i^w$  is  $\swarrow X_i^w$  (respectively,  $\searrow X_i^w$ ) if the slot is the end of a left (respectively, right) loose distant dependency.

**Theorem 20**  $\phi_1$  learns rigid untyped dependency net grammars.

**Proof:**  $\phi_1$  is monotonic: if  $\mathbf{s}_1 \subseteq \mathbf{s}_2$  then

$\phi_1(\mathbf{s}_2)$  returns a failure or  $\phi_1(\mathbf{s}_1)$  and

$\phi_1(\mathbf{s}_2)$  succeeds and

$\text{UNL}(\phi_1(\mathbf{s}_1)) \subseteq \text{UNL}(\phi_1(\mathbf{s}_2))$ .

This is a consequence of the fact that the equality system corresponding

to  $\mathbf{s}_1$  is a subset

of the equality system corresponding to  $\mathbf{s}_2$ .

Let  $G$  be a rigid CDG and  $(N_i)_{i \in N}$  be an infinite sequence of untyped dependency nets that enumerates  $\text{UNL}(G)$ . For  $i \in N$ ,

$\phi_1(N_0, \dots, N_i)$  does not return a failure because  $G$  is

rigid so there exists a way to assign a unique type to each word in the

untyped dependency nets of  $\text{UNL}(G)$ . Because  $\phi_1$  is monotonic,

$(G_i = \phi_1(N_0, \dots, N_i))_{i \in N}$  defines an infinite sequence

of growing languages  $\text{UNL}(G_0) \subseteq \text{UNL}(G_1) \subseteq \dots$ .

The property of finite elasticity implies that this sequence must converge to a language  $L_\infty$  that must be (equal or) a superset of  $\text{UNL}(G)$  since the sequence enumerates

$\text{UNL}(G)$ . In fact, for  $i \in N$ ,  $G$

verifies the equality system used by  $\phi_1$  with

$N_0, \dots, N_i$  as input, so

$$\mathcal{UNL}(\phi_1(N_0, \dots, N_i)) \subseteq \mathcal{UNL}(G).$$

Thus, we also have  $L_\infty \subseteq \mathcal{UNL}(G)$  and the sequence of languages converges. Because if  $G_1$  and  $G_2$  are two grammars such that  $G_1 \subseteq G_2$  and  $\mathcal{UNL}(G_1) = \mathcal{UNL}(G_2)$  that are reduced with respect to  $\mathcal{UNL}(G_1) = \mathcal{UNL}(G_2)$  then  $G_1 = G_2$ : the sequence of grammars converges. ■

## 1.6 $k$ -valued CDGs without optional or iterative category are learnable from strings

We can define a finite-valued relation between the set of untyped dependency nets that are images of a  $k$ -valued CDG through  $\mathcal{L}$ . The class of rigid untyped dependency net languages having finite elasticity, we can apply Theorem 14 and see that  $k$ -valued CDGs without optional or iterative categories are learnable from strings. In fact, we define two relations and use Theorem 14 twice: first time from rigid untyped dependency net languages to rigid string languages, and the second time from rigid string languages to  $k$ -valued string languages.

**Lemma 21** *String languages of rigid CDG without optional or iterative categories have finite elasticity.*

**Proof:** Given any string  $x = w_1 \dots w_n$ , the maximum number of dependencies drawn over  $x$  is  $2n - 2$ , because there is at most one local or distant dependency and one anchored dependency coming in each word (except the main word). Given any untyped dependency net and any node in this dependency net, all slots are connected to at least one dependency. Therefore there exists a finite number of untyped link dependency nets corresponding to a string  $x$ . The class of rigid untyped link dependency nets has finite elasticity, so by theorem 14 the class of rigid CDG string languages without optional or iterative categories also has finite elasticity. ■

**Lemma 22**  *$k$ -valued CDGs without optional or iterative category have finite elasticity.*

**Proof:** This is very standard. The finite-valued relation associates  $k$ -valued CDG over  $W$  and rigid CDG over  $W \times \{1, \dots, k\}$ . A rigid CDG over  $W \times \{1, \dots, k\}$  corresponds to the  $k$ -valued CDG where the types associated to  $(a, 1), \dots, (a, k)$  are merged into the same entry for  $a$ . ■

## 1.7 Conclusion and perspectives

We have proved that a class of rigid and  $k$ -valued non-projective DGs has finite elasticity and so is learnable from strings and we have obtained some unlearnability results, as summarized below.

Class	Learnable from strings		Finite elasticity on strings		Finite elasticity on structures		Finite-valued relation
$A^*$	no	$\Rightarrow$	no		yes	$\Rightarrow$	no
$A^?$	no	$\Rightarrow$	no		yes	$\Rightarrow$	no
$A^+$	yes	$\Leftarrow$	yes	$\Leftarrow$	yes		yes

The positive results may be compared to other learnability results in the same domain in particular in the field of  $k$ -valued categorical grammars. For instance, Kanazawa’s positive result on classical categorical grammars corresponds to the learnability of the subclass of projective CDGs. On the other hand, some more complex but rather close systems like rigid Lambek calculus or pregroups have been proved to be not learnable from strings Foret and Le Nir (2002a,b). One of possible reasons of this effect might be that - in contrast with the CDGs - in these classes of grammars, reasoning from strings, one can not bound the number of “interactions” (axiom links in terms of proof nets) between two words. This remark may lead to other learnable classes of logical categorical grammars laying between classical categorical grammars and Lambek calculus.

## References

- Bechet, Denis. 2003.  $k$ -valued link grammars are learnable from strings. In *Proceedings of the 8th conference on Formal Grammar (FGVienna)*.
- Besombes, Jérôme and Jean-Yves Marion. 2001. Identification of reversible dependency tree languages. In L. Popelínský and M. Nepil, eds., *Proceedings of the 3d Workshop on Learning Language in Logic*, pages 11–22. Strasbourg, France.
- Buszkowski, Wojciech and Gerald Penn. 1989. Categorical grammars determined from linguistic data by unification. Tech. Rep. TR-89-05, Department of Computer Science, University of Chicago.
- Dikovsky, Alexander. 2004. Dependencies as categories. In G.-J. Kruiff and D. Duchier, eds., *Proc. of Workshop “Recent Advances in Dependency Grammars”*. In conjunction with COLING 2004. Geneva, Switzerland.
- Foret, Annie and Yannick Le Nir. 2002a. Lambek rigid grammars are not learnable from strings. In *COLING’2002, 19th International Conference on Computational Linguistics*. Taipei, Taiwan.
- Foret, Annie and Yannick Le Nir. 2002b. On limit points for some variants of rigid lambek grammars. In *ICGI’2002, the 6th International Colloquium on Grammatical Inference*, no. 2484 in Lecture Notes in Artificial Intelligence. Springer-Verlag.
- Gaifman, Haim. 1961. Dependency systems and phrase structure systems. Report p-2315, RAND Corp. Santa Monica (CA). Published in: *Information and Control*, 1965, v. 8, n 3, pp. 304-337.

- Kanazawa, Makoto. 1998. *Learnable classes of categorial grammars*. Studies in Logic, Language and Information. FoLLI & CSLI. distributed by Cambridge University Press.
- Lombardo, Vincenzo and Leonardo Lesmo. 1996. An earley-type recognizer for dependency grammar. In *Proc. 16th COLING*, pages 723–728.
- Mel'cuk, I. 1988. *Dependency Syntax: Theory and Practive*. State University of New York Press.
- Moreau, Erwan. 2001. *Apprentissage des grammaires catgorielles et de dpendances*. Master's thesis, Université de Nantes, France. (in French).
- Motoki, Tatsuya, Takeshi Shinohara, and Keith Wright. 1991. The correct definition of finite elasticity: Corrigendum to identification of unions. In *The fourth Annual Workshop on Computational Learning Theory*, page 375. San Mateo, Calif.: Morgan Kaufmann.
- Shinohara, T. 1990. Inductive inference from positive data is powerful. In *The 1990 Workshop on Computational Learning Theory*, pages 97–110. San Mateo, California: Morgan Kaufmann.
- Shinohara, T. 1991. Inductive inference of monotonic formal systems from positive data. *New Generation Computing* 8 (4):371–384. Special Issue on Algorithmic Learning Theory for ALT'90.
- Sleator, D. and D. Temperley. 1993. Parsing English with a link grammar. In *Third International Workshop on Parsing Technologies*.
- Wright, K. 1989. Identifications of unions of languages drawn from an identifiable class. In *The 1989 Workshop on Computational Learning Theory*, pages 328–333. San Mateo, Calif.: Morgan Kaufmann.

---

# Learning Dependency Languages from a Teacher

JÉRÔME BESOMBES AND JEAN-YVES MARION

We investigate learning dependency grammar from partial data and membership queries as a model of natural language acquisition. We define a learning paradigm based on a dialogue between the learner and a referent who knows the target language. This dialogue consists in a presentation of structured partial sentences and queries about the membership of original sentences constructed by the learner. We define an efficient algorithm corresponding to this paradigm and illustrate it on examples.

## 2.1 Introduction

For the definition of our model we consider several hypotheses which are largely inspired by the works of the linguist Chomsky Chomsky (1986) the psycholinguist Pinker Pinker (1994). First, the child learns the language of his parents or more specifically, the language he hears. Correct sentences are so presented to the learner, possibly partially understood and constitute the input data of the model. These data are not only linear sentences but pre-calculated structures. Indeed, semantic information or prosody are information included in the signal

The structures are commonly considered as tree in which nodes are labelled by the words of the sentence. Since the linear order of the words is conserved during the structuration process, we will consider dependency trees as a relevant model of the input data (Figure 1).

Pinker underlines that the learner and the referent take part in a

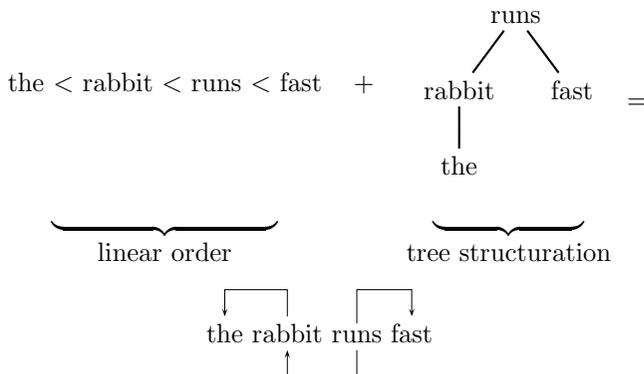


FIGURE 1 Dependency structure

Natural learning properties	Algorithmic model properties
Finite set of correct sentences are needed	Input data are a finite set of partial positive examples
Chomsky's universal grammar	Learning algorithm independent of a particular language
Structured data	Dependency tree languages
Communication with a referent	Membership queries

FIGURE 2 Correspondance between properties of the natural language aquisition and properties of the algorithmic model

communication process; this communication is a key point of the learning process since it turns out that the language acquisition is not possible with no physical presence of the referent. If we suppose that a new sentence produced by a child and not understood by the referent can provide the conclusion that this sentence is not correct (doesn't belong to the language he learns), we will take into account membership queries: the algorithm submits sentences to an Oracle who replies yes or no whether they belongs to te target language or not.

The algorithm  $A$  learns a class of language if and only if, for any language  $L$  in the class, there is a finite set of partial data  $RS$  (representative sample) such that  $A$  determines  $L$  from  $RS$  with help of membership queries. Properties of  $A$  are summerized in Figure 2

## Related works

Angluin first introduced the paradigm of learning with queries in Angluin (1987) for the case of regular languages and in Angluin (1988) is studied a paradigm of learning from positive examples, membership queries and equivalence queries (the possibility to ask an Oracle whether a guess language corresponds to the target language or not). Obviously, for our motivation of modelling, this kind of queries are not relevant. Angluin's works have been extended in particular by Sakakibara (1987b,a, 1990) for the inference of context-free grammars from structured data. The learnability of dependency languages has been studied in Besombes and Marion (2002); in this work, an algorithm for a sub-class of lexical dependency languages has been defined. As far as we know, the idea of learning from partial data and membership queries is original.

## 2.2 Lexical dependency grammar

Following Dikovsky and Modina (2000), we present a class of projective dependency grammars which was introduced by Hays (1961) and Gaifman (1965).

A *lexical dependency grammar* (LDG)  $\Gamma$  is a quadruplet  $\langle \Sigma, N, P, S \rangle$ , where:

- $\Sigma$  is the set of terminal symbols,
- $N$  is the set of non-terminal symbols,
- $S \in N$  is the start symbol,
- $P$  is the set of productions.

Each production is of the form

$$X \rightarrow X_1 \dots X_p a X_{p+1} \dots X_q$$

or of the form

$$X \rightarrow a^1$$

where  $X$  and each  $X_i$  are in  $N$  and  $a$  in  $\Sigma$ . The terminal symbol  $a$  is called the *head* of the production. In other words, the head is the root of the flat tree formed by the production right handside. Actually, if we forget dependencies, we just deal with context free grammars.

Given a grammar  $G$ , *partial dependency trees*  $t$  generated by a non-terminal  $X$  of  $G$  are recursively defined as follows.

- $X$  is a partial dependency tree.

---

<sup>1</sup>This form is corresponding to the previous with  $p = q = 0$ .

- If  $\dots X' \dots b \dots$  is a partial dependency tree generated by  $X$ ,

and if  $X' \rightarrow X_1 \dots X_p a X_{p+1} \dots X_q$  is a production of  $G$ , then

$\dots X_1 \dots X_p a X_{p+1} \dots X_p \dots b \dots$ <sup>2</sup> is a partial dependency tree generated by  $X$ .

We note  $X \overset{*}{\rightarrow} t$  to express that  $t$  is generated by  $X$ .

A *dependency tree* generated by a non-terminal  $X$  is a partial dependency tree generated by  $X$  in which all nodes are terminal symbols. A *dependency tree* is a dependency sub-tree generated by  $S$ . The language  $\mathcal{DL}(G)$  is the set of all dependency trees ( $\mathcal{DL}(G) = \{d : \text{dependency tree and } S \overset{*}{\rightarrow} d\}$ ).

- (1) Example. Consider the grammar  $G$  defined by:

$$G = \langle \Sigma, N, P, S \rangle$$

where

- $\Sigma = \{a, b, c\}$ ,
- $N = \{S, X_1, X_2, X_3, X_4\}$ ,
- $P$  is the following set of productions.

$$\begin{array}{l}
 S \rightarrow X_2 a X_3 \\
 X_2 \rightarrow X_1 b \qquad X_3 \rightarrow c X_4 \\
 X_1 \rightarrow X_2 b \qquad X_3 \rightarrow c \\
 X_1 \rightarrow b \qquad X_4 \rightarrow c X_3
 \end{array}$$

The language  $\mathcal{DL}(G)$  is the set of dependency trees

$$\underbrace{\{b \dots b a c \dots c\}}_{n \text{ even} \quad m \text{ odd}}$$

---

<sup>2</sup>Dependencies can be drawn either over or under the word line for a reason of clarity.

A *subtree* of a dependency tree is inductively defined as follows:

- of  $d = a$  for a terminal symbol  $a$ , then  $d$  is the only subtree of  $d$ ,



- if  $d = d_1 \dots d_p a d_{p+1} \dots d_q$  is a dependency tree then:
  - $d$  is a *subtree* of  $d$ ,
  - any subtree of  $d_i$  is a subtree of  $d$ .

If  $d$  is a dependency tree,  $S(d)$  is the set of subtrees of  $d$  and if  $D$  is a set of dependency trees,  $S(D)$  is the set of all subtrees of the elements of  $D$ . A *context*  $c[\#]$  of a dependency tree  $d$  is obtained by replacing exactly one occurrence of a subtree of  $d$  by a special symbol  $\#$ . In particular  $\#$  is a context of all dependency trees. If  $d$  is a dependency tree,  $C(d)$  is the set of contexts of  $d$  and if  $D$  is a set of dependency trees,  $C(D)$  is the set of all contexts of the elements of  $D$ .

We will also use the notation  $d = c[d']$  to express that  $d'$  is a subtree of  $d$ .

A grammar homomorphism  $\phi$  between two grammars  $G = \langle \Sigma, N, P, S \rangle$  and  $G' = \langle \Sigma, N', P', S' \rangle$  is defined from a surjective mapping from  $N$  to  $N'$  which satisfies the following properties:

- $\phi(S) = S'$

- $P'$  is the set of productions  $\phi(X) \rightarrow \phi(X_1) \dots \phi(X_p) a \phi(X_{p+1}) \dots \phi(X_q)$

for every production  $X \rightarrow X_1 \dots X_p a X_{p+1} \dots X_q$  of  $P$ .

We note  $G' = \phi(G)$  and in this case we have  $\mathcal{DL}(G) \subseteq \mathcal{DL}(G')$ .

### 2.3 Observation table

Following Angluin Angluin (1988), information obtained from the membership queries is stored in a table. Let  $\mathcal{DL}$  be a dependency language,  $D$  a finite set of subtrees and  $C$  a finite set of contexts. The *observation table*  $T = T_{\mathcal{DL}}(S(D), C)$  is the table defined by:

- rows are labelled by the subtrees of  $D$ ,
- columns are labelled by elements of  $C$ ,
- cells  $T_{\mathcal{DL}}(d, c[\#])$ , where  $d \in S(D)$  and  $c[\#] \in C$ , are labelled with 1 and 0 in such a way that:

$$T_{\mathcal{DL}}(d, c[\#]) = \begin{cases} 1 & \text{if } c[d] \in \mathcal{DL} \\ 0 & \text{otherwise} \end{cases}$$

For any  $d \in S(D)$ , we denote by  $row_T(d)$  the binary word corresponding to the reading from left to right of the row labelled by  $d$  in  $T$ .

(2) Example. Let be  $\mathcal{DL} = \mathcal{DL}(G)$  the dependency language defined

in Example 1,  $D$  the singleton  $\{b \overbrace{b} \overbrace{a} \overbrace{c} \overbrace{c} \overbrace{c}\}$  and  $C$  the set of

contexts  $\{\# \overbrace{b} \overbrace{a} \overbrace{c} \overbrace{c} \overbrace{c}, \# \overbrace{a} \overbrace{c} \overbrace{c} \overbrace{c}, b \overbrace{b} \overbrace{a} \overbrace{c} \overbrace{c} \#, b \overbrace{b} \overbrace{a} \overbrace{c} \#, b \overbrace{b} \overbrace{a} \#\}$ .

The corresponding observation table  $T = T_{\mathcal{DL}}(S(D), C)$  is the table of figure 2.

An observation table  $T = T_{DL}(S(D), C)$  is *coherent* if and only if for any pair  $(d, d')$  of trees in  $D \times D$ ,  $row_T(d) = row_T(d')$ . A coherent observation table  $T = T_{DL}(S(D), C)$  defines a grammar  $G_T$ :

$$G_T = \langle \Sigma, N, P, S \rangle$$

where:

- $\Sigma$  is set of symbols occuring in  $D$ ,
- $N = \{row_T(d) : d \in S(D)\}$
- $S = row_T(d)$  for any dependency tree  $d \in D$

- $P$  is the set of productions of the form  $row_T(d_1 \dots d_p a d_{p+1} \dots d_q)$

$$\rightarrow row_T(d_1) \dots row_T(d_p) a row_T(d_{p+1}) \dots row_T(d_q)$$

for all  $d_1 \dots d_p a d_{p+1} \dots d_q$  in  $S(D)$ .

(3) Example. The table of Example 2 is coherent and the corresponding grammar is  $\phi(G)$ , where  $G$  is the grammar given in Example 1 and  $\phi$  the homomorphism defined by  $\phi(S) = 100000$ ,  $\phi(X_1) = 010000$ ,  $\phi(X_2) = 001000$ ,  $\phi(X_3) = 000101$ ,  $\phi(X_4) = 000010$ .

A coherent table  $T = T_{DL}(S(D), C)$  is *consistent* if and only if for every

dependency trees  $d = d_1 \dots d_p a d_{p+1} \dots d_q$  and  $d' = d'_1 \dots d'_p a d'_{p+1} \dots d'_q$

	#	# $\begin{array}{c} \downarrow \quad \downarrow \quad \downarrow \\ b \ a \ c \ c \ c \\ \uparrow \quad \uparrow \quad \uparrow \end{array}$	# $\begin{array}{c} \downarrow \quad \downarrow \quad \downarrow \\ a \ c \ c \ c \\ \uparrow \quad \uparrow \quad \uparrow \end{array}$	# $\begin{array}{c} \downarrow \quad \downarrow \quad \downarrow \\ b \ b \ a \ c \ c \ # \\ \uparrow \quad \uparrow \quad \uparrow \end{array}$	# $\begin{array}{c} \downarrow \quad \downarrow \quad \downarrow \\ b \ b \ a \ c \ # \\ \uparrow \quad \uparrow \quad \uparrow \end{array}$	# $\begin{array}{c} \downarrow \quad \downarrow \\ b \ b \ a \ # \\ \uparrow \quad \uparrow \end{array}$
$\begin{array}{c} \downarrow \quad \downarrow \quad \downarrow \\ b \ b \ a \ c \ c \ c \\ \uparrow \quad \uparrow \quad \uparrow \end{array}$	1	0	0	0	0	0
$\begin{array}{c} \downarrow \\ b \ b \end{array}$	0	0	1	0	0	0
$b$	0	1	0	0	0	0
$\begin{array}{c} \downarrow \\ c \ c \ c \\ \uparrow \end{array}$	0	0	0	1	0	1
$\begin{array}{c} c \ c \\ \uparrow \end{array}$	0	0	0	0	1	0
$c$	0	0	0	1	0	1

FIGURE 3 An observation table

in  $S(D)$ , for all  $i$ ,  $row_T(d_i) = row_T(d'_i)$  implies that  $row_T(d) = row_T(d')$ .

### 2.4 Representative sample

We now define the property, for a finite set of subtrees of a language, to contain the minimum information necessary to explicitly identify this language. This constitutes a minimal hypothesis to conclude in the learnability of the language. Let  $\mathcal{DL}$  be a dependency language generated by a grammar  $G$ . Any finite subset  $RS$  of  $S(\mathcal{DL})$  is said to be *representative* for  $\mathcal{DL}$  if and only if for any transi-

tion  $X \rightarrow X_1 \dots X_p a X_{p+1} \dots X_q$  of  $G$ , there is an element  $d =$   
 $d_1 \dots d_p a d_{p+1} \dots d_q$  in  $S(RS)$  such that for all  $i$ ,  $X_i \xrightarrow{*} d_i$ . Informally, a finite set  $RS$  is a representative sample for  $G$  if and only if each production of  $G$  has been used at least once to produce the elements of  $RS$ .

**Lemma 1** *Let  $G$  be a dependency grammar,  $RS$  a representative sample for  $\mathcal{DL}(G)$  and  $C$  a finite set of contexts containing  $C(RS)$ , if  $T_{\mathcal{DL}(G)}(S(RS), C)$  is consistent then  $\mathcal{DL}(G_T) = \mathcal{DL}(G)$ .*

**Theorem 2** *The algorithm defined in Figure 4 learns the class of dependency languages from representative samples and membership queries.*

The algorithm works as follows: it take a finite set of dependency trees as input and this set is decomposed in a finite set of subtrees and a finite set of contexts. With help of membership queries, a first observation table is constructed and the consistence is checked. If the table is not consistent, new contexts are calculated and added in the table which is then completed. The process stops as the table is consistent and a grammar is then output.

### 2.5 Examples

(4) Example. The singleton  $\{b b a c c c\}$  is a representative sample

for the dependency tree language defined in Example 1. The observation table of Figure 3 is constructed from this input with help of membership queries; this table is consistent that implies

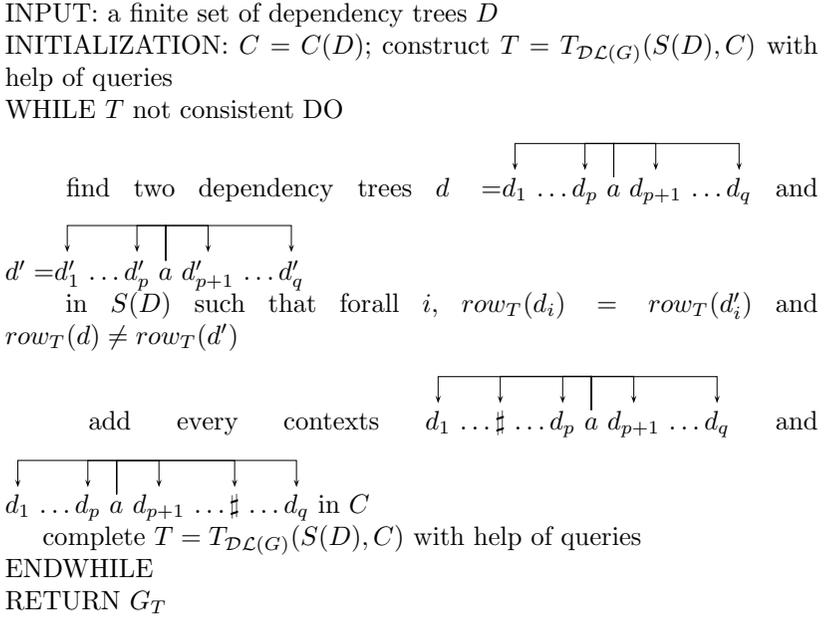


FIGURE 4 The learning algorithm

that the corresponding dependency grammar given in Example 3 is computed by the algorithm and the language is learnt immediately (the loop is not processed).

The following example illustrates the iterative behavior of the algorithm.

(5) Example. Let  $G$  be the following grammar:

$$\begin{array}{l}
 S \rightarrow aX_1, aX_2, bX_2 \\
 X_1 \rightarrow dX_3, c \qquad X_3 \rightarrow e \\
 X_2 \rightarrow dX_4 \qquad X_4 \rightarrow f
 \end{array}$$

We have:  $\mathcal{DL} = \{b \ c, a \ c, a \ d \ e, b \ d \ e, a \ d \ f\}$ .

Let now consider the following representative sample:

$$RS = \{b \quad c, a \quad d \quad e, d \quad f\}$$

From it, the algorithm constructs a first table that is not consistent (Figure 5). Indeed we have:

$$row_T(e) = row_T(f)$$

but

$$row_T(d \quad e) \neq row_T(d \quad f)$$



The new context  $b \quad d \quad \#$  is computed and added to the table that is completed with queries. The table obtained is then consistent and the process stops with the construction of the grammar  $\phi(G)$ , where  $\phi$  is defined by  $\phi(S) = 10000$ ,  $\phi(X_1) = 01010$ ,  $\phi(X_2) = 00010$ ,  $\phi(X_3) = 00101$ ,  $\phi(X_4) = 00100$

## References

- Angluin, D. 1987. Learning regular sets from queries and counter examples. *Information and Control* 75:87–106.
- Angluin, D. 1988. Queries and concept learning. *Machine learning* 2:319–342.
- Besombes, J. and J.Y. Marion. 2002. Apprentissage des langages réguliers d’arbres et applications. *Conférence d’Apprentissage, Orléans 17, 18 et 19 juin 2002* pages 55–70.
- Chomsky, N. 1986. *Knowledge of Language*. Praeger, New York.
- Dikovskiy, A. and L. Modina. 2000. Dependencies on the other side of the curtain. *Traitement automatique des langues* 41(1):67–96.
- Gaifman, H. 1965. Dependency systems and phrase structure systems. *Information and Control* 8(3):304–337.
- Hays, D.G. 1961. Grouping and dependency theories. In *National symp. on machine translation*.
- Pinker, S. 1994. *The language instinct*. Harper.
- Sakakibara, Y. 1987a. Inductive inference of logic programs based on algebraic semantics. Tech. Rep. ICOT, 79.
- Sakakibara, Y. 1987b. Inferring parsers of context-free languages from structural examples. Tech. Rep. ICOT, 81.
- Sakakibara, Y. 1990. Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science* 76:223–242.

	#	$\overbrace{b}^{\curvearrowright}$ #	$\overbrace{a d}^{\curvearrowright}$ #	$\overbrace{a}^{\curvearrowright}$ #	$\overbrace{b d}^{\curvearrowright}$ #
$\overbrace{b c}^{\curvearrowright}$	1	0	0	0	0
$c$	0	1	0	1	0
$\overbrace{a d e}^{\curvearrowright}$	1	0	0	0	0
$\overbrace{d e}^{\curvearrowright}$	0	1	0	1	0
$\overbrace{d f}^{\curvearrowright}$	0	0	0	1	0
$e$	0	0	1	0	1
$f$	0	0	1	0	0

↓

	#	$\overbrace{b}^{\curvearrowright}$ #	$\overbrace{a d}^{\curvearrowright}$ #	$\overbrace{a}^{\curvearrowright}$ #	$\overbrace{b d}^{\curvearrowright}$ #
$\overbrace{b c}^{\curvearrowright}$	1	0	0	0	0
$c$	0	1	0	1	0
$\overbrace{a d e}^{\curvearrowright}$	1	0	0	0	0
$\overbrace{d e}^{\curvearrowright}$	0	1	0	1	0
$\overbrace{d f}^{\curvearrowright}$	0	0	0	1	0
$e$	0	0	1	0	1
$f$	0	0	1	0	0

FIGURE 5 The learning algorithm processing



---

# An integrated approach to French liaison

OLIVIER BONAMI, GILLES BOYÉ, JESSE TSENG

Consonant liaison at word boundaries in French is the result of a complex interplay of grammatical and extragrammatical factors. In this paper we offer a descriptive overview of syntactic factors influencing liaison. We provide a detailed analysis in the framework of HPSG, that integrates the morphophonological and syntactic conditions governing this feature of French grammar. Although we do not directly model other factors influencing liaison (such as frequency effects, prosodic considerations, or sociolinguistic variables) our analysis is modular enough to accommodate additional conditions resulting from more complete empirical studies.<sup>1</sup>

## 3.1 A descriptive overview of liaison

Many French words come in two shapes, which we will refer to as the “short form” and the “long form”.<sup>2</sup> The short form is always used before consonants and in utterance-final position; the long form is used, in some cases, when the following word is vowel initial. The term *liaison* refers to the realization of the long form in appropriate contexts.<sup>3</sup>

We separate two issues in the grammar of liaison: the identification of

---

<sup>1</sup>We thank Anne Abeillé, Elisabeth Delais-Roussarie, Danièle Godard and three anonymous *Formal Grammar* referees for their comments and suggestions.

<sup>2</sup>By “words” we mean fully inflected lexical items, as opposed to lexemes or paradigm headwords (which can correspond to many distinct surface forms).

<sup>3</sup>A few isolated words, such as *six* ‘six’, have two long forms, one for liaison contexts ([siz]) and one for utterance-final position and certain other contexts ([sis]).

the contexts where liaison is possible and the relationship between short and long forms. In the simplest cases of short/long form alternation, the long form is just the short form with an additional, so-called “latent” final consonant. For instance, the adverb *très* ‘very’ has a short form [tʁɛ], found e.g. in *très doué* ‘very gifted’ [tʁɛdwe], and a long form [tʁɛz] found e.g. in *très intelligent* ‘very intelligent’ [tʁɛzɛ̃tɛlizɑ̃]. In section 3.1.1 we restrict our attention to this basic type of short/long form alternation, to examine the syntactic contexts where liaison is possible and/or mandatory. We defer the examination of more unusual short/long form pairs until section 3.1.2.

### 3.1.1 Syntactic contexts for liaison

Given a word  $w_1$  with distinct short and long forms and a following vowel-initial<sup>4</sup> word  $w_2$ , liaison is observed to be obligatory (indicated by  $w_1 = w_2$ ), optional (indicated by  $w_1 \circ w_2$ ), or impossible (indicated by  $w_1 \neq w_2$ ). This is reflected in prescriptive manuals of French, which include long, but rather arbitrary, lists of “correct” and “incorrect” liaisons (Delattre, 1966, Fouché, 1959). Early formal approaches to liaison concentrated on finding appropriate generalizations about the domain of liaison. Selkirk (1972, 1974) suggests that the domains of obligatory and optional (or “stylistically elevated”) liaison can be determined by means of simple, cross-categorial rules referring to explicitly defined notions like “phonological word” and “inflected lexical head”. Morin and Kaye (1982), however, present many counterexamples to Selkirk’s generalizations, and conclude that the status of liaison (obligatory, optional, impossible) must more or less be stipulated on a construction-by-construction basis. In later work, liaison has been used as a prime argument for distinguishing syntactic constituency from prosodic constituency: according to Selkirk (1986), obligatory and optional liaison occur within two different kinds of prosodic constituents, determined by applying an algorithm to syntactic structures that results in new constituent boundaries. Later work has shown that obligatory liaison is much more limited than the prosodic approach suggests (de Jong, 1994), and that liaison domains do not coincide with independently identifiable prosodic domains (Post, 2000).

At the same time, many studies have emphasized the importance of sociolinguistic factors in the realization of liaison (see Encrevé 1988 for a detailed discussion), the lexical conditioning of many liaison contexts (Tranel, 1981, de Jong, 1994, Morin, 1998) and the influence of frequency (Bybee, 2001) or prosodic factors (Fougeron et al., 2001) on

---

<sup>4</sup>For the time being we ignore the problem of *h aspiré*; see section 3.2.1.

the actual realization of liaison. Thus the current consensus is that there is a wide variety of factors involved in liaison, and few (if any) studies attempt to provide actual grammar fragments predicting the contexts where liaison is realized.

Our goal in this paper is to present a grammar fragment that incorporates genuinely syntactic constraints on liaison realization. Using observations taken from the previous literature and new evidence, we outline an updated list of the contexts where syntax forces liaison or makes it impossible. We assume that optional liaison is the default situation, and that specific syntactic environments make liaison impossible or mandatory in particular cases.<sup>5</sup>

What follows is a survey of the main phrasal structures of French; in anticipation of our formal analysis, we are guided by the description of French phrase structure in HPSG of Abeillé and Godard (2000, 2002).

In subject-head combinations, liaison is impossible between the daughters, irrespective of the head's category (compare (1a) and (1b)). Note that by contrast, liaison is obligatory between a weak form subject pronoun and the verb (2). This is part of the motivation for analyzing these pronouns as affixes on the finite verb rather than daughters in a syntactic combination (Miller and Sag, 1997, Miller, 1992).

- (1) a. [Les enfants≠ont mangé].  
       'The children have eaten.'  
       b. [Les enfants≠au lit], on sert le dessert.  
       (With) the children in bed, dessert was served.'
- (2) [Ils=étaient contents].  
       'They were happy.'

Liaison is obligatory between a specifier and the following head, as shown in (3).<sup>6</sup>

- (3) a. mon=ami  
       'my friend'

---

<sup>5</sup>It is important to interpret "optionality" in this paper as the absence of any *syntactic* condition on the realization of liaison in a particular phrasal configuration. Lexical and other factors typically intervene to make liaison more or less likely in specific instances of this configuration, even to the point of making it obligatory or impossible. As a simple example, liaison is possible in *enfants [z] intelligents* 'intelligent children' but not in \**enfant [t] intelligent* 'intelligent child'; it is nearly always realized in *très [z] intelligent* 'very intelligent', while other adverbs in the exact same structure give rise to liaison less systematically.

<sup>6</sup>Note that in our analysis of French, [Det N'] combinations are the only instances of specifier-head combinations.

- b. [mon=[ancien collègue]]  
 ‘my former colleague’

Head-complement structures in French are uniformly head-initial, and liaison is possible between the head and the first complement (4). There may be several complement daughters; liaison is only possible between a “lite” complement and a following complement (5a,b). Lite elements include the pronouns *tout* and *rien* and past participles in compound tenses (Abeillé and Godard, 2000, 2002). Other complements cannot give rise to liaison (6).<sup>7</sup>

- (4) a. Paul [pensait<sup>o</sup>à Marie].  
 ‘Paul was thinking about Marie.’  
 b. [dans<sup>o</sup>une semaine]  
 ‘in one week’s time’
- (5) a. Paul [donnera tout<sup>o</sup>à Marie].  
 ‘Paul will give everything to Marie.’  
 b. Paul [a été mis<sup>o</sup>à pied].  
 ‘Paul has been put on suspension.’
- (6) Jean [présentera ses enfants<sup>≠</sup>à Marie].  
 ‘Jean will introduce his children to Marie.’

In head-adjunct combinations, liaison is generally possible between the head daughter and the adjunct daughter, which exhibit both possible word orders (7).

- (7) a. [amis<sup>o</sup>intimes]  
 ‘close friends’  
 b. [très<sup>o</sup>intéressant]  
 ‘very interesting’  
 c. [bien<sup>o</sup>équipé]  
 ‘well equipped’

Prenominal attributive adjectives deserve special attention. It is traditionally assumed that liaison is obligatory between a prenominal adjective and the noun (e.g., *petit enfant* ‘small child’), while it is optional between the noun and a postnominal adjective (7a). However Post (2000) provides decisive evidence that while liaison is more frequent prenominally, it is in fact optional in both cases (see Morin and

---

<sup>7</sup>Modifying adverbs realized among complements in the VP may give rise to liaison (even when they are not lite), as noted by Morin and Kaye (1982). We leave these aside since their status (adjunct vs. complement) is controversial. Remember that we assume that pronominal clitics are affixes, not words; thus clitic liaison is a lexical phenomenon which falls outside of the scope of this paper.

Kaye (1982) and de Jong (1994) for earlier hints to this effect). In a reading task, Post observed that subjects realized liaison only 88% of the time with plural pronominal adjectives. This is especially significant since (i) liaison is more often realized in reading than in conversation (Fougeron et al., 2001), and (ii) the liaison rate is highly dependent on the choice of the noun and the adjective (falling to 61% for certain pairs), a situation typical of optional liaison contexts. We thus conclude that attributive adjectives conform to the general case of head-adjunct combinations, where liaison is not obligatory but only quite frequent.<sup>8</sup>

Liaison in filler-head combinations seems to be impossible (8a,b). Apparent counterexamples are predicative *quel(le)s*, which exhibits obligatory liaison (8c), and *dont*, with which liaison is optional (8d). But independent evidence shows that these items are not fillers: Comorovski (to appear) argues that predicative *quel* is a clitic combining directly with the verb; and obligatory liaison is just what we expect if *quel* combines with the verb in the lexicon. *Dont* is arguably a complementizer rather than a *wh*-word, since it does not give rise to pied-piping (9), and (ii) it cannot be followed by the complementizer *que* in varieties that allow this with *wh*-items (10); thus *dont* relatives are head-complement structures, and liaison is expected to be optional.

- (8) a. Quelles tartes≠ont-ils mangées ?  
 ‘Which pies did they eat?’  
 b. les enfants [auxquels≠elle a parlé]  
 ‘the children to whom she spoke’  
 c. Quels [z] étaient les enjeux ?  
 ‘What were the issues at stake?’  
 d. l’homme dont il a parlé  
 ‘the man he spoke about?’
- (9) a. Voilà l’homme au frère de qui j’ai parlé.  
 ‘Here is the man to whose brother I spoke.’  
 b. \*Voilà l’homme au frère dont j’ai parlé.
- (10) a. % Voilà l’homme à qui que j’ai parlé.  
 ‘Here is the man I spoke to.’  
 b. \*Voilà l’homme dont que j’ai parlé.  
 ‘Here is the man I spoke about.’

---

<sup>8</sup>Note that Post provides data only for *plural* pronominal adjectives; in the absence of relevant evidence we suppose that the same situation holds in the masculine singular.

In coordinations, liaison is generally possible between the penultimate conjunct and the conjunction (11a) and between the conjunction and the final conjunct (11b). However, when there are more than two conjuncts, liaison is impossible between adjacent conjuncts (11c).

- (11) a. [petits $\circ$ et grands]  
           ‘small and large’  
       b. [gentil mais $\circ$ idiot]  
           ‘nice but dumb’  
       c. livres [petits $\neq$ abîmés $\circ$ et chers]  
           ‘small, damaged, and expensive books’

Coordination reveals some previously overlooked liaison data. As we have just seen, liaison is in general optional before a conjunction, and so we can use coordination to identify further constraints associated with the right edge of particular phrasal combinations. For example, liaison remains possible when the conjunct preceding the conjunction is a specifier-head (12) or adjunct-head (13) combination (the last example also illustrates optional liaison after a coordinated structure).

- (12) [les amis] $\circ$ et les collègues de Marie  
           ‘Marie’s friends and colleagues’  
       (13) [[très bien] $\circ$ et très chaleureusement] $\circ$ accueilli  
           ‘very well and very warmly received’

What is more surprising is that liaison is blocked when the conjunct is a subject-head (14), filler-head (15), or head-complement (16) combination (note that liaison is blocked even when the final complement is *lite*). This suggests that phrase types constrain the possibility of liaison not only between their daughters, but also between the phrase as a whole and following material.<sup>9</sup>

- (14) [Paul dort] $\neq$ et Marie travaillait.  
           ‘Paul was sleeping and Marie was working.’  
       (15) [Qui dort] $\neq$ et qui ne dort pas ?  
           ‘Who was sleeping and who wasn’t?’  
       (16) a. Paul doit [acheter ces livres] $\neq$ ou les emprunter.  
           ‘Paul must buy those books or borrow them.’  
           b. impressionné [par les arguments] $\neq$ et par les exemples  
           ‘impressed by the arguments and by the examples’

---

<sup>9</sup>The constraints on filler-head and subject-head combinations probably follow from a more general constraint against liaison between a clause and the following material. Such a constraint is necessary to block liaison after a single-word clause: *Sortez $\neq$ et restez dehors !* ‘Get out and stay out!’

- c. les [femmes de marins]≠et leurs amants  
'sailors' wives and their lovers'
- d. Paul était [fier de tout]≠et enthousiasmé par n'importe quoi.  
'Paul was proud of everything and enthusiastic about just anything.'

### 3.1.2 The shape of the long form

As we stated above, for most words exhibiting a liaison alternation, the long form is identical to the short form except that it contains an extra final consonant. In many cases, the same consonant is also relevant for morphological processes. For instance, the masculine singular adjective *petit* 'small' has a long form [pətit] that ends in [t] just like the feminine singular form of the same adjective *petite* [pətit]. The same consonant shows up in derived words such as *petitesse* 'smallness' [pətitɛs], where the derivational suffix is [ɛs]. This type of data motivates the traditional idea that French phonological representations may contain a final "latent" consonant which shows up only when followed by material in the same word or at word boundaries where liaison is realized. Determining the shape alternation in these cases is a question of realization vs. non-realization of the latent consonant.

Although it is clear that the notion of latent consonant has some role to play in the grammar of French, it is important to make a clear distinction between the presence (or absence) of a latent consonant and the possibility (or impossibility) of liaison. First, singular nouns never give rise to liaison, despite the fact that some of them do have a latent consonant that is relevant morphologically (e.g., *dent* 'tooth' is realized as [dɑ̃] in all contexts, but is the base for the derived words *dentaire* [dɑ̃tɛʁ] 'dental' and *dentiste* [dɑ̃tist] 'dentist'). Thus the lexical representation of a word may include a latent consonant that is not involved in liaison.

Second, and more importantly, prenominal masculine singular adjectives may have a long form that is not related in this simple way to the corresponding short form. Three adjectives (*vieux* 'old', *beau* 'beautiful', and *nouveau* 'new') have a masculine singular long form phonologically identical to the feminine form of the adjective (resp. [vjɛj], [bɛl], and [nuvɛl]) but quite distinct from the masculine singular short form (resp. [vjø], [bo], and [nuvo]).<sup>10</sup> A dozen adjectives have a masculine singular long form whose final consonant is distinct from the one found in the feminine or in derived words (e.g., *gros* 'big', masculine singular

<sup>10</sup>The adjectives *mou* 'soft' and *fou* 'crazy' are usually also cited in this context, but their long forms have fallen out of use in contemporary French, except in a few fixed expressions.

short form [gʁo], masculine singular long form [gʁoz], feminine singular [gʁos], typical derived noun: *grosseur* ‘bigness’ [gʁosœʁ]).<sup>11</sup> Finally, many adjectives are simply not possible in the masculine singular before a noun triggering liaison (Miller, 1992, Morin, 1998). A typical example is *chaud*:

- (17) a. une ambiance chaude / une chaude ambiance  
           ‘a lively atmosphere’  
       b. un débat chaud / un chaud débat  
           ‘a lively debate’  
       c. un entretien chaud / \*un chaud entretien  
           ‘a lively discussion’

Prenominal position introduces a stilted stylistic effect for many adjectives, making liaison judgments difficult to evaluate. It is clear, however, that *chaud* is not an isolated case. Dozens of other adjectives have the same curious property of simply not having an acceptable masculine singular prenominal liaison form.

The data just discussed show that for the case of prenominal adjectives in the masculine singular, the relationship between short and long forms can involve more than just the realization or non-realization of a latent consonant. To account for this data, we assume that the paradigm of French adjectives contains an extra slot for the masculine singular prenominal liaison form (Bonami and Boyé, 2003). This form is identical to the feminine singular form by default; thus when the feminine singular is suppletive (as in the cases of *vieux/vieille*, *beau/belle*, *nouveau/nouvelle* discussed above), the masculine singular long form is identical to the feminine form, not the masculine singular short form. The default identity with the feminine singular is overridden in the case of *gros* and similar adjectives. Finally adjectives like *chaud* are simply defective—this lexeme is missing one of its inflectional forms.

To sum up, the relation between the short and long forms of a word can be determined by one of two factors: either the word has a latent consonant and the long form is the short form with the latent consonant realized at the end, or the short and long forms occupy distinct slots in

<sup>11</sup>The masculine singular long form of these adjectives is sometimes taken to be derived from the feminine singular by a phonological process turning [s] into [z] and [d] into [t] (Steriade, 1999). However this process would affect a non-natural class of segments, perform totally opposite operations on them (voicing vs. devoicing), and affect just a few lexical items sharing the same category and morphosyntactic features in a syntactically defined environment—and even then, not fully productively (for example, *chaud* ‘hot’ and *froid* ‘cold’ are never realized as \*[ʃot] and \*[fʁwat]). In light of these properties of the phenomenon, a lexical treatment is clearly preferable.

the inflectional paradigm of the lexeme, and are related by inflectional morphology.<sup>12</sup>

### 3.2 An HPSG analysis for optional liaison

In this section we outline the general analysis and show how it applies to optional liaison contexts; we defer discussion of obligatory and impossible liaison to section 3.3.

#### 3.2.1 Feature inventory

We introduce a number of new features to lexical and phrasal representations in order to encode the morphophonological conditions and effects of liaison.

$$(18) \quad \textit{sign} \rightarrow \left[ \begin{array}{l} \text{LEFT} \quad \left[ \begin{array}{l} \text{LTRIG} \quad \textit{boolean} \end{array} \right] \\ \text{RIGHT} \quad \left[ \begin{array}{l} \text{LFORM} \quad \textit{boolean} \\ \text{APP} \quad \textit{list(segment)} \end{array} \right] \end{array} \right]$$

First, a boolean-valued attribute LIAISON-TRIGGER indicates whether a word (potentially) licenses liaison to its left. Consonant-initial words (like *doué* in *très doué*) carry the feature [LTRIG −], except when the consonant is a glide ([j], [w], [ɥ]). In that case both possibilities exist: some words are [LTRIG +] (*mes* [z] *yeux* ‘my eyes’, *des* [z] *oiseaux* ‘birds’, *belles* [z] *huîtres* ‘beautiful oysters’), others are [LTRIG −] (*\*les* [z] *hiéroglyphes* ‘hieroglyphics’, *\*bon* [n] *week-end* ‘good weekend’, *\*des* [z] *huées* ‘jeers’). Vowel-initial words (e.g., *intelligent*, *ami*, *à*, *et*) are typically [LTRIG +], but there are exceptions too. So-called “*h aspiré*” words are (phonetically) vowel-initial, but they must be lexically specified as [LTRIG −] because they fail to trigger liaison: *\*curieux* [z] *hasard* ‘funny coincidence’, *\*tes* [z] *onze enfants* ‘your eleven children’.

Next, two features are needed for representing liaison target status. The feature APPENDIX encodes the latent consonant (for both the liaison alternation and morphological derivation); thus a word such as *très* has a [z] in its APP, relevant for liaison, and the noun *dent* has a [t], relevant only morphologically. Of course many words simply have an empty appendix, if there is no reason to postulate a latent consonant.

The feature LIAISON-FORM indicates whether or not a word realizes liaison—in other words, for a word with distinct long and short forms,

<sup>12</sup>The plural forms of adjectives also give rise to liaison, but in these cases the shape alternation is systematically of the simple type, involving the latent consonant [z]. Note also that there are four determiners (three possessives *ma/mon*, *ta/ton*, *sa/son*, and the singular demonstrative *ce/cet(te)*) which give rise to lexically-controlled alternations similar to those found with adjectives.

the LFORM value determines which one will be chosen as the phonological realization of the word. For words with only one shape (e.g., feminine singular adjectives), the value of LFORM has no consequence on the phonology. All singular nouns are lexically specified as [LFORM –], so that even though they may have a latent consonant in APP (e.g., *dent*), they do not have long forms in liaison contexts. Masculine singular adjective forms are lexically specified as either [LFORM +] (e.g., *vieil*, *bel*) or [LFORM –] (e.g., *vieux*, *beau*).<sup>13</sup> Finally, words with distinct short and long forms differing only in the realization or non-realization of a latent consonant (e.g., *très*, plural adjectives and nouns), are lexically underspecified for the feature LFORM. The contextually instantiated value of the feature will determine whether the appendix is realized or not (as explained in section 3.2.3).

### 3.2.2 Propagation

Up to now we have only seen how the attributes LTRIG, LFORM, and APP operate in lexical entries. But since liaison can occur between words that are not sisters in a local tree, we need to specify a mechanism for the propagation of these features in syntactic combinations so that the relevant liaison information is visible at the phrasal level. It is clear that this propagation is not uniformly head-driven, or indeed driven by any syntactic considerations; it depends only on the linear order of the daughters. If the first word in a phrase is vowel-initial, then of course the phrase itself is vowel-initial, and similarly if the last word in a phrase is a long form that must appear in a liaison context, then the phrase as a whole must appear in a liaison context. More formally, a dominating phrase will always have the same liaison trigger status (LTRIG value) as its left-most daughter, and the same liaison target status and latent consonant (LFORM and APP) as its right-most daughter. Our liaison features can therefore be treated as “edge features”, which have also been used for the formal analysis of phrasal affixes in French (Miller, 1992, Tseng, 2003b), and in an earlier HPSG treatment of liaison (Tseng, 2003a). The Edge Feature Principle (19) allows feature propagation along the right and left edges of phrases.<sup>14</sup>

---

<sup>13</sup>These values are part of the morphosyntactic properties regularly associated with the two relevant slots of adjectival paradigms; they are not stipulated word by word.

<sup>14</sup>We adopt an encoding of phrase structure in the spirit of Sag et al. (2003), whereby all daughters of a phrase (including the head daughter) are listed in the DTRS value, which is the locus of linear precedence constraints. Note that further work is needed to determine how the current analysis can be integrated with linearization-based analyses of French syntax (see e.g. Bonami, Godard, and Marandin, 1999).

## (19) Edge Feature Principle

$$\begin{array}{l}
 \text{a.} \\
 \text{phrase} \rightarrow \left[ \begin{array}{l} \text{LEFT } \boxed{1} \\ \text{DTRS } \langle [\text{LEFT } \boxed{1}] \rangle \oplus \text{list}(\text{sign}) \end{array} \right] \\
 \\
 \text{b.} \\
 \text{phrase} \rightarrow \left[ \begin{array}{l} \text{RIGHT } \boxed{1} \\ \text{DTRS } \text{list}(\text{sign}) \oplus \langle [\text{RIGHT } \boxed{1}] \rangle \end{array} \right]
 \end{array}$$

In combination with the feature geometry introduced in (18), liaison information appears correctly on all phrasal signs. (20) illustrates the percolation of features in a simple phrase.<sup>15</sup>

 (20) *amis intimes* ‘close friends’:

$$\left[ \begin{array}{l} \text{L} \left[ \begin{array}{l} \text{LTRIG } \boxed{3}+ \end{array} \right] \\ \text{R} \left[ \begin{array}{l} \text{LFORM } \boxed{4} \textit{bool} \\ \text{APP } \boxed{5} \langle \textit{z} \rangle \end{array} \right] \\ \\ \text{DTRS } \left\langle \left[ \begin{array}{l} \text{PHON } \langle \textit{ami} \rangle \\ \text{CAT } \textit{N} \\ \text{L} \left[ \begin{array}{l} \text{LTRIG } \boxed{3}+ \\ \text{LFORM } \textit{bool} \end{array} \right] \\ \text{R} \left[ \begin{array}{l} \text{APP } \langle \textit{z} \rangle \end{array} \right] \end{array} \right] , \left[ \begin{array}{l} \text{PHON } \langle \textit{\u00e9} \textit{tim} \rangle \\ \text{CAT } \textit{Adj} \\ \text{L} \left[ \begin{array}{l} \text{LTRIG } + \\ \text{LFORM } \boxed{4} \textit{bool} \end{array} \right] \\ \text{R} \left[ \begin{array}{l} \text{APP } \boxed{5} \langle \textit{z} \rangle \end{array} \right] \end{array} \right] \right\rangle \end{array} \right]$$

### 3.2.3 Phonological realization

We still need to explain formally how the various combinations of values of LFORM, LTRIG, and APP give rise to the characteristic phonological aspects of the liaison alternation. We define a function **dtrs-to-phon**, taking a list of signs and producing a list of phonological strings, for this purpose.<sup>16</sup>

$$(21) \text{sign} \rightarrow \left[ \begin{array}{l} \text{PHON } \text{dtrs-to-phon}(\boxed{\Sigma}) \\ \text{DTRS } \boxed{\Sigma} \end{array} \right]$$

<sup>15</sup>In this figure L abbreviates LEFT and R abbreviates RIGHT. In later figures the features LEFT and RIGHT are omitted, since no ambiguity can arise. In addition, we abbreviate CAT values to traditional category labels (V, VP, S, etc.)

<sup>16</sup>For the sake of concreteness, we treat phonological representations as lists of segment sequences, and we treat phonological combination as list concatenation. This extremely simplified view of phonology is sufficient for our purposes.

The first clause of the definition (22) takes care of the realization of liaison between two daughters in a phrase. If the first daughter is a liaison form ([LFORM +]) and the next daughter is a liaison trigger ([LTRIG +]), this clause adds both the PHON value of the first daughter and its APP (latent consonant, if any) to the phonology of the phrase. The recursive call to **dtrs-to-phon** specifies what must be done with the phonology of the remaining daughter(s).

$$(22) \quad \text{dtrs-to-phon} \left( \left\langle \left\langle \begin{bmatrix} \text{PHON} & \boxed{1} \\ \text{APP} & \boxed{2} \\ \text{LFORM} & + \end{bmatrix}, \boxed{3}[\text{LTRIG } +] \right\rangle \oplus \boxed{\Sigma} \right\rangle \right) \\ = \boxed{1} \oplus \boxed{2} \oplus \text{dtrs-to-phon} \left( \boxed{3} \oplus \boxed{\Sigma} \right)$$

The second clause of **dtrs-to-phon** (23) covers cases where liaison is not realized after the first daughter. This daughter's PHON value is incorporated into the phrasal phonology, its APP value is ignored, and **dtrs-to-phon** is called recursively to handle the rest of the list (which must be non-empty). Note that this clause does not check the LTRIG status of the following daughter. This is in accordance with our decision to treat optional liaison as the default situation: since liaison is optional in the general case, the absence of liaison does not need to be licensed by properties of the next daughter.

$$(23) \quad \text{dtrs-to-phon} \left( \left\langle \left\langle \begin{bmatrix} \text{PHON} & \boxed{1} \\ \text{LFORM} & - \end{bmatrix} \right\rangle \oplus \boxed{\Sigma} \text{ nelist}(\text{sign}) \right\rangle \right) \\ = \boxed{1} \oplus \text{dtrs-to-phon} \left( \boxed{\Sigma} \right)$$

Finally, the last sign of every phrase's DTRS list is handled by clause (24). No matter what the final sign's LFORM value is, we simply add its PHON value to the phrasal phonology, without appending the APP list, and the calculation of the phrasal phonology terminates.<sup>17</sup>

---

<sup>17</sup>Of course, the right-most daughter of a phrase can be [LFORM +], and it can have a non-empty APP, but clause (24) ignores these features. The information is not lost, however, because the phrase itself shares its LFORM and APP values with this right-most daughter, in accordance with the EFP (19). Consequently, when this phrase combines with other material to form a larger phrase, it will appear on a higher DTRS list, and its right-edge liaison features will either be taken into account there (see e.g. example (30)), or they will continue to propagate to the next higher phrase. We assume that complete utterances are always [LFORM -], so they can never end with a liaison form (like *vieil*), and the formulation of **dtrs-to-phon** ensures that a latent consonant is never realized at the end of a complete utterance (i.e. maximal DTRS list).

$$(24) \text{ dtrs-to-phon} \left( \left\langle \left[ \text{PHON } \boxed{1} \right] \right\rangle \right) = \boxed{1}$$

Going back to example (20), note that the value of LFORM on *amis* is underspecified. Thus there are two possible outputs of **dtrs-to-phon**: either *amis* is [LFORM +] and (22) applies, or it is [LFORM -] and (23) applies. This situation is typical of optional liaison contexts.<sup>18</sup>

$$(25) \text{ a. } \text{dtrs-to-phon} \left( \left[ \begin{array}{ll} \text{PHON} & \langle \text{ami} \rangle \\ \text{LFORM} & + \\ \text{APP} & \langle z \rangle \end{array} \right], \left[ \begin{array}{ll} \text{PHON} & \langle \tilde{\epsilon} \text{tim} \rangle \\ \text{LTRIG} & - \end{array} \right] \right) = \langle \text{ami}, z, \tilde{\epsilon} \text{tim} \rangle$$

$$\text{ b. } \text{dtrs-to-phon} \left( \left[ \begin{array}{ll} \text{PHON} & \langle \text{ami} \rangle \\ \text{LFORM} & - \\ \text{APP} & \langle z \rangle \end{array} \right], \left[ \begin{array}{ll} \text{PHON} & \langle \tilde{\epsilon} \text{tim} \rangle \\ \text{LTRIG} & - \end{array} \right] \right) = \langle \text{ami}, \tilde{\epsilon} \text{tim} \rangle$$

In the structurally identical case of (26), since the second element on DTRS is [LTRIG -], clause (22) cannot apply, and thus only one realization (with no liaison) is possible.

(26) *amis chers* [amiʃɛʁ] ‘dear friends’:

$$\left[ \begin{array}{l} \text{PHON} \quad \text{dtrs-to-phon}(\boxed{1}, \boxed{2}) = \langle \text{ami}, \text{ʃ}\epsilon\text{ʁ} \rangle \\ \text{LTRIG} \quad \boxed{3}+ \\ \text{LFORM} \quad \boxed{4} \\ \text{APP} \quad \boxed{5} \langle z \rangle \\ \\ \text{DTRS} \quad \left\langle \left[ \begin{array}{ll} \text{PHON} & \langle \text{ami} \rangle \\ \text{CAT} & \text{N} \\ \text{LTRIG} & \boxed{3}+ \\ \text{LFORM} & \textit{bool} \\ \text{APP} & \langle z \rangle \end{array} \right], \left[ \begin{array}{ll} \text{PHON} & \langle \text{ʃ}\epsilon\text{ʁ} \rangle \\ \text{CAT} & \text{Adj} \\ \text{LTRIG} & - \\ \text{LFORM} & \boxed{4} \\ \text{APP} & \boxed{5} \langle z \rangle \end{array} \right] \right\rangle \end{array} \right]$$

An important feature of the **dtrs-to-phon** function is that it decides whether liaison occurs not on the basis of the content of the APPENDIX, but on the basis of the LFORM feature. Thus words with an empty appendix can still be liaison forms licensed by clause (22) (prenominal

<sup>18</sup>Note that a property of our analysis is that the liaison consonant occurs as an autonomous, unsyllabified element on the PHON list. This leaves it to the phonology proper to determine where the liaison consonant is syllabified, in accordance with the well-known observation that both syllabifications (rightward and leftward) are possible (Encrevé, 1988), and that pauses may occur on either side of the liaison consonant (Morin and Kaye, 1982).

adjectives like *vieil*), and words with a non-empty appendix can fail to give rise to liaison (e.g., *dent*) because they are [LFORM –]. More generally, many (if not most) words both have an empty appendix and are underspecified for the LFORM feature (e.g., *frère*, *vrai*, *avec*). For these words, applying clause (22) (if allowed by the following DTRS element) or (23) leads to exactly the same result.

### 3.3 Constraining liaison contexts

In accordance with the observations in section 3.1, we now provide an account of contexts where liaison is obligatory or impossible. Remember that since the formulation of *dtrs-to-phon* treats optional liaison as the default case, nothing needs to be added to the grammar to account for the optional cases.

#### 3.3.1 Obligatory liaison

To account for obligatory liaison between specifier and head, we assume that the LFORM feature of the specifier must be identical to the LTRIG of the head (27). This makes the realization of liaison entirely dependent on the trigger status of the second daughter. As (28) illustrates, the effect of this specification is that *dtrs-to-phon* can only produce one result when the second daughter is [LTRIG +], unlike in optional liaison contexts (e.g., (20), (25))

$$(27) \textit{hd-spr-ph} \rightarrow \left[ \text{DTRS} \left\langle [\text{LFORM } \boxed{1}], [\text{LTRIG } \boxed{1}] \right\rangle \right]$$

(28) Obligatory liaison with latent consonant: *mes amis* [mezami]

$$\left[ \begin{array}{l} \textit{hd-spr-ph} \\ \text{PHON} \quad \textit{dtrs-to-phon}(\boxed{1},\boxed{2}) = \langle \textit{me}, \textit{z}, \textit{ami} \rangle \\ \text{LTRIG} \quad \boxed{3} - \\ \text{LFORM} \quad \boxed{4} \\ \text{APP} \quad \boxed{5} \langle \textit{z} \rangle \\ \\ \text{DTRS} \quad \left\langle \boxed{1} \left[ \begin{array}{l} \text{PHON} \quad \langle \textit{me} \rangle \\ \text{CAT} \quad \textit{Det} \\ \text{LTRIG} \quad \boxed{3} - \\ \text{LFORM} \quad \boxed{6} + \\ \text{APP} \quad \langle \textit{z} \rangle \end{array} \right], \boxed{2} \left[ \begin{array}{l} \text{PHON} \quad \langle \textit{ami} \rangle \\ \text{CAT} \quad \textit{N} \\ \text{LTRIG} \quad \boxed{6} + \\ \text{LFORM} \quad \boxed{4} \\ \text{APP} \quad \boxed{5} \langle \textit{z} \rangle \end{array} \right] \right\rangle \end{array} \right]$$

#### 3.3.2 Impossible liaison

We now turn to cases of impossible liaison. To block liaison in a particular context, it is sufficient to constrain the relevant sign to be

[LFORM -]. We illustrate first with subject-head combinations. The constraint in (29) forces both daughters in a head-subject phrase to be [LFORM -]. Thus when we combine the NP in (28) with a VP, the *head-subj-ph* type forces a [LFORM -] specification on the NP, irrespective of the VP's LTRIG value. This is illustrated with an [LTRIG +] VP in (30). The [LFORM -] specification on the second daughter in (29) blocks liaison between the whole clause and a further constituent, even though the clause has an appendix (which originates on the verb). This accounts for the observation in (14) that there can be no liaison between a subject-head combination and a following constituent.

$$(29) \textit{head-subj-ph} \rightarrow \left[ \text{DTRS} \left\langle [\text{LFORM -}], [\text{LFORM -}] \right\rangle \right]$$

(30) *Mes amis arrivent* [mezamiariv] ‘My friends are coming’

$$\left[ \begin{array}{l} \textit{hd-subj-ph} \\ \text{PHON} \quad \text{dtrs-to-phon}(\boxed{1}, \boxed{2}) = \langle \text{me, z, ami, aʁiv} \rangle \\ \text{LTRIG} \quad \boxed{3} - \\ \text{LFORM} \quad \boxed{4} - \\ \text{APP} \quad \boxed{5} \langle t \rangle \\ \\ \text{DTRS} \quad \left\langle \boxed{1} \left[ \begin{array}{l} \text{PHON} \quad \langle \text{me, z, ami} \rangle \\ \text{CAT} \quad \text{NP} \\ \text{LTRIG} \quad \boxed{3} - \\ \text{LFORM} \quad - \\ \text{APP} \quad \langle z \rangle \end{array} \right], \boxed{2} \left[ \begin{array}{l} \text{PHON} \quad \langle \text{aʁiv} \rangle \\ \text{CAT} \quad \text{VP} \\ \text{LTRIG} \quad + \\ \text{LFORM} \quad \boxed{4} - \\ \text{APP} \quad \boxed{5} \langle t \rangle \end{array} \right] \right\rangle \end{array} \right]$$

Head-filler phrases are subject to a constraint exactly parallel to that on head-subject phrases (31). For head-complement phrases, we need to account for the fact that liaison is possible after the head and lite complement daughters, but that it is impossible after nonlite complements. The constraint in (32) licenses this behavior (assuming that the head always left-most in head-complement structures in French). In addition, the whole head-complement phrase is [LFORM -], blocking liaison to the right of the whole phrase, in accordance with the observation in (16).

$$(31) \textit{head-fill-ph} \rightarrow \left[ \text{DTRS} \left\langle [\text{LFORM -}], [\text{LFORM -}] \right\rangle \right]$$

$$(32) \textit{head-comps-ph} \rightarrow \left[ \begin{array}{l} \text{LFORM} \quad - \\ \text{DTRS} \quad \langle [ ] \rangle \oplus \textit{nelist}([\text{WEIGHT lite}] \vee [\text{LFORM -}]) \end{array} \right]$$

### 3.4 Conclusion

In this paper we provided both new data illustrating the syntactic constraints on French liaison, and an HPSG grammar fragment that accounts for the observed distribution of obligatory, optional, and impossible liaison.

The account is incomplete in that it does not consider non-syntactic factors that make optional liaison more or less probable (in extreme cases, almost mandatory or almost impossible). Two approaches can be taken to integrate such data into the current analysis. One possibility is to leave the grammar as it is, and introduce a post-grammatical component to determine the likelihood of a particular liaison, taking into account lexical, prosodic, collocational, and other information. A second, more promising line of analysis is to make a more sophisticated use of the *dtrs-to-phon* function. Since this function operates on complete representations of the signs it combines, it has access to all the necessary information. Thus *dtrs-to-phon* could output, in addition to the phonological form of a phrase with or without liaison, an indication of the probability of actually realizing that particular option.

### References

- Abeillé, Anne and Danièle Godard. 2000. French word order and lexical weight. In R. D. Borsley, ed., *The Nature and Function of Syntactic Categories*, vol. 32 of *Syntax and Semantics*, pages 325–360. San Diego: Academic Press.
- Abeillé, Anne and Danièle Godard. 2002. The syntactic structure of French auxiliaries. *Language* 78:404–452.
- Bonami, Olivier and Gilles Boyé. 2003. La nature morphologique des allomorphes conditionnées. Les formes de liaison des adjectifs en français. In Fradin et al. (2003).
- Bonami, Olivier, Danièle Godard, and Jean-Marie Marandin. 1999. Constituency and word order in French subject inversion. In G. Bouma, E. Hinrichs, G.-J. Kruijff, and R. T. Oehrle, eds., *Constraints and Resources in Natural Language Syntax and Semantics*, pages 21–40. Stanford, CA: CSLI Publications.
- Bybee, Joan. 2001. Frequency effects on French liaison. In J. Bybee and P. Hopper, eds., *Frequency and the emergence of linguistic structure*, pages 337–359. Amsterdam: John Benjamins.
- Comorovski, Ileana. To appear. Quel. In F. Corblin and H. de Swart, eds., *Handbook of French Semantics*. Stanford, CA: CSLI Publications.
- de Jong, Daan. 1994. La sociophonologie de la liaison orléanaise. In C. Lyche, ed., *French Generative Phonology: Retrospective and Perspectives*, pages 95–130. Salford: AFLS/ESRI.

- Delattre, Pierre. 1966. *Studies in French and comparative phonetics*. The Hague: Mouton.
- Encrevé, Pierre. 1988. *La liaison avec et sans enchaînement*. Paris: Seuil.
- Fouché, Pierre. 1959. *Traité de prononciation française*. Paris: Klincksieck.
- Fougeron, Cécile, Jean-Philippe Goldman, Alicia Dart, Laurence Guélat and Clémentine Jeager. 2001. Influence de facteurs stylistiques, syntaxiques et lexicaux sur la réalisation de la liaison français. *Actes de TALN 2001*, pages 173–182.
- Fradin, Bernard, Georgette Dal, Françoise Kerleroux, Nabil Hathout, Marc Plénat, and Michel Roché, eds. 2003. *Les Unités morphologiques / Morphological Units*. Villeneuve d'Ascq: Silex.
- Miller, Philip H. 1992. *Clitics and Constituents in Phrase Structure Grammar*. New York: Garland.
- Miller, Philip H. and Ivan A. Sag. 1997. French clitic movement without clitics or movement. *Natural Language and Linguistic Theory* 15:573–639.
- Morin, Yves-Charles. 1998. Remarks on prenominal liaison consonants in French. Ms., Université de Montréal.
- Morin, Yves-Charles and Jonathan D. Kaye. 1982. The syntactic bases for French liaison. *Journal of Linguistics* pages 291–330.
- Post, Brechtje. 2000. Pitch accents, liaison and the phonological phrase in French. *Probus* 12:127–164.
- Sag, Ivan A., Thomas Wasow, and Emily M. Bender. 2003. *Syntactic Theory: a Formal Introduction*. Stanford, CA: CSLI Publications.
- Selkirk, Elisabeth O. 1972. *The phrase phonology of English and French*. Ph.D. thesis, MIT.
- Selkirk, Elisabeth O. 1974. French liaison and the  $\bar{X}$  convention. *Linguistic Inquiry* 5:573–590.
- Selkirk, Elisabeth O. 1986. On derived domains in sentence phonology. *Phonology Yearbook* 3:371–405.
- Steriade, Donka. 1999. Lexical conservatism in French adjectival liaison. In J.-M. Authier, B. E. Bullock, and L. Reed, eds., *Formal perspectives on Romance linguistics*, pages 243–270. Amsterdam: John Benjamins.
- Tranel, Bernard. 1981. *Concreteness in Generative Phonology: Evidence from French*. Berkeley: University of California Press.
- Tseng, Jesse. 2003a. Edge features and French liaison. In J.-B. Kim and S. Wechsler, eds., *Proceedings of the 9th International HPSG Conference*, pages 313–333. Stanford, CA: CSLI Publications.
- Tseng, Jesse. 2003b. Un traitement lexical des affixes syntagmatiques du français. In Fradin et al. (2003).



# On Induction of Morphology Grammars and its Role in Bootstrapping

DAMIR ČAVAR, JOSHUA HERRING, TOSHIKAZU IKUTA,  
PAUL RODRIGUES, GIANCARLO SCHREMENTI

Different Alignment Based Learning (ABL) algorithms have been proposed for unsupervised grammar induction, e.g. Zaanen (2001) and Déjean (1998), in particular for the induction of syntactic rules. However, ABL seems to be better suited for the induction of morphological rules. In this paper we show how unsupervised hypothesis generation with ABL algorithms can be used to induce a lexicon and morphological rules for various types of languages, e.g. agglutinative or polysynthetic languages. The resulting morphological rules and structures are optimized with the use of conflicting constraints on the size and statistical properties of the grammars, i. e. MINIMUM DESCRIPTION LENGTH and MINIMUM RELATIVE ENTROPY together with MAXIMUM AVERAGE MUTUAL INFORMATION. Further, we discuss how the resulting (optimal and regular) grammar can be used for lexical clustering/classification for the induction of syntactic (context free) rules.

## 4.1 Introduction

In previous approaches grammar induction algorithms consisted of three fundamental phases, see e.g. van Zaanen and Adriaans (2001), Zaanen (2001), Déjean (1998):

- Hypothesis generation

*Proceedings of Formal Grammar 2004.*

Gerhard Jäger, Paola Monachesi, Gerald Penn, and Shuly Winter (eds.).

Copyright © 2004, the individual authors.

- Hypothesis selection
- Induction

From the computational perspective the main problems lie on the generational and selectional level. Both of these components try to reduce the set of hypotheses about the structure of selected natural language input to the smallest possible amount that provides the best coverage given a targeted formalism or description level. That is, it tries to maximize relevant and minimize irrelevant hypotheses based on the need to reduce computational complexity and errors in the final induction phase. Thus, the art of grammar induction is to find the equilibrium between the amount of hypotheses generated and the effort invested to select the *best* candidates.

In what follows, we will discuss the results from investigations into unsupervised grammar induction algorithms that make use of *string alignment* for hypothesis generation driven purely by previous experience, or, in other words, by the lexicon and the hypotheses generated at every step in the incremental consumption and induction procedure. ABL is such an approach, see for example Zaanen (2001). Its proponents have thus far hesitated to recognize ABL as an approach that is attractive from computational as well as a cognitive perspectives. ABL constrains the hypothesis space from the outset to the set of hypotheses that are motivated by previous experience/input or a preexisting grammar. Such constraining characteristics make ABL attractive from a cognitive point of view, both because the computational complexity is reduced on account of the reduced set of potential hypotheses, and also because the learning of new items, rules, or structural properties is related to a general learning strategy and previous experience only. The approaches that are based on a brute-force first order explosion of hypotheses with subsequent filtering of relevant or irrelevant structures are both memory intensive and require more computational effort.

The basic concepts in ABL go back to notions of *substitutability* and/or *complementarity*, as discussed in Harris (1955) and Harris (1961). The concept of *substitutability* is used in the central part of the induction procedure itself, the assumption being that substitutable elements (e.g. substrings, words, structures) are assumed to be of the same type (represented e.g. with the same symbol). The notion of “same type” is not uncontroversial. Its use in syntax as a test for membership in a particular “part-of-speech” category, for example, was rightly criticized in Pinker (1994) and Chomsky (1955). However, it remains a rather reliable constituent test, and is certainly reliable in the sense in which it is understood in this paper. By “substitutabil-

ity,” we understand not so much an instrument to identify constituents of the same type but rather of a method of identifying constituents as such. The typing of constituents could be the result of independent components that use alignment information and other statistical properties.

Nevertheless, the ABL approach has disadvantages if not used properly. The size of the grammar can affect the runtime behavior of ABL systems, as can learning errors. In the following, we will describe our implementation and the use case, the problems and solutions for a grammar induction algorithm based on ABL.

While ABL is used in a slightly restricted way for hypothesis generation, we make use of different methods in the evaluation component to reduce the error rate and increase the performance of the algorithm, both with respect to the runtime behavior as well as the output quality. Interacting weighted constraints are used to increase the efficiency of the resulting grammar and eliminate irrelevant structural descriptions. In particular, the central constraints we use are:

- MAXIMUM AVERAGE MUTUAL INFORMATION (MI), which requires that the mutual information between segments of a structural description and the complete structural descriptions acquired so far (the hypotheses space for the induction procedure), is maximized. Hypotheses that maximally contribute to the average mutual information are preferred.
- MINIMUM RELATIVE ENTROPY (RE), which requires that the relative entropy for all resulting structural descriptions is minimized.
- MINIMUM DESCRIPTION LENGTH (MDL), which requires that the size of the resulting grammar (including structural descriptions) is minimized.

All constraints are used for evaluation and selection of the best hypotheses by taking into account the properties of the resulting grammar and the structural descriptions it generates. All the constraints seem to be well-motivated from a cognitive perspective, assuming that the cognitive resources are limited with respect to e. g. memory capacity and processing time. Grammar acquisition is seen as a compression process that emerges under memory and processing time restrictions, i. e. compressing the language input as much as possible while maintaining online usability. The compression ratio is limited by the processing capacities and time constraints imposed by language use. Unlimited creativity is thus seen here to be a side effect of complementary constraints of memory driven compression (grammar induction) and time and processing driven usability. The algorithm is parametrized along

these lines, allowing for fine grained restrictions of the runtime environment. This potentially allows us to test the impact of the different constraints on the resulting grammar.

One of the underlying research hypotheses here is also that a large amount of valuable syntactic information can be induced if information from other linguistic domains is used. That is, if large parts of the morphological, phonological or prosodic restrictions can be induced, assuming that these are to a great extent regular, we expect this information to be used as bootstraps to syntactic structure, assuming that this is mainly context free (or mildly context sensitive). In other words, the hypothesis here is that at least parts of context free grammar can be learned if regular grammars are used that describe or generate parts of natural language input. This is how we understand bootstrapping in grammar induction or natural language acquisition.

Thus, the design criteria for the algorithm presented here are computational, cognitive, and linguistic in nature, and though we assume that the algorithm can be used in virtually all linguistic domains (prosody, phonology, syntax), our concern here is mainly with the induction of morphological structure and any underlying rules that might (or might not) be used to describe such structure. The main research question is to what extent can we use this type of algorithm to induce morphological grammars that can then serve, together with the morphological terminals, as cues for syntactic structure and rules.

## 4.2 Specification of the algorithm

On the basis of the design criteria for the algorithm, as discussed above, a first implementation of the ABL-based induction algorithm was incremental. The algorithm was designed as an iterative procedure that consumes utterances, generates hypotheses for the morphological structure of every individual word in the utterance, and adds the most accurate hypotheses to the grammar and/or lexicon. The entire cycle of hypothesis generation, evaluation and induction is passed through for each word in the input.<sup>1</sup>

The variant of the ABL algorithm for morphology that we are using makes use of simple substring matching, described in further detail below. If a morpheme is found as a submorpheme in an input word, its edges are assumed to represent potential morpheme boundaries within that word. We apply the restriction that only words that occur as independent morphemes are used for this alignment based hypothesis

---

<sup>1</sup>A Python implementation and more detailed information is available at <http://jones.ling.indiana.edu/abugi/>.

generation.

In order to reduce the computational complexity of the algorithm, we take all generated rules and lexical entries as final if there is more than one occurrence of a similar pattern in the grammar. There is no revision of the made hypotheses that enter the final hypothesis space and are part of the structural descriptions (SD) that serve as input to the induction procedure, generated wrong SDs being expected to become statistically insignificant as the incremental grammar induction process runs.

As mentioned, the input to the algorithm is a list of words that is processed incrementally. That is, we make use of word boundaries, presupposing an existing segmentation of the input. For the purpose here, and to generate better results, we focus on the generation of morphological segmentation for a given set of words. In principle, however, there is no reason why the same algorithm, with minor refinements, couldn't be used to segment a raw string of alphanumeric characters into usable units separated by word boundaries. Indeed, we believe that a defining feature of our work, and one which distinguishes it from previous work, e. g. Goldsmith (2001), is that there is a concentrated attempt to eliminate all built-in knowledge from the system. The algorithm starts with a clean state and uses only statistical, numerical, and string matching techniques in an effort to remain as close as possible to a cognitive model, with a central focus on unsupervised induction. The main goal of this strategy is to identify the algorithms that allow for induction of specific linguistic knowledge, and to identify the possibly necessary supervision for each algorithm type.

Thus we assume for the input:

- Alphabet: a non-empty set  $A$  of  $n$  symbols  $\{s_1, s_2, \dots s_n\}$
- Word: a word  $w$  a non-empty list of symbols  $w = [s_1, s_2, \dots s_n]$ , with  $s \in A$
- Corpus: a non-empty list  $C$  of words  $C = [w_1, w_2, \dots w_n]$

The output of the ABL hypothesis generation is a set of hypotheses for a given input word. A hypothesis is a tuple:

- $H = \langle w, f, g \rangle$ , with  $w$  the input word,  $f$  its frequency in  $C$ , and  $g$  a list of substrings that represent a linear list of morphemes  $i$   $w$ ,  $g = [m_1, m_2, \dots m_n]$

The hypotheses are collected in a hypotheses space. The hypothesis space is defined as a list of hypotheses:

- Hypotheses space:  $S = [H_1, H_2, \dots H_n]$

The algorithm does not make any assumptions about types of morphemes. There is no expectation of specific structure in the input, nor does it use notions like *stem*, *prefix*, or *suffix*. We assume only linear sequences. The fact that single morphemes exist as stems or suffixes is a side effect of their statistical properties (including Frequency and left and right Pointwise Mutual Information, a term that will be explained below) and alignment within the corpus, or rather within words.

There are no language specific rules built-in, such as what a morpheme must contain or how frequent it should be. All of this knowledge is learned, based on statistical analysis of prior experience. However, as discussed in the last section, at certain points in the learning procedure we lose the performance benefit of not relying on such rules to escape linguistic and statistical anomalies that might lead the program astray.

Each iteration of the incremental learning process consists of the following steps:

1. ABL Hypotheses Generation
2. Hypotheses Evaluation and Selection
3. Grammar Extension

In the ABL Hypotheses Generation, a given word in the utterance is checked against all the morphemes in the grammar. If an existing morpheme  $m$  aligns with the input word  $w$ , a hypothesis is generated suggesting a morphological boundary at the alignment positions:

$$w(\textit{speaks}) + m(\textit{speak}) = H[\textit{speak}, s] \quad (4.1)$$

Another design criterion for the algorithm is complete language independence. It should be able to identify morphological structures of Indo-European type of languages, as well as agglutinative languages (e.g. Japanese and Turkish) and polysynthetic languages like some Bantu dialects or Native American languages like Lakhota. In order to guarantee this behavior, we extended the Alignment Based hypothesis generation with a pattern identifier that extracts patterns of character sequences of the types:

1. A – B – A
2. A – B – A – B
3. A – B – A – C

This component is realized with cascaded finite state transducers that are able to identify and return the substrings that correspond to the repeating sequences.<sup>2</sup>

---

<sup>2</sup>This addition might be understood to be a sort of *supervision* in the system.

All possible alignments for the existing grammar at the current state are collected and evaluated. The Hypothesis Evaluation and Selection step uses a set of different criteria to find the best hypotheses. The following evaluation criteria are used:

- Maximization of Pointwise Mutual Information between the morphemes
- Minimization of the Relative Entropy for the resulting grammar
- Minimization of the Description Length for the resulting grammar
- Minimization of the number of morphemes
- Maximization of the length of morphemes
- Maximization of the frequency of a morpheme boundary over all ABL Hypotheses

Each of these criteria is weighted relative to the others. While the different choices for the relative weights of the criteria were partially arbitrary in our experiments,<sup>3</sup> the choice of criteria is not.

The criteria are related to assumptions we make about cognitive aspects of language and grammar. Specifically, we assume that the properties of natural language grammars are constrained by limited memory resources resulting in a preference for smaller grammars which are maximally efficient in terms of run time, computational complexity and memory space consumption. We employ an interaction of MDL, Maximum MI and Minimum RE to reach an optimally-sized and efficient grammar. We relate efficiency to Information Theoretic notions of coding length, channel capacity and transmission time, as well as symbol replacement operations for the processing and generation of natural language utterances. Thus, indirectly the number of morphemes and their length is related to usability aspects, since the number of morphemes is related to the number of symbols used in induced rules, and thus to the number of replacement operations in processing and generation. Along these lines we group the above listed evaluation constraints into memory and usability oriented constraints.

The choice of evaluation criteria is also influenced by the expectation that languages will differ with respect to the importance of particular constraints at specific linguistic levels. The well-known correlation between richness of morphology and restrictiveness of word order as well

---

However, as shown in recent research on human cognitive abilities, and especially on the ability to identify patterns in the speech signal by very young infants Marcus et al. (1999), we can assume such an ability to be part of the general cognitive endowment, maybe not even language specific.

<sup>3</sup>Currently we are working on automatic adaptation of these weights during the learning process. This is potentially the locus of a self-supervision strategy, as also pointed out by one reviewer.

as the quantitative correlation between the number of words per utterance and the number of morphemes per word is expected to be due to different weights on a set of constraints that natural languages are subject to.

In the following sections the components of the evaluation module are described in more detail.

#### 4.2.1 Mutual Information (MI)

For the purpose of this experiment we use a variant of standard Mutual Information (MI), see Charniak (1996) and MacKay (2003) for some use cases. Information theory tells us that the presence of a given morpheme restricts the possibilities of the occurrence of morphemes to the left and right, thus lowering the amount of bits needed to store its neighbors. Thus we should be able to calculate the amount of bits needed by a morpheme to predict its right and left neighbors respectively. To calculate this, we have designed a variant of mutual information that is concerned with a single direction of information.

This is calculated in the following way. For every morpheme  $y$  that occurs to the right of  $x$  we sum the pointwise MI between  $x$  and  $y$ , but we relativize the pointwise MI by the probability that  $y$  follows  $x$ , given that  $x$  occurs. This then gives us the expectation of the amount of information that  $x$  tells us about which morpheme will be to its right. Note that  $p(\langle x, y \rangle)$  is the probability of the bigram  $\langle x, y \rangle$  occurring and is not equal to  $p(\langle y, x \rangle)$  which is the probability of the bigram  $\langle y, x \rangle$  occurring.

We calculate the MI on the right side of  $x \in G$  by:

$$\sum_{y \in \{\langle x, Y \rangle\}} p(\langle x, y \rangle | x) \lg \frac{p(\langle x, y \rangle)}{p(x)p(y)} \quad (4.2)$$

and the MI on the left of  $x \in G$  respectively by:

$$\sum_{y \in \{\langle Y, x \rangle\}} p(\langle y, x \rangle | x) \lg \frac{p(\langle y, x \rangle)}{p(y)p(x)} \quad (4.3)$$

One way we use this as a metric, is by summing up the left and right MI for each morpheme in a hypothesis. We then look for the hypothesis that results in the maximal value of this sum. The tendency for this to favor hypotheses with many morphemes is countered by our criterion of favoring hypotheses with fewer morphemes, a topic we will discuss in greater detail below.

Another way to use the left and right MI is in judging the quality of morpheme boundaries. In a good boundary, the morpheme on the left

side should have high right MI and the morpheme on the right should have high left MI. Unfortunately, MI is not initially very reliable because of the low frequency of many words, and removing hypotheses with poor boundaries prevents the algorithm from bootstrapping itself as all boundaries are poor in the beginning. We are currently experimenting with phasing this in as MI is deemed more reliable in making these judgments.

#### 4.2.2 Description Length (DL)

The principle of Minimum Description Length (MDL) as used in recent work on grammar induction and unsupervised language acquisition, e. g. Goldsmith (2001), Marcken (1996), and Grünwald (1998), explains the grammar induction process as an iterative minimization procedure of the grammar size, where the smaller grammar corresponds to the *best* grammar for the given data/corpus.

The description length metric, as we use it here, tells us how many bits of information would be required to store a word given a hypothesis of the morpheme boundaries, using our grammar. For each morpheme in the hypothesis that doesn't occur in the grammar we need to store the string representing the morpheme. For morphemes that do occur in our grammar we just need to store a pointer to that morpheme's entry in the grammar. We use a simplified calculation, taken from Goldsmith (2001), of the cost of storing a string that takes the number of bits of information required to store a letter of the alphabet and multiply it by the length of the string.

$$\lg(\text{length}(A)) * \text{len}(\text{morpheme}) \quad (4.4)$$

We have two different methods of calculating the cost of the pointer. The first takes a cue from Morse code and gives a variable cost depending on the frequency of the morpheme that it is pointing to. So first we calculate the frequency rank of the morpheme being pointed to, (e. g. the most frequent has rank 1, the second rank 2, etc.). We then calculate:

$$\text{floor}(\lg(\text{freqrank}) - 1) \quad (4.5)$$

to get a number of bits similar to the way Morse code assigns lengths to various letters.

The second is simpler and only calculates the entropy of the grammar of morphemes and uses this as the cost of all pointers to the grammar. The entropy equation is as follows:

$$\sum_{x \in G} p(x) \lg \frac{1}{p(x)} \quad (4.6)$$

The second equation doesn't give variable pointer lengths, but it is preferred since it doesn't carry the heavy computational burden of calculating the frequency rank.

We calculate the description length for each hypothesis individually by summing up the cost of each morpheme in the hypothesis. Those with low description lengths are favored. Note that we do not calculate the sizes of the grammars with and without any given hypothesis. The computational load of the algorithm is thus significantly reduced; by calculating only the relative increase for every suggested hypothesis and favoring the hypothesis with the smallest increase in overall description length, a large number of potentially wasteful computational steps are avoided. In subsequent versions of the algorithm the description length will be calculated on the basis of the resulting grammars after the induction step.

### 4.2.3 Relative Entropy (RE)

We use RE as a measure for the cost of adding a hypothesis to the existing grammar. We look for hypotheses that, when added to the grammar, will result in minimal divergence from the original.

We calculate RE as a variant of the Kullback-Leibler Divergence, see e. g. Charniak (1996) and MacKay (2003). Given grammar  $G_1$ , the grammar generated so far, and  $G_2$ , the grammar with the extension generated for the new input increment,  $P(X)$  is the probability mass function (*pmf*) for grammar  $G_2$ , and  $Q(X)$  the *pmf* for grammar  $G_1$ :

$$\sum_{x \in X} P(x) \lg \frac{P(x)}{Q(x)} \quad (4.7)$$

Note that with every new iteration a new element can appear that is not part of  $G_1$ . Our variant of RE takes this into account by calculating the costs for such a new element  $x$  to be the point-wise entropy of this element in  $P(X)$ , summing up over all new elements:

$$\sum_{x \in X} P(x) \lg \frac{1}{P(x)} \quad (4.8)$$

These two sums then form the RE between the original grammar and the new grammar with the addition of the hypothesis. Hypotheses with low RE are favored.

This metric behaves similarly to description length, discussed above,

in that both calculate the distance between our original grammar and the grammar with the inclusion of the new hypothesis. The primary difference is that RE also takes into account how the probability mass function differs between the two grammars and that our variation punishes new morphemes based upon their frequency relative to the frequency of other morphemes. MDL does not consider frequency in this way, which is why we include RE as metric. We are currently investigating this to identify under what conditions they behave differently.

#### 4.2.4 Further Metric

In addition to the mentioned metric, we take into account the following criteria:

- Frequency of Morpheme Boundaries
- Number of Morpheme Boundaries
- Length of Morphemes

The frequency of morpheme boundaries is given by the number of hypotheses that contain this boundary. The basic intuition is that the higher this number, i. e. the more alignments are found at a certain position within a word, the more likely this position represents a morpheme boundary. We favor hypotheses with high values for this criterion.

To prevent the algorithm from degenerating into a state where each letter is identified as a morpheme, we favor hypotheses with lower numbers of morpheme boundaries. For this same reason, we also take morpheme length into account, preferring hypotheses with longer morphemes (again, to avoid running into a situation where every letter in a word is taken to be morphologically significant).

#### 4.2.5 Pre-grammar generation

With every successful evaluation of hypotheses a set of signatures for each morpheme is generated, similar to the approach suggested in Goldsmith (2001). An example signature is given in the following, where the symbol  $X$  represents the slot for the respective morpheme left of the arrow.

- `clock`  $\rightarrow$   $[[['X', 's$'], 1], [['X'], 1]]$

The signature contains the possible realizations of the word *clock*, either with the suffix *s* or alone. Each possible realization contains the count of total occurrences of the respective word form.

With every evaluation result, the potential hypotheses are evaluated on the basis of the existing grammar by calculating the likelihood of the new potential signature, given the existing signatures in the grammar.

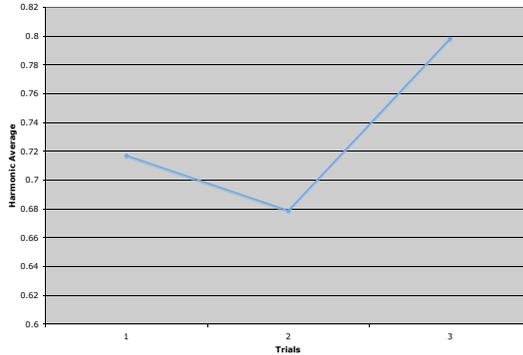
A hypothesis that fits into the general pattern found in the grammar is preferred.

Grammar generation is performed by replacement of all words with equal signatures with a symbol and merger of all signatures to one. The resulting grammar represents the basis for calculations of the description length. Further, the resulting signatures are used to derive type information for the respective morphemes, as described below.

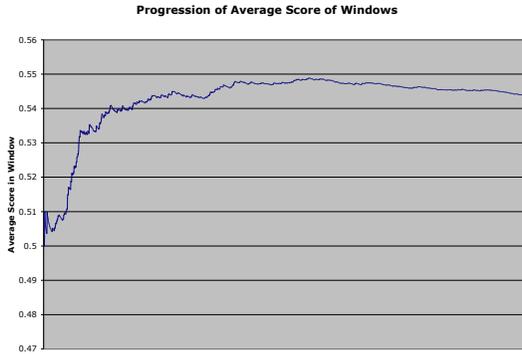
### 4.3 Evaluation

We used two methods to evaluate the performance of the algorithm. The first analyzes the accuracy of the morphological rules produced by the algorithm after an increment of  $n$  words. The second looks at how accurately the algorithm parsed each word that it encountered as it progressed through the corpus.

The first analysis looks at each grammar rule generated by the algorithm and judges it on the correctness of the rule and the resulting parse. A grammar rule consists of a stem and the suffixes and prefixes that can be attached to it, similar to the signatures used in Goldsmith (2001). The grammar rule was then marked as to whether it consisted of legitimate suffixes and prefixes for that stem and also as to whether the stem of the rule was a true stem, as opposed to a stem plus another morpheme that wasn't identified by the algorithm. The number of rules that were correct in these two categories were then summed, and precision and recall figures were calculated for the trial. The trials described in the graph below were run on three increasingly large portions of the general fiction section of the Brown Corpus. The first trial was run on one randomly chosen chapter, the second trial on two chapters, and the third trial run on three chapters. The graph shows the harmonic average (F-score) of precision and recall.



The second analysis is conducted as the algorithm is running and examines each parse the algorithm produces. The algorithm's parses are compared with the correct morphological parse of the word using the following method to derive a numerical score for a particular parse. The first part of the score is the distance in characters between each morphological boundary in the two parses, with a score of one point for each character apart in the word. The second part is a penalty of two points for each morphological boundary that occurs in one parse and not the other. These scores were examined within a moving window of words that progressed through the corpus as the algorithm ran. The average scores of words in each such string of words were calculated as the window advanced. The purpose of these windows was to allow the performance of the algorithm to be judged at a given point without prior performance in the corpus affecting the analysis of the current window. The following chart shows how the average performance of the windows of analyzed words as the algorithm progresses through five randomly chosen chapters of general fiction in the Brown Corpus amounting to around 10,000 words. The window size for the following chart was set to 40 words.



We are currently performing detailed testing of the algorithm on Esperanto and Japanese corpora. The highly regular morphology of Esperanto should provide an interesting comparison against the fractured morphology of English. Likewise, the agglutinative nature of Japanese should provide a fertile test bed for morphological analysis.

The primary experiments conducted to date have been performed using the Brown Corpus of Standard American English, consisting of 1,156,329 words from American texts printed in 1961 organized into 59,503 utterances and compiled by W.N. Francis and H. Kucera at Brown University.<sup>4</sup>

#### 4.4 Conclusion

The algorithm generates very good structures for the initial input, achieving, under certain settings, a precision of up to 100% (meaning that it returns a wordlist consisting entirely of “usable” words). Recall was significantly less accurate, but still respectable, scoring in the 60% range on the settings that reached 100% precision. It will have been noted in the graph provided above that the process is also quite stable and improves steadily (if slowly) over time, never falling even temporarily behind.

Our main focus in this project was to derive the necessary type

---

<sup>4</sup>Additional experiments were done each for Classical Japanese and Esperanto. The Japanese experiment used a roman-character version of “Genji Monogatari” (The Tales of Genji), compiled by Prof. Eichi Shibuya of Takachiho University. Due to the highly regular (and pervasive) nature of the morphology, Esperanto provided an interesting frame for comparison. Tests were conducted on two corpora compiled from the Internet.

information for words that can be used in the induction of syntactic structures. As discussed in Elghamry and Čavar (2004), the type information can be used in a cue-based learning system to derive higher-level grammatical rules, up to the level of syntactic frames. The high levels of precision achieved suggest that errors in the input will not be a barrier in this next step. Using the morphological information discovered here, it should be possible to induce word types based on their morphological signatures (in context). The main concern would be whether the algorithm generates results with high enough recall to provide a sufficient amount of information on which to base such an induction. The recall numbers achieved in our experiments strongly suggest that it does.

The weights of the system are not fixed and can be adjusted to increase recall, decreasing precision. This might be of relevance for other domains and applications of this approach.

Ongoing studies with different language types will help us in the development of the necessary self-supervision component, especially in the adaptation of the weights of the evaluation constraints during runtime. Given the post-evaluation component that evaluates the relevance of signatures for words, we are already able to predict that certain weights should be reduced, specifically those that are responsible for the generation of irrelevant hypotheses. More results will be available after detailed evaluation on data from agglutinative and synthetic or polysynthetic languages.

## References

- Charniak, Eugene. 1996. *Statistical language learning*. Cambridge, Mass.: MIT Press, 1st edn.
- Chomsky, Noam. 1955. *The Logical Structure of Linguistic Theory*. Distributed by Indiana University Linguistics Club. Published in part by Plenum, 1975.
- Déjean, Hervé. 1998. *Concepts et algorithmes pour la découverte des structures formelles des langues*. doctoral dissertation, Université de Caen Basse Normandie.
- Elghamry, Khaled and Damir Čavar. 2004. Bootstrapping cues for cue-based bootstrapping. Mscr. Indiana University.
- Goldsmith, John. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics* 27(2):153–198.
- Grünwald, Peter. 1998. *The Minimum Description Length Principle and Reasoning under Uncertainty*. doctoral dissertation, Universiteit van Amsterdam.
- Harris, Zellig S. 1955. From phonemes to morphemes. *Language* 31(2):190–222.

- Harris, Zellig S. 1961. *Structural linguistics*. Chicago: University of Chicago Press. Published in 1951 under title: *Methods in structural linguistics*.
- MacKay, David J. C. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge: Cambridge University Press.
- Marcken, Carl G. de. 1996. *Unsupervised Language Acquisition*. Phd dissertation, Massachusetts Institute of Technology.
- Marcus, G. F., S. Vijayan, S. Bandi Rao, and P. M. Vishton. 1999. Rule-learning in seven-month-old infants. *Science* 283:77–80.
- Pinker, Steven. 1994. *The language instinct*. New York, NY: W. Morrow and Co.
- van Zaanen, Menno M. and Pieter Adriaans. 2001. Comparing two unsupervised grammar induction systems: Alignment-based learning vs. emile. Tech. Rep. TR2001.05, University of Leeds.
- Zaanen, Menno M. van. 2001. *Bootstrapping Structure into Language: Alignment-Based Learning*. Doctoral dissertation, The University of Leeds.

# Bidirectional Optimality for Regular Tree Languages

STEPHAN KEPSEK

## 5.1 Introduction

Optimality theory (OT henceforth) has been introduced by Prince and Smolensky (1993) originally as a model for generative phonology. In recent years, this approach has been applied successfully to a range of syntactic phenomena, and it is currently gaining popularity in semantics and pragmatics as well. It is based on the idea that a mapping from one level of linguistic representation to another should be described in terms of rules and filters. The novel contribution of OT is that filters – or, synonymously, constraints – are ranked and violable. Thus the result of a rule-based generation process may still be acceptable although it violates certain constraints as long as other results violate more constraints or constraints that are higher ranked.

In other words, the rules generate a set of candidates that are competitors. On this set, the constraints are applied in the order of their ranking starting with the highest ranked constraint. A candidate may violate a constraint more than once. The application of the highest ranked constraint assigns each candidate the number of violations of that constraint. Some of the candidates are now optimal with respect to this constraint in the sense that they violate the constraint the fewest times. These, and only these, are retained for the next round of constraint application. In each round, the current constraint is applied to the set of candidates remaining from the previous rounds. And only those candidates that are optimal with respect to the current constraint

make it into the next round. In the end, after applying all constraints, a set of candidates is reached which is optimal with respect to the given ranking of the constraints. The method is therefore comparable to a high jump competition in athletics.

Frank and Satta (1998) show that certain classes of OT-systems can be handled by finite state techniques. Their approach is influenced by ideas from computational phonology, the original field of application for OT. In this view, the generation of candidates is a relation on strings, and this relation is defined by a finite state transducer. In order to also render constraints by finite state automata, two restrictions have to be made. The first one is that constraints have to be binary, that is to say, each constraint assigns each candidate either 0 or 1. The second restriction demands constraints to be *output* constraints. An output constraint is a constraint that assigns a number to a candidate pair purely on the base of its output. Under these restrictions, constraints can be rendered as regular string languages over the output. The aim of the paper by Frank and Satta (1998) is to provide a modularity result for the complexity of an OT-system in the following sense. Suppose that the set of candidates is given by a finite state transducer and all constraints are expressible by regular languages. Then the whole OT-system can be rendered by finite state techniques and is no more complex than its components. The success of the approach by Frank and Satta is based on well-known closure properties of regular string languages.

The work by Frank and Satta was extended into two diverging directions. Based on the observation that natural language syntax and semantics have trees as their underlying data structures and not strings Wartena (2000) and Kepser and Mönnich (2003) propose ways to extend the approach by Frank and Satta to tree languages. Wartena (2000) shows that the original results for strings can be extended straight forwardly to trees, since the closure properties for regular string languages needed by Frank and Satta also hold for regular tree languages. Observing that there are certain non-regular phenomena in some natural languages (see, e.g., (Shieber, 1985)) Kepser and Mönnich (2003) extend the result by Wartena to linear context-free tree languages. In their work, the generator is split into a source for input trees defined by means of a linear context-free tree grammar and a relation between input trees and output trees defined by a linear tree transducer. Constraints are defined by monadic second-order formulae or, equivalently, regular tree languages. Their modularity result is based on closure properties of linear context-free tree languages also shown in that paper.

The second direction concerns the notion of optimality. All of the above described approaches are *unidirectional* in the sense that they describe ways to find optimal output for a given input. This view is apparently generation driven. Blutner (2000) points out that in particular in semantics and pragmatics unidirectional optimality may not suffice. The optimal interpretation of an utterance is obtained by an interplay between the generation process on the speaker side and the parsing process on the hearer side. Blutner therefore introduces the notion of *bidirectional* optimality theory. Formal properties of bidirectional OT are studied by Jäger (2002, 2003). He shows that the modularity result of Frank and Satta extends to bidirectional OT-systems of regular string languages. Jäger (2003) also shows that for bidirectional OT-systems the restriction to binary constraints is essential to achieve the modularity result.

Jäger states that the construction for bidirectional optimality of string languages in the earlier paper extends to bidirectional optimality of regular tree languages, if some automaton or tree transducer representing the Cartesian product of two regular tree languages can be provided. Actually, the proofs Jäger presents are that general in nature that they need not change for the case of regular tree languages. The present paper closes this gap. Cartesian products of regular tree languages can be defined by means of so-called tree tuple automata (Comon et al., 1997). And tree tuple automata can be integrated into the finite state construction used to compute bidirectionally optimal pairs. Thus the modularity result by Jäger (2002) for bidirectional optimality extends to the case of regular *tree* languages.

For obvious reasons the present paper follows Jäger (2002) very tightly quoting it frequently.

## 5.2 Preliminaries

### Regular Tree Grammars

A *ranked alphabet* (or *ranked operator domain*)  $\Sigma$  is an indexed family  $\langle \Sigma_n \rangle_{n \in \mathbb{N}}$  of disjoint sets. A symbol  $f$  in  $\Sigma_n$  is called an *operator of rank  $n$* . If  $n = 0$ , then  $f$  is also called a constant. For a ranked alphabet  $\Sigma$ , the set of *trees over  $\Sigma$*  (or  $\Sigma$ -*trees* or *terms over  $\Sigma$* ), denoted  $T_\Sigma$  is the smallest set of strings over  $\Sigma \cup \{ (, ) \}$  such that  $\Sigma_0 \subseteq T_\Sigma$  and, for  $n \geq 1$ , if  $f \in \Sigma_n$  and  $t_1, \dots, t_n \in T_\Sigma$ , then  $f(t_1, \dots, t_n) \in T_\Sigma$ . A subset of  $T_\Sigma$  is called a *tree language over  $\Sigma$* .

Any set  $M$  can be interpreted as a signature in which all of the elements of  $M$  are constants. Thus  $T_{\Sigma \cup M}$  denotes the set of trees over  $\Sigma$  and  $M$ . Let  $X = \{x_1, x_2, x_3, \dots\}$  be an infinite set (of variables). Then

$T_\Sigma(x_1, \dots, x_n) = T_{\Sigma \cup \{x_1, \dots, x_n\}}$  denotes the set of trees over  $\Sigma$  and additional variables  $\{x_1, \dots, x_n\}$ . From the point of view of a signature, a “variable” is a constant. If  $t \in T_\Sigma(X)$  then we also write  $t(x_1, \dots, x_n)$  to indicate that the variables of  $t$  are a subset of the set  $\{x_1, \dots, x_n\}$ . Let  $\Sigma$  and  $\Omega$  be two signatures, let  $t(x_1, \dots, x_n) \in T_\Sigma(x_1, \dots, x_n)$ , and let  $t_1, \dots, t_n \in T_\Omega$ . Then  $t(t_1, \dots, t_n) \in T_{\Sigma \cup \Omega}$  is the result of simultaneously replacing each occurrence of  $x_j$  in  $t(x_1, \dots, x_n)$  by  $t_j$  (with  $1 \leq j \leq n$ ).

Now we define the notion of a regular tree grammar.

**Definition 1** A *regular tree grammar* is a quadruple  $G = (\Sigma, \mathcal{F}, S, P)$  where

- $\Sigma$  is a finite ranked alphabet of *terminals*,
- $\mathcal{F}$  is a finite set of *nonterminals* or *function symbols*, disjoint with  $\Sigma$ ,
- $S \in \mathcal{F}$  is the *start symbol*, and
- $P$  is a finite set of productions (or rules) of the form  $F \rightarrow t$ , where  $F \in \mathcal{F}$  and  $t \in T_{\Sigma \cup \mathcal{F}}$ .

For a regular tree grammar  $G = (\Sigma, \mathcal{F}, S, P)$  the derivation relation is given as follows. Let  $s_1, s_2 \in T_{\Sigma \cup \mathcal{F}}$ . We say  $s_1 \Rightarrow s_2$  if and only if there is a production  $F \rightarrow t$  and  $F$  is a leaf node of  $s_1$ . Tree  $s_2$  is obtained from  $s_1$  by replacing  $F$  with the tree  $t$ . As usual,  $\overset{*}{\Rightarrow}$  stands for the reflexive-transitive closure of  $\Rightarrow$ . For a regular tree grammar  $G$ , we define  $L(G) = \{t \in T_\Sigma \mid S \overset{*}{\Rightarrow} t\}$ .  $L(G)$  is called the *tree language* generated by  $G$ .

### Tree Automata and Tree Transducers

For regular tree languages there exists an automaton model that corresponds to finite state automata for regular string languages. Let  $\Sigma$  be a signature. A *frontier-to-root tree automaton* is a triple  $\mathcal{A} = (Q, F, \delta)$  where  $Q$  is a finite set of *states*,  $F \subseteq Q$  a set of *final* states and  $\delta$  is a finite transition relation. Each transition has the form

$$f(q_1, \dots, q_n) \rightarrow q$$

where  $f \in \Sigma_n$ ,  $q, q_1, \dots, q_n \in Q$ . On an intuitive level, a frontier-to-root tree automaton labels the nodes in a tree with states starting from the leaves and going to the root. Suppose  $n$  is a node in the tree and  $f$  is the  $k$ -ary function symbol at node  $n$  and the  $k$  daughters of  $n$  are already labelled with states  $q_1, \dots, q_k$ , and furthermore  $f(q_1, \dots, q_k) \rightarrow q$  is a transition of  $\mathcal{A}$ , then node  $n$  can be labelled with state  $q$ . A tree is accepted if the root can be labelled with a final state.

We will now report some results about the theory of regular tree languages. For more information, consult the work by Gécsecz and Steinby

(1984, 1997). A tree language  $L$  is regular if and only if there is a tree automaton that accepts  $L$ . Regular tree languages are closed under union, intersection, and complement. There are corresponding constructions for tree automata. And finally, for every tree automaton accepting language  $L$  there exists a tree automaton also accepting  $L$  which has a single final state.

Tree automata can be generalised to automata that transform one tree into another, so-called tree transducers. The following exposition on tree transduction is taken from (Gécseg and Steinby, 1997). Let  $\Sigma$  and  $\Omega$  be two signatures. A binary relation  $\tau \subseteq T_\Sigma \times T_\Omega$  is called a *tree transformation*. A pair  $(s, t) \in \tau$  is interpreted to mean that  $\tau$  may transform  $s$  into  $t$ . We can speak of compositions, inverses, domains, and ranges of tree transformations as those of binary relations. We will now define frontier-to-root tree transducers.

**Definition 2** A *frontier-to-root tree transducer* (or F-transducer) consists of a quintuple  $\mathcal{A} = (\Sigma, \Omega, Q, P, F)$  where  $\Sigma$  and  $\Omega$  are signatures;  $Q$  is a finite set of *states*, each element of  $Q$  is a unary function;  $F \subseteq Q$  is the set of *final states*; and  $P$  is a finite set of *productions* of the following type:

$$f(q_1(x_1), \dots, q_m(x_m)) \rightarrow q(t(x_1, \dots, x_m))$$

where  $f \in \Sigma_m$ ,  $q_1, \dots, q_m, q \in Q$ ,  $t(x_1, \dots, x_m) \in T_\Omega(x_1, \dots, x_m)$ .

The transformation induced by an F-transducer is defined as follow. We write  $QT_\Omega$  for the set  $\{q(t) \mid q \in Q, t \in T_\Omega\}$  and regard  $QT_\Omega$  as a set of constants. Let  $s, t \in T_{\Sigma \cup QT_\Omega}$  be two trees. It is said that  $t$  can be obtained by a direct derivation from  $s$  in  $\mathcal{A}$  iff  $t$  can be obtained from  $s$  by replacing an occurrence of a subtree  $f(q_1(t_1), \dots, q_m(t_m))$  (with  $f \in \Sigma_m, q_1, \dots, q_m \in Q, t_1, \dots, t_m \in T_\Omega$ ) in  $s$  by  $q(t(t_1, \dots, t_m))$ , where  $f(q_1(x_1), \dots, q_m(x_m)) \rightarrow q(t(x_1, \dots, x_m))$  is a production from  $P$ . If  $s$  directly derives  $t$  in  $\mathcal{A}$  then we write  $s \Rightarrow_{\mathcal{A}} t$ . The reflexive transitive closure  $s \Rightarrow_{\mathcal{A}}^* t$  is the derivation relation.

Intuitively, an F-transducer traverses a tree  $s$  from the leaves to the root rewriting it at the same time. In a single derivation step we consider a node  $n$  in  $s$  with label  $f$  where all the daughter nodes are already transformed into trees of  $T_\Omega$  and each daughter node is in some state  $q_i$ . Then we replace the subtree of node  $n$  with the tree  $t$  from the production where the place holder variables of  $t$  are replaced by the trees of the daughter nodes of  $n$ . The root of this subtree is put into state  $q$ .

The relation

$$\tau_{\mathcal{A}} = \{(s, t) \mid s \in T_\Sigma, t \in T_\Omega, s \Rightarrow_{\mathcal{A}}^* q(t) \text{ for some } q \in F\}$$

is the transformation relation induced by  $\mathcal{A}$ . A relation  $\tau \subseteq T_\Sigma \times T_\Omega$  is an *F-transformation* if there exists an F-transducer  $\mathcal{A}$  such that  $\tau = \tau_{\mathcal{A}}$ . For a tree language  $L \subseteq T_\Sigma$  we define  $\mathcal{A}(L) = \{t \in T_\Omega \mid \exists s \in L \text{ with } (s, t) \in \tau_{\mathcal{A}}\}$ .

A production  $f(q_1(x_1), \dots, q_m(x_m)) \rightarrow q(t(x_1, \dots, x_m))$  is called *linear* if each variable  $x_1, \dots, x_m$  occurs at most once in  $t$ . An F-transducer is linear if each production is linear. We denote a linear F-transducer by LF-transducer. We will make use of the following results about LF-transducers.

**Proposition 1** *LF-transducers are closed under composition.*

*The classes of regular tree language is closed under LF-transductions. The domain and range of an LF-transducer is a regular tree language. For every regular tree language  $L$  there is an LF-transducer  $\iota$  such that  $\text{Dom}(\iota) = \text{Rng}(\iota) = L$  and  $\iota$  is the identity on  $L$ .*

These results can be found, e.g., in (Gécseg and Steinby, 1984, 1997). If  $\mathcal{A}_1$  and  $\mathcal{A}_2$  are two LF-transducers, we write  $\mathcal{A}_1 \circ \mathcal{A}_2$  for the composition of first  $\mathcal{A}_1$  and then  $\mathcal{A}_2$ . Let  $L_1$  and  $L_2$  be two tree languages. A binary relation  $R \subseteq L_1 \times L_2$  is called *rational* if there exists an LF-transducer  $\mathcal{A}$  such that  $\tau_{\mathcal{A}} = R$ .

LF-transducers define relations between tree languages where there is a strong connection between the input trees and the output trees. Obviously, an output tree is constructed on the base of a structural decomposition of the input tree. As stated in the introduction, we also need an automaton or a transducer representation of the Cartesian product of arbitrary regular tree languages. In a Cartesian product, every element of the input tree language is related to every element of the output tree language. Hence there is in general no structural relation between an input tree and an output tree. Therefore LF-transducers are not capable of defining the Cartesian product of two regular tree languages, as already observed by Jäger (2002).

But the Cartesian product of two (or more) regular tree languages can be defined by means of *tree tuple automata*. Comon et al. (1997, Sect. 3.2) describe three classes of tree tuple automata two of which can be used for the definition of a Cartesian product. The simplest class, which suffices already for our purposes, is based on pairs of automata. Let  $L_1$  and  $L_2$  be two regular tree languages, and let  $\mathcal{A}_1$  be a bottom-up tree automaton accepting  $L_1$ , and  $\mathcal{A}_2$  a bottom-up tree automaton accepting  $L_2$ . Define for any two trees  $s, t$  that  $(s, t) \in (\mathcal{A}_1, \mathcal{A}_2)$  iff  $s \in \mathcal{A}_1$  and  $t \in \mathcal{A}_2$ . Then, obviously,  $(\mathcal{A}_1, \mathcal{A}_2)$  defines the relation  $L_1 \times L_2$ . This relation is called  $\text{Rec}_\times$  in (Comon et al., 1997).

A more powerful definition of tree tuple automata is given, if the re-

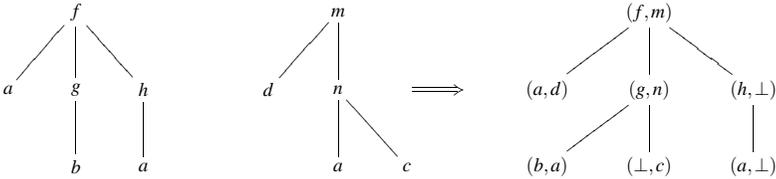


FIGURE 1 Coding two trees in a single one.

lation is expressed by a single automaton instead of a pair of automata. To make this approach work, two trees have to be coded in a single one. Let  $\Sigma$  and  $\Omega$  be two signatures. We define trees with labels as pairs of  $(\Sigma \cup \{\perp\} \times \Omega \cup \{\perp\})$  where  $\perp$  is a new (padding) symbol not contained in  $\Sigma$  or  $\Omega$ . For a pair of trees  $(s, t) \in (T_\Sigma \times T_\Omega)$  we define inductively their coding  $[s, t]$  by

$$[f(t_1, \dots, t_n), g(u_1, \dots, u_m)] = \begin{cases} (f, g)([t_1, u_1], \dots, [t_m, u_m], [t_{m+1}, \perp], \dots, [t_n, \perp]) & \text{if } n \geq m \\ (f, g)([t_1, u_1], \dots, [t_n, u_n], [\perp, u_{n+1}], \dots, [\perp, u_m]) & \text{if } m \geq n \end{cases}$$

Basically, the union of the tree domains is constructed and the nodes are labelled with pairs of labels. If one tree lacks a branch, then the padding symbol  $\perp$  is used. See Figure 1 for an example.

Now,  $Rec$  is the set of relations  $R \subseteq T_\Sigma \times T_\Omega$  such that  $\{(s, t) \mid (s, t) \in R\}$  is accepted by a bottom-up tree automaton on the signature  $(\Sigma \cup \{\perp\} \times \Omega \cup \{\perp\})$ .

We quote some results on tree tuple automata (see Comon et al., 1997).  $Rec_\times$  is strictly included in  $Rec$ .  $Rec_\times$  and  $Rec$  are closed under union, intersection and complement.

There exists an extended definition of tree transducers by Comon et al. (1997), who introduce  $\epsilon$ -rules. Extended tree transducers are strictly more powerful than standard tree transducers as defined above in the sense that every relation definable by a standard tree transducer is definable by an extended tree transducer. But there are relations definable by an extended tree transducer that cannot be defined by a standard tree transducer. These extended tree transducers, however, are not capable of defining arbitrary Cartesian products of regular tree languages. Therefore they cannot be used to substitute the above defined tree tuple automata.

### Monadic Second-Order Logic

Monadic second-order logic (MSO) is an extension of first-order logic by set variables and quantification over set variables. MSO is quite

a powerful logic. Many graph theoretical properties can be expressed in MSO, for example that a graph is a proper tree. Courcelle (1990) showed that if a relation is definable in MSO, then so is its transitive closure. For tree languages, it is known that a tree language is regular iff it is definable by an MSO formula (see, e.g., Gécseg and Steinby, 1984). Thus an MSO formula can be translated into a tree automaton accepting the same tree language.

### 5.3 Optimality Theory

We now turn to optimality theory. Let us start by making the notions of optimality theory more precise. In the general case, an OT-system consists of a binary relation **GEN** and a finite set of constraints that are linearly ordered. Constraints may be violated several times. So a constraint should be construed as a function from **GEN** into the natural numbers. Thus an OT-system assigns each candidate pair from **GEN** a sequence of natural numbers. The ordering of the elements of **GEN** that is induced by the OT-system is the lexicographic ordering of these sequences.

**Definition 3** An *OT-system* is a pair  $(\mathbf{GEN}, C)$  where **GEN** is a binary relation and  $C = \langle c_1, \dots, c_p \rangle, p \in \mathbb{N}$ , is a linearly ordered sequence of functions from **GEN** to  $\mathbb{N}$ . Let  $a, b \in \mathbf{GEN}$ . We say  $a$  is more economical than  $b$  ( $a < b$ ), if there is a  $k \leq p$  such that  $c_k(a) < c_k(b)$  and for all  $j < k : c_j(a) = c_j(b)$ .

Intuitively, an output  $o$  is optimal for some input  $i$  iff **GEN** relates  $o$  to  $i$  and  $o$  is optimal amongst the possible outputs for  $i$ . This is the notion of unidirectional optimality, which is obviously generation-driven. Bidirectional optimality reflects the fact that in semantics and pragmatics the relation between input (a form) and output (a meaning) can and perhaps should be regarded as an interplay between parsing optimality and generation optimality. Hence Jäger (2002), formalising ideas by Blutner (1998, 2000), defines *bidirectional* optimality as follows.

**Definition 4** A form-meaning pair  $(f, m)$  is *bidirectionally optimal* iff

1.  $(f, m) \in \mathbf{GEN}$ ,
2. there is no bidirectionally optimal  $(f', m)$  such that  $(f', m) < (f, m)$ ,
3. there is no bidirectionally optimal  $(f, m')$  such that  $(f, m') < (f, m)$ .

Thus, checking whether a form-meaning pair is bidirectionally optimal requires simultaneous evaluation of form alternatives and meaning al-

ternatives of this pair. This definition is not circular in cases where the ordering of pairs is well-founded. As was shown by Jäger (2002), the ordering of pairs given by the definition of an OT-system is indeed well-founded. Hence bidirectional optimality of OT-systems is not a circular notion.

The type of constraints considered in the literature on finite state OT and also used here is not quite as general as Definition 3 insinuates. Firstly, constraints have to be binary, i.e., a constraint assigns a candidate pair either the number 0 or 1. This restriction is not quite as severe as it may seem. Every constraint with an upper bound on the number of violations can be translated into a sequence of binary constraints.

Secondly, so-called markedness constraints are considered only. A markedness constraint is a constraint that either evaluates solely the input or solely the output.

**Definition 5** Let  $(\mathbf{GEN}, (c_1, \dots, c_p))$  be an OT-system.

A constraint  $c_j$  is an *output markedness constraint* iff  $c_j(i, o) = c_j(i', o)$  for all  $(i, o), (i', o) \in \mathbf{GEN}$ .

A constraint  $c_j$  is an *input markedness constraint* iff  $c_j(i, o) = c_j(i, o')$  for all  $(i, o), (i, o') \in \mathbf{GEN}$ .

To gain a better understanding of how bidirectional optimality is evaluated on markedness constraints in a (finite) OT-system consider this example by Jäger (2002). The following text is a direct quote from (Jäger, 2002, p. 441f).

“Suppose  $\mathbf{GEN} = \{1, 2, 3\} \times \{1, 2, 3\}$ , and we have two constraints which both say ‘Be small!’ One of its instances applies to the input and one to the output. Thus formally we have

- $\mathcal{O} = (\mathbf{GEN}, C)$ .
- $\mathbf{GEN} = \{1, 2, 3\} \times \{1, 2, 3\}$ .
- $C = \langle c_1, c_2 \rangle$ .
- $c_1(\langle i, o \rangle) = i$ .
- $c_2(\langle i, o \rangle) = o$ .

It follows from the way constraints are evaluated that  $\langle i_1, o_1 \rangle <_{\mathcal{O}} \langle i_2, o_2 \rangle$  iff  $i_1 \leq i_2, o_1 \leq o_2$ , and  $\langle i_1, o_1 \rangle \neq \langle i_2, o_2 \rangle$ . Now obviously  $\langle 1, 1 \rangle$  is bidirectionally optimal since both its input and its output obey the constraints in an optimal way. Accordingly,  $\langle 1, 2 \rangle, \langle 2, 1 \rangle, \langle 1, 3 \rangle$ , and  $\langle 3, 1 \rangle$  are blocked, since they all share a component with a bidirectionally optimal candidate. There are still candidates left which are neither marked as optimal nor as blocked, so we have to repeat this procedure. Amongst the remaining candidates,  $\langle 2, 2 \rangle$  is certainly bidirectionally

optimal because all of its competitors in either dimension are known to be blocked. This candidate in turn blocks  $\langle 2, 3 \rangle, \langle 3, 2 \rangle$ . The only remaining candidate,  $\langle 3, 3 \rangle$ , is again bidirectionally optimal since all its competitors are blocked.<sup>1</sup> This example illustrates the general strategy for the finite case: Find the cheapest input-output pairs in the whole of **GEN** and mark them as bidirectionally optimal. Next mark all candidates that share either the input or the output component (but not both) with one of these bidirectionally optimal candidates as blocked. If there are any candidates left that are neither marked as bidirectionally optimal nor as blocked, repeat the procedure until **GEN** is exhausted.” And this ended the quote.

The construction of an OT-system for tree languages looks in principle as follows. The generation relation **GEN** is expressed by an LF-transducer, i.e., it is a rational relation. The constraints are expressed by MSO-sentences either on the input or on the output. As stated before, an MSO-sentence can be translated into a regular tree language and an LF-transducer. It is important to note that a constraint in OT is violable, and this is also true for binary constraints. If some candidates fulfil the constraint, then all others are filtered out. But if all candidates miss the constraint, all of them pass through, because no candidate is relatively better than the others. Frank and Satta (1998) provide a construction called conditional intersection that gives a transducer implementing this violability. Wartena (2000) shows how to extend this construction to trees for unidirectional optimality. And Jäger (2002) provides the extension for optimality theory on regular string languages.

The proposal by Jäger (2002) can be transferred to regular tree languages in a rather direct manner. Let  $R$  be a rational relation and  $L \subseteq \text{Rng}(R)$  be a regular tree language. The conditional intersection  $R \uparrow L$  is defined as

$$R \uparrow L := (R \circ \iota_L) \cup (\iota_{\text{Dom}(R) - \text{Dom}(R \circ \iota_L)} \circ R).$$

For an input markedness constraint represented by  $L \subseteq \text{Dom}(R)$  we set dually

$$R \downarrow L := (\iota_L \circ R) \cup (R \circ \iota_{\text{Rng}(R) - \text{Rng}(\iota_L \circ R)}).$$

These intersections relate individually optimal input and output pairs. But in bidirectional optimality we are looking for globally optimal pairs. Hence bidirectional intersection is defined as follows. Let  $R$  be a rational

---

<sup>1</sup>Bidirectional optimality thus predicts iconicity: the pairing of cheap inputs with cheap outputs is optimal, but also the pairing of expensive inputs with expensive outputs. See Blutner’s (1998, 2000) papers for further discussion of this point.

relation and  $c$  a binary markedness constraint. Let  $*$  be an arbitrary constant, i.e., a tree consisting of a single node, the root, labelled with  $*$ . Then

$$R \uparrow c := \begin{cases} R \circ \iota_{Rng(\{\{*\} \times Rng(R)\} \uparrow c)} & \text{if } c \text{ is an output} \\ & \text{markedness constraint} \\ \iota_{Dom((Dom(R) \times \{*\}) \downarrow c)} \circ R & \text{else} \end{cases}$$

The construction works as follows.  $\{*\} \times Rng(R)$  relates the regular tree language  $\{*\}$  to any possible output of  $R$ , which is also a regular tree language. Conditional intersection with  $c$  gives

$$((\{*\} \times Rng(R)) \circ \iota_c) \cup (\iota_{\{*\} - Dom((\{*\} \times Rng(R)) \circ \iota_c)} \circ (\{*\} \times Rng(R))).$$

By definition, this is equal to

$$(\{*\} \times (Rng(R) \cap c)) \cup ((\{*\} - Dom(\{*\} \times (Rng(R) \cap c))) \times Rng(R)).$$

Since tree tuple automata are closed under union and their domains and ranges are obviously regular tree languages, the construction is well defined. It is defined in such a way that either the left hand side or the right hand side of the  $\cup$  is the empty relation, depending on whether  $Rng(R) \cap c$  is empty or not.

If  $Rng(R) \cap c$  is non-empty, then the above reduces to  $\{*\} \times (Rng(R) \cap c)$ . If  $Rng(R) \cap c$  is empty, then this reduces to  $\{*\} \times Rng(R)$ . In either way,  $Rng(\{*\} \times Rng(R)) \uparrow c$  is the set of outputs of  $R$  that are optimal with respect to  $c$ , and it is a regular tree language. Thus  $R \uparrow c$  is a rational relation, and it is the set of pairs  $(i, o) \in R$  that are optimal with respect to  $c$ . If  $c$  is an input markedness constraint, the dual statements are true.

**Lemma 2** *Let  $\mathcal{O} = (\mathbf{GEN}, C)$  be an OT-system, where  $C = (c_1, \dots, c_p)$  is a sequence of binary markedness constraints. Then*

$$(i, o) \in \mathbf{GEN} \uparrow c_1 \dots \uparrow c_p$$

*iff  $(i, o) \in \mathbf{GEN}$  and there is no  $(i', o') \in \mathbf{GEN}$  such that  $(i', o') < (i, o)$ .*

The proof for this lemma is identical to the proof of Lemma 3 on page 443 of (Jäger, 2002).

The application of a sequence of constraints  $C = (c_1, \dots, c_p)$  to a rational relation  $R$  can be seen as an operator  $C(R) := R \uparrow c_1 \dots \uparrow c_p$  filtering out the globally optimal pairs. As a result,  $R$  is partitioned into three sets:  $C(R)$ , the set  $B$  of pairs that share one component with a pair of  $C(R)$  (but not both), and the set  $U$  of pairs that share no component with a pair of  $C(R)$ . The pairs that share one component with a pair in  $C(R)$  are blocked, they cannot be bidirectionally optimal. But no statement can be made about the pairs in  $U$ . Some of them

may be bidirectionally optimal, some may not. Thus, similarly to the toy example given before, the filtering procedure has to be applied to  $U$ . And this step of filtering and finding the unmarked pairs has to be iterated till  $R$  is exhausted.

**Definition 6** Let  $\mathcal{O} = (\mathbf{GEN}, C)$  be an OT-system. Define

$$\begin{aligned} X_0 &:= \emptyset, \\ X_{n+1} &:= X_n \cup C( \iota_{\text{Dom}(\mathbf{GEN}) - \text{Dom}(X_n)} \circ \mathbf{GEN} \\ &\quad \circ \iota_{\text{Rng}(\mathbf{GEN}) - \text{Rng}(X_n)} ), \\ X &:= \bigcup_{n \geq 0} X_n. \end{aligned}$$

For every natural number  $n$ ,  $X_{n+1}$  adds those pairs to  $X_n$  that are not blocked by  $X_n$  and optimal. Hence,  $X$  is the set of all bidirectionally optimal pairs.

**Lemma 3** Let  $\mathcal{O} = (\mathbf{GEN}, C)$  be an OT-system. Then  $(i, o) \in X$  iff  $(i, o) \in \mathbf{GEN}$  and  $(i, o)$  is bidirectionally optimal.

The proof of this lemma is given as the proof of Lemma 4, page 444 of (Jäger, 2002).

As Jäger (2002) remarks, the operation  $X_n$  is a cumulative definition of bidirectional optimality. And on the assumption that  $\mathbf{GEN}$  and  $X_n$  are rational relations and the constraints are binary markedness constraints expressible as regular tree languages, the relation  $X_{n+1}$  is again a rational relation, i.e., expressible by finite state means. Since  $\emptyset = \emptyset \times \emptyset$  is a rational relation on trees, bidirectional optimality of regular tree languages can be computed with finite state techniques provided the iteration of exhausting  $\mathbf{GEN}$  terminates after a finite number of steps, i.e., there is a  $k \in \mathbb{N}$  such that  $X = X_k$ .

**Lemma 4** Let  $\mathcal{O} = (\mathbf{GEN}, C)$  be an OT-system with  $C = (c_1, \dots, c_p)$  where all  $c_i$  are binary markedness constraints. Then  $X = X_{2^p-1}$ .

The proof of this lemma is given as the proof of Lemma 5, page 447 of (Jäger, 2002).

The following main theorem integrates the observations made in this section.

**Theorem 5** Let  $\mathcal{O} = (\mathbf{GEN}, C)$  be an OT-system with  $C = (c_1, \dots, c_p)$  where all  $c_i$  are binary markedness constraints. Furthermore let  $\mathbf{GEN}$  be a rational relation and all  $c_i$  be MSO-sentences. Then the set of bidirectionally optimal elements of  $\mathbf{GEN}$  is again a rational relation.

## 5.4 Conclusion

This paper extends the approach by Jäger (2002) for bidirectional optimality from regular string languages to regular tree languages. We have shown that if the generator of an OT-system consists of a linear frontier-to-root transducer and the constraints are expressed by MSO-sentences on either the input or the output, then the OT-system can be rendered by finite state tree automata and tree transducers. This implies that the complexity of the whole OT-system is not larger than the complexity of its most complex components.

The most important difference of the construction for tree languages as compared to the one for string languages can be found in the fact that we need two types of finite state devices for tree languages. Finite state string transducers are capable of expressing the Cartesian product of two regular string languages. Therefore the construction for the computation of optimal pairs needs finite state string transducers only. In the case of tree language, the situation is different. Cartesian products of arbitrary regular tree languages cannot be defined by means of LF-transducers, not even if one allows for  $\epsilon$ -rules. Therefore we introduced tree tuple automata into the construction. On the other hand there exist relations of regular tree languages definable by LF-transducers – rational relations, as we called them – that cannot be defined by means of tree tuple automata. These are in particular relations where there is an intricate relationship between the input and the output trees. Thus tree tuple automata cannot replace the LF-transducers, and we need indeed both types of finite state tree devices.

A interesting and important question is whether this approach can be extended to more complex generation relations that are based context-free instead of regular tree grammars. These extensions seem rather difficult. It looks like there is a way to compute the globally optimal pairs using LF-transducers. But whether the recursion step involved in bidirectional optimality (the definition of the  $X_n$ ) can be rendered by finite state means is doubtful.

## References

- Blutner, Reinhard. 1998. Lexical pragmatics. *Journal of Semantics* 15:115–162.
- Blutner, Reinhard. 2000. Some aspects of optimality in natural language interpretation. *Journal of Semantics* 17:189–216.
- Comon, Hubert, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. 1997. Tree automata techniques and applications. Available at:

<http://www.grappa.univ-lille3.fr/tata>. Release October, 1st 2002.

- Courcelle, Bruno. 1990. Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, vol. B, chap. 5, pages 193–242. Elsevier.
- Frank, Robert and Giorgio Satta. 1998. Optimality theory and the generative complexity of constraint violability. *Computational Linguistics* 24:307–315.
- Gécseg, Ferenc and Magnus Steinby. 1984. *Tree Automata*. Budapest: Akademiai Kiado.
- Gécseg, Ferenc and Magnus Steinby. 1997. Tree languages. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages, Vol 3: Beyond Words*, pages 1–68. Springer-Verlag.
- Jäger, Gerhard. 2002. Some notes on the formal properties of bidirectional optimality theory. *Journal of Logic, Language, and Information* 11:427–451.
- Jäger, Gerhard. 2003. Recursion by optimization: On the complexity of bidirectional optimality theory. *Natural Language Engineering* 9(1):21–38.
- Kepser, Stephan and Uwe Mönnich. 2003. A note on the complexity of optimality theory. In G. Scollo and A. Nijholt, eds., *Proceedings Algebraic Methods in Language Processing (AMiLP-3)*, pages 153–166.
- Prince, Alan and Paul Smolensky. 1993. Optimality theory: Constraint interaction in generative grammar. Tech. Rep. RuCCTS-TR 2, Rutgers University.
- Shieber, Stuart. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy* 8:333–343.
- Wartena, Christian. 2000. A note on the complexity of optimality systems. In R. Blutner and G. Jäger, eds., *Studies in Optimality Theory*, pages 64–72. University of Potsdam. Also available at Rutgers Optimality Archive as ROA 385-03100.

---

# Grammatical Framework and Multiple Context-Free Grammars

PETER LJUNGLÖF

Grammatical Framework (GF) (Ranta, 2004) is a grammar formalism originating from logical frameworks for dependent type theory. It is already known that the parsing problem for GF is undecidable, which has to do with the possibility to formulate undecidable propositions. But for subclasses of GF, in particular GF with a context-free backbone, parsing is decidable. Until now the parsing complexity of context-free GF has been unknown, which we aim to change with this article.

We show that there is a simple one-to-one correspondence between Grammatical Framework with context-free backbone and Multiple Context-Free Grammars (MCFG) (Seki et al., 1991). Since the parsing complexity for MCFGs is known to be polynomial in the length of the input, we get the same result for context-free GF.

GF with dependent types is undecidable in general, but the separation of abstract and concrete syntax makes it possible to use a two-step process in parsing. First a MCFG parser is used to create a chart, then each item in the chart is converted to a Horn clause. The resulting logic program then solves the parsing problem.

## 6.1 Grammatical Framework

Grammatical Framework (Ranta, 2004) is a grammar formalism built upon a Logical Framework. What GF adds to the logical framework is a possibility to define concrete syntax, that is, notations expressing formal concepts in user-readable ways. Although GF grammars are

*Proceedings of Formal Grammar 2004.*

Gerhard Jäger, Paola Monachesi, Gerald Penn, and Shuly Winter (eds.).

Copyright © 2004, the individual authors.

bidirectional, the perspective of GF is on generation rather than parsing. A difference from usual grammar formalisms is the support for multilinguality; it is possible to define several concrete syntaxes upon one abstract syntax. The abstract syntax then works as an interlingua between the concrete syntaxes. The development of GF as an authoring system started as a plug-in to the proof editor ALF, to permit natural-language rendering of formal proofs (Hallgren and Ranta, 2000). The extension of the scope outside mathematics was made in the Multilingual Document Authoring project at Xerox (Dymetman et al., 2000). In continued work, GF has been used in areas like software specifications (Hähnle et al., 2002) and dialogue systems (Ranta and Cooper, 2004). In this article we have changed the notation somewhat, compared to other GF articles. The reason for this is to make the comparison with Generalized Context-Free Grammars simpler. There are also some features of GF that we do not mention, mainly syntactic sugar and some default notations.

### 6.1.1 GF with a context-free backbone

Grammatical Framework is a strongly typed functional language; functional in the sense that a grammar defines a set of functions, and strongly typed in the sense that each function has a given typing, known at compile-time. Basic types in GF are called *categories*, and are specified by the grammar. For now we assume that there are only a finite number of categories – in section 6.1.3 we generalize the concept to dependent and higher-order categories.

A function is specified by a typing, an (optional) definition and a linearization. To retain compositionality, all terms of a given category  $A$  must linearize to the same *linearization type*, written  $\llbracket A \rrbracket$ .

**Function typings** The typing of a function tells us how many arguments it takes, what their categories are and what category the result is.

$$f : A_1 \rightarrow \dots \rightarrow A_\delta \rightarrow A$$

The function may take no arguments; in which case it is called a *ground term*. Given a grammar, we define the abstract terms, or syntax trees, as follows. The term  $f(t_1, \dots, t_\delta)$  is of category  $A$  whenever each  $t_i$  is of category  $A_i$ .

**Abstract definitions** An abstract definition specifies a computation rule on terms, thereby defining a notion of equality between terms. This equality is a semantic equality, since it does not affect the concrete linearizations; even though two terms are equal by definition, they can be linearized to different strings. Since they do not affect the concrete

syntax, we do not mention them further in this article.

**Linearization terms and types** Linearizations are written as terms in a typed functional programming language, which is limited to ensure decidability in generation and in parsing. The language has records and finite-domain functions (called tables); and the basic types are terminal lists (called strings) and finite data types (called parameter types). There are also local definitions, lambda-abstractions and global function definitions. The parameters are declared by the grammar; they can be hierarcical but not recursive, to ensure finiteness.

**Linearization definitions** The linearization of a function term  $f$  is a function (also written  $f$ ) from the linearization types of the argument categories to the linearization type of the result category; or as a functional typing,  $f : \llbracket A_1 \rrbracket \rightarrow \dots \rightarrow \llbracket A_\delta \rrbracket \rightarrow \llbracket A \rrbracket$ .

$$f(x_1, \dots, x_\delta) = \phi$$

The linearization  $\phi$  must of course be of type  $\llbracket A \rrbracket$  (written  $\phi : \llbracket A \rrbracket$ ), given that each  $x_i$  is of type  $\llbracket A_i \rrbracket$ . Note that a limitation on the linearization is that it is not possible to examine the structure of variables by e.g. case analysis; which is one way of defining compositionality for grammars.

### 6.1.2 Canonical GF

The concrete syntax of any GF grammar can be partially evaluated to a grammar in canonical form, as shown in Ranta (2004). In canonical form, all local and global definitions disappear, as well as function applications; and all tables are fully expanded. Hierarcical parameters can be flattened; thus we can assume that the parameters are declared by giving a finite set  $\text{Par}$  of parameter types, each  $P \in \text{Par}$  being a set of parameters  $\mathbf{p}_1, \dots, \mathbf{p}_n$ . A linearization term in canonical GF is of the following form:

- A string constant is of type  $\text{Str}$ ; and a concatenation  $s_1 \cdot s_2 : \text{Str}$ , whenever  $s_1, s_2 : \text{Str}$ .
- A constant parameter  $\mathbf{p} : P$ , whenever  $\mathbf{p} \in P$ .
- A record  $\{ r_1 = \phi_1 ; \dots ; r_n = \phi_n \}$  is of type  $T = \{ r_1 : T_1 ; \dots ; r_n : T_n \}$ , whenever each  $\phi_i : T_i$ .
- A record projection  $\phi.r_i : T_i$ , whenever  $\phi$  is of the record type  $T$  above.
- A table  $[\mathbf{p}_1 \Rightarrow \phi_1 ; \dots ; \mathbf{p}_n \Rightarrow \phi_n]$  is of type  $P \Rightarrow T$ , whenever  $P = \{ \mathbf{p}_1, \dots, \mathbf{p}_n \}$ , and each  $\phi_i : T$ .
- A table selection  $\phi! \psi : T$ , whenever  $\phi : P \Rightarrow T$  and  $\psi : P$ .
- An argument variable  $x_i : \llbracket A_i \rrbracket$ .

$$\begin{aligned}
\llbracket S \rrbracket &= \{ s : \text{Str} \} \\
\llbracket N \rrbracket &= \{ s : \text{Str}; n : \text{Num} \} \\
\llbracket V \rrbracket &= \{ s : \text{Num} \Rightarrow \text{Str} \} \\
p &: N \rightarrow V \rightarrow S \\
p(x, y) &= \{ s = x.s \cdot y.s \mid (x.n) \} \\
q &: V \rightarrow N \rightarrow V \\
q(x, y) &= \{ s = [n \Rightarrow x.s \mid n \cdot y.s] \} \\
a : N &= \{ s = \text{“animals”}; n = \text{Pl} \} \\
b : N &= \{ s = \text{“Bernie”}; n = \text{Sg} \} \\
l : V &= \{ s = [\text{Sg} \Rightarrow \text{“loves”}; \text{Pl} \Rightarrow \text{“love”}] \}
\end{aligned}$$

FIGURE 1 Example GF grammar

Together with this there are computation rules for string concatenation, record projection and table selection.

**An example grammar** In figure 1, there is a simple example GF grammar, which is not entirely in canonical form. The linearization of  $q$  contains a non-expanded table  $[n \Rightarrow x.s \mid n \cdot y.s]$ , whose canonical form is  $[\text{Sg} \Rightarrow x.s \mid \text{Sg} \cdot y.s; \text{Pl} \Rightarrow x.s \mid \text{Pl} \cdot y.s]$ .

### 6.1.3 GF with dependent categories

Full GF have a much more expressive abstract syntax than above; categories can depend on other categories. E.g. the grammar could specify that category  $A$  depends of category  $B$ , meaning that  $A(x)$  is a category whenever  $x$  is a term of category  $B$ .

Another extension is that arguments to function typings (and dependent categories) can be functions and not just of a basic category.

These extensions turn the abstract syntax into a logical framework, where e.g. undecidable propositions can be formulated, thus giving a very expressive formalism. A natural question is then how to parse and linearize terms with these extensions, which will be addressed in section 6.4.1.

## 6.2 Generalized CFG

Generalized Context-Free Grammars (GCFG) were introduced by Pollard in the 80's as a way of formally describing Head Grammars (Pollard, 1984). Later people have used GCFG as a framework for describing many other formalisms, such as Linear Context-Free Rewriting Systems (Vijay-Shanker et al., 1987) and Multiple Context-Free Grammars (Seki

et al., 1991); and here we will use it to describe GF with a context-free backbone.

There are several definitions of GCFG in the literature, and we introduce yet another one, in which a Generalized Context-Free Grammar consists of an abstract grammar together with a concrete interpretation.

**Abstract grammar** The abstract grammar is a tuple  $(C, S, F, R)$ , where  $C$  and  $F$  are finite sets of categories and function symbols respectively,  $S \in C$  is the starting category, and  $R \subseteq C \times F \times C^*$  is a finite set of abstract syntax rules. For each function symbol  $f \in F$  there is an associated context-free syntax rule.

$$A \rightarrow f(A_1, \dots, A_\delta)$$

The tree rewriting relation  $A \Rightarrow t$  is defined as  $A \Rightarrow f(t_1, \dots, t_\delta)$  whenever  $A_1 \Rightarrow t_1, \dots, A_\delta \Rightarrow t_\delta$ .

**Concrete interpretation** To each category  $A$  is associated a *linearization type*  $\llbracket A \rrbracket$ , which is not further specified. To each function symbol  $f$  is associated a partial *linearization function* (also written  $f$ ), taking as many arguments as the abstract syntax rule specifies.

$$f \in \llbracket A_1 \rrbracket \rightarrow \dots \rightarrow \llbracket A_\delta \rrbracket \rightarrow \llbracket A \rrbracket$$

The linearization of a syntax tree is defined as  $\llbracket f(t_1, \dots, t_\delta) \rrbracket = f(\llbracket t_1 \rrbracket, \dots, \llbracket t_\delta \rrbracket)$ , if the application is defined. Note that we impose no restrictions on the linearization types or the linearization functions; this is left to the actual grammar formalism.

### 6.2.1 GF with a context-free backbone

Grammatical Framework with a context-free backbone is obviously an instance of GCFG, where the abstract GF rule  $f : A_1 \rightarrow \dots \rightarrow A_\delta \rightarrow A$  is just another way of writing the abstract GCFG rule  $A \rightarrow f(A_1, \dots, A_\delta)$ .

### 6.2.2 Multiple Context-Free Grammars

Multiple Context-Free Grammars (Seki et al., 1991) were introduced in the late 80's as a very expressive formalism, incorporating Linear Context-Free Rewriting Systems and other mildly context-sensitive formalisms, but still with a polynomial parsing algorithm. MCFG is an instance of GCFG, with the following restrictions on linearizations:

- Linearization types are restricted to tuples of strings.
- The only allowed operations in linearization functions are tuple projections and string concatenations.

Since records can be seen as syntactic sugar for tuples, we can use records in this article without changing the definition of MCFG.

**Comparison with GF** Obviously MCFG is an instance of context-free GF, but without tables and table selections. The fact that GF can have nested records does not change anything – all nestings can be flattened. Also, an expanded table

$$[\mathbf{p}_1 \Rightarrow \phi_1; \dots; \mathbf{p}_n \Rightarrow \phi_n] : \mathbf{P} \Rightarrow T$$

is equivalent to a record

$$\{\mathbf{p}_1 = \phi_1; \dots; \mathbf{p}_n = \phi_n\} : \{\mathbf{p}_1 : T; \dots; \mathbf{p}_n : T\}$$

and an instantiated selection  $\phi! \mathbf{p}_i$  is equivalent to a projection  $\phi.\mathbf{p}_i$ .

There are two fundamental differences:

- MCFG cannot have parameters as linearization values.
- A table selection in GF does not necessarily have to be an instantiated parameter; it can be any term of the correct linearization type.

In the example grammar, the linearization of  $p$  contains a selection  $y.s!(x.n)$ , which is not instantiated.

### 6.3 Converting GF to MCFG

In this section we show that it is possible to convert a GF grammar into an equivalent grammar where all table selections are instantiated, and containing no parameters. By the argument above, this means that context-free GF and MCFG are equivalent. This conversion is done in two steps, described later in sections 6.3.2 and 6.3.3, and the following theorem is a consequence.

**Theorem 23** *Any GF grammar with a context-free backbone can be reduced to an equivalent MCFG grammar.*

#### 6.3.1 Preliminaries

A linearization term  $\phi$  is in  *$\eta$ -normal form* if the structure follows the structure of its linearization type; i.e.  $\phi$  is a record if the type is a record type, and  $\phi$  is an expanded table if the type is a table type. The subterms of  $\phi$  which are of the basic linearization types, **Str** or **P**  $\in$  **Par**, are called the *leaves* of  $\phi$ . A *path* is a sequence of record projections and table selections; meaning that  $\epsilon$ ,  $\sigma.r$  and  $\sigma!\phi$  are paths if  $\sigma$  is a path. A path that does not contain any argument variables  $x_i$  is called *instantiated*; in which case the selections  $\phi$  can only be parameters. A non-instantiated path is called *nested*; this is because if a path contains an argument variable  $x_i$ , then that variable is always followed by a (possibly empty) path.

A linearization type  $T$  as well as a linearization  $\phi$  can be partitioned into parameter paths and string paths:

$$\begin{aligned} T^{\text{Str}} &= \{ \sigma : \text{Str} \mid T.\sigma = \text{Str} \} \\ T^{\text{Par}} &= \{ \sigma : \text{P} \mid T.\sigma = \text{P} \in \text{Par} \} \\ \phi^{\text{Str}} &= \{ \sigma = \phi.\sigma \mid \phi.\sigma : \text{Str} \} \\ \phi^{\text{Par}} &= \{ \sigma = \phi.\sigma \mid \phi.\sigma : \text{P} \in \text{Par} \} \end{aligned}$$

Note that we equate nestings of tables/records and sets of path-value pairs, and that we extend paths to linearization types in the obvious way. Also note that there are only a finite number of instantiated parameter records  $\pi : T^{\text{Par}}$ , since there are only finitely many parameters.

**The example grammar** For the term  $a : \llbracket N \rrbracket$  in the example grammar we have that

$$\begin{aligned} \llbracket N \rrbracket^{\text{Str}} &= \{ s : \text{Str} \} \\ \llbracket N \rrbracket^{\text{Par}} &= \{ n : \text{Num} \} \\ a^{\text{Str}} &= \{ s = \text{“animals”} \} \\ a^{\text{Par}} &= \{ n = \text{PI} \} \end{aligned}$$

### 6.3.2 A normal form for linearizations

**Definition 1** A GF linearization is in table normal form if it is of the form

$$f(x_1, \dots, x_\delta) = [\pi_1 \Rightarrow \phi_1; \dots; \pi_n \Rightarrow \phi_n]! \xi$$

and the following hold:

- $\xi$  contains all parameter paths of the arguments  $x_1, \dots, x_\delta$ ; in other words  $\xi = (x_1^{\text{Par}}, \dots, x_\delta^{\text{Par}})$ .
- Each  $\pi_k$  is a possible parameter instantiation of  $\xi$ ; in other words  $\pi_k : \llbracket A_1 \rrbracket^{\text{Par}} \times \dots \times \llbracket A_\delta \rrbracket^{\text{Par}}$ .
- Each  $\phi_k$  is in  $\eta$ -normal form where the leaves are either parameters or concatenations of constant strings and instantiated string paths.

The following algorithm converts any GF linearization in canonical form into normal form.

**Algorithm 1** First, add the outer table as in the definition of table normal form:

$$\begin{aligned} f(x_1, \dots, x_\delta) &= [\pi_1 \Rightarrow \phi; \dots; \pi_n \Rightarrow \phi]! \xi \\ \xi &= (x_1^{\text{Par}}, \dots, x_\delta^{\text{Par}}) \\ \pi_k &: \llbracket A_1 \rrbracket^{\text{Par}} \times \dots \times \llbracket A_\delta \rrbracket^{\text{Par}} \end{aligned}$$

Second, for each instantiation  $\pi_k$ , convert  $\phi$  to  $\phi_k$ , by repeating the following substitution until there are no parameter paths left:

- Substitute each instantiated parameter path  $x_i.\sigma$  for its  $\pi_k$ -instantiation  $(\pi_k)_i.\sigma$ .

**Lemma 24** *The algorithm, together with the standard computation rules, yields an equivalent linearization in table normal form.*

**The example grammar** There are two linearizations in the example that are not in table normal form, and this is how they look after conversion:

$$\begin{aligned}
 p(x, y) &= [\text{Sg} \Rightarrow \{s = x.s \cdot y.s ! \text{Sg}\}; \\
 &\quad \text{Pl} \Rightarrow \{s = x.s \cdot y.s ! \text{Pl}\}] ! x.n \\
 q(x, y) &= [\text{Sg} \Rightarrow \{s = [\text{Sg} \Rightarrow x.s ! \text{Sg} \cdot y.s; \\
 &\quad \text{Pl} \Rightarrow x.s ! \text{Pl} \cdot y.s]\}; \\
 &\quad \text{Pl} \Rightarrow \{s = [\text{Sg} \Rightarrow x.s ! \text{Sg} \cdot y.s; \\
 &\quad \text{Pl} \Rightarrow x.s ! \text{Pl} \cdot y.s]\}] ! y.n
 \end{aligned}$$

### 6.3.3 Refining the abstract syntax

To get an MCF grammar, we have to get rid of the parameters in some way; and this we do by moving them to the abstract syntax. Each table row  $\pi_k \Rightarrow \phi_k$  above will then give rise to a unique function symbol with linearization  $\phi_k$ .

**Algorithm 2** *Given a GF grammar where all linearizations are in table normal form, create a grammar with the following categories, function symbols and linearizations:*

- For each category  $A$  and each instantiated parameter record  $\pi : \llbracket A \rrbracket^{\text{Par}}$ , create a new category  $\hat{A} = A[\pi]$ . The linearization type is the same as the string paths of the original linearization type,  $\llbracket \hat{A} \rrbracket = \llbracket A \rrbracket^{\text{Str}}$
- For each syntax rule  $A \rightarrow f(A_1, \dots, A_\delta)$ , and all new categories  $\hat{A}$ ,  $\hat{A}_1, \dots, \hat{A}_\delta$ , create a new syntax rule  $\hat{A} \rightarrow \hat{f}(\hat{A}_1, \dots, \hat{A}_\delta)$ ; where  $\hat{f}$  is a unique function symbol,  $\hat{f} = f[\hat{A} \rightarrow \hat{A}_1 \dots \hat{A}_\delta]$ .
- For each linearization function

$$f(x_1, \dots, x_\delta) = [\pi_1 \Rightarrow \phi_1; \dots; \pi_n \Rightarrow \phi_n] ! \xi$$

and each table row  $\pi_k \Rightarrow \phi_k$ , create a new linearization function for  $\hat{f} = f[\hat{A} \rightarrow \hat{A}_1 \dots \hat{A}_\delta]$ :

$$\begin{aligned}
 \hat{f}(x_1, \dots, x_\delta) &= \phi_k^{\text{Str}} \\
 \hat{A} &= A[\phi_k^{\text{Par}}] \\
 \hat{A}_i &= A_i[(\pi_k)_i] \quad (1 \leq i \leq \delta)
 \end{aligned}$$

where we by  $(\pi_k)_i$  mean the  $i$ th component of  $\pi_k$ .

$$\begin{aligned}
[[\hat{N}_1]] &= [[\hat{N}_2]] = [[\hat{S}]] = \{s : \text{Str}\} \\
[[\hat{V}]] &= \{s! \text{Sg} : \text{Str}; s! \text{Pl} : \text{Str}\} \\
\hat{p}_1 &: \hat{N}_1 \rightarrow \hat{V} \rightarrow \hat{S} \\
\hat{p}_1(x, y) &= \{s = x.s \cdot y.s! \text{Sg}\} \\
\hat{p}_2 &: \hat{N}_2 \rightarrow \hat{V} \rightarrow \hat{S} \\
\hat{p}_2(x, y) &= \{s = x.s \cdot y.s! \text{Pl}\} \\
\hat{q}_1 &: \hat{V} \rightarrow \hat{N}_1 \rightarrow \hat{V} \\
\hat{q}_1(x, y) &= \{s! \text{Sg} = x.s! \text{Sg} \cdot y.s; \\
&\quad s! \text{Pl} = x.s! \text{Pl} \cdot y.s\} \\
\hat{q}_2 &: \hat{V} \rightarrow \hat{N}_2 \rightarrow \hat{V} \\
\hat{q}_2(x, y) &= \hat{q}_1(x, y) \\
\hat{a} : \hat{N}_2 &= \{s = \text{“animals”}\} \\
\hat{b} : \hat{N}_1 &= \{s = \text{“Bernie”}\} \\
\hat{l} : \hat{V} &= \{s! \text{Sg} = \text{“loves”}; s! \text{Pl} = \text{“love”}\}
\end{aligned}$$

FIGURE 2 Grammar after conversion to MCFG

Obviously the resulting grammar is an MCF grammar, since all linearizations are records of strings.

**Lemma 25** *The resulting grammar is equivalent to the original.*

**The example grammar** Figure 2 shows how the example grammar looks like after conversion to MCFG.

### 6.3.4 Non-deterministic reduction

There is a more direct conversion, using a non-deterministic substitution algorithm. This can also reduce the size of the resulting grammar, when argument parameters are not mentioned in linearizations.

**Algorithm 3** *Assume the following abstract syntax rule, together with its linearization function:*

$$\begin{aligned}
A &\rightarrow f(A_1, \dots, A_\delta) \\
f(x_1, \dots, x_\delta) &= \phi
\end{aligned}$$

*Repeat the following non-deterministic substitution until there are no instantiated parameter paths left, accumulating the parameter records  $\pi_1, \dots, \pi_\delta$ :*

- *Substitute each instantiated parameter path  $x_i.\sigma : P$  with any  $p \in P$ , such that adding the row  $\sigma = p$  to  $\pi_i$  is consistent.*

Supposing that the final substituted linearization is  $\psi$ , we can add the following rule for the new function symbol  $\hat{f}$ :

$$\begin{aligned} \hat{A} &\rightarrow \hat{f}(\hat{A}_1, \dots, \hat{A}_\delta) \\ \hat{f}(x_1, \dots, x_\delta) &= \psi^{\text{Str}} \\ \hat{A} &= A[\psi^{\text{Par}}] \\ \hat{A}_i &= A_i[\pi_i] \quad (1 \leq i \leq \delta) \end{aligned}$$

The algorithm is non-deterministic, and we get the final grammar by finding all solutions for each function symbol  $f$ ; which can be done by a standard all-solutions predicate, such as `findall` in Prolog.

**Coercions between categories** There is a difference between algorithm 3 and the previous algorithms; if an argument parameter  $x_i.\sigma$  is not mentioned in  $\phi$ , then there will be no  $\sigma$ -row in the constraint record  $\pi_i$ . This means that the new category  $\hat{A}_i = A_i[\pi_i]$  will only contain a subrecord of  $A_i[\phi_i^{\text{Par}}]$ , where  $\phi_i$  is a linearization of type  $\llbracket A_i \rrbracket$ .

**Algorithm 4** *Given two reduced syntax rules,*

$$\begin{aligned} \hat{A} &\rightarrow \hat{f}(\dots \hat{B}_1 \dots) \\ \hat{B}_2 &\rightarrow \hat{g}(\dots) \end{aligned}$$

where  $\hat{B}_1 = B[\pi_1]$  and  $\hat{B}_2 = B[\pi_2]$ . If  $\pi_1$  is a subrecord of  $\pi_2$ , add the coercion function  $\hat{c} = c[\pi_1 \pi_2]$ :

$$\begin{aligned} \hat{B}_1 &\rightarrow \hat{c}(\hat{B}_2) \\ \hat{c}(x) &= x \end{aligned}$$

**The example grammar** One function symbol gets a different linearization from algorithm 3 than in figure 2; the functions  $\hat{q}_1$  and  $\hat{q}_2$  get merged into one function  $\hat{q}$ .

$$\begin{aligned} \hat{q} &: \hat{V} \rightarrow \hat{N} \rightarrow \hat{V} \\ \hat{q}(x, y) &= \{ s! \text{Sg} = x.s! \text{Sg} \cdot y.s; \\ &\quad s! \text{Pl} = x.s! \text{Pl} \cdot y.s \} \end{aligned}$$

where  $\hat{N} = N[]$ . This yields coercions for the more specific types  $\hat{N}_1 = N[n = \text{Sg}]$  and  $\hat{N}_2 = N[n = \text{Pl}]$ :

$$\begin{aligned} \hat{N} &\rightarrow \hat{c}_i(\hat{N}_i) \quad (i = 1, 2) \\ \hat{c}_i(x) &= x \end{aligned}$$

## 6.4 Implications to Parsing

**Definition 2** *A chart for a GCFG is a finite set of tuples  $(f, \phi, \phi_1, \dots, \phi_\delta)$ , where  $\phi = f(\phi_1, \dots, \phi_\delta)$ .*

A tree  $t = f(t_1, \dots, t_\delta)$  is represented by the chart if it contains  $(f, \llbracket t \rrbracket, \llbracket t_1 \rrbracket, \dots, \llbracket t_\delta \rrbracket)$ , and each subtree  $t_i$  is represented by the chart.

The following lemma is just another way of saying that GCFG grammars are compositional:

**Lemma 26** *The set of GCFG trees  $\{t \mid \llbracket t \rrbracket = \phi\}$ , for a given  $\phi$ , can be represented by a single chart.*

In other words, a correct parsing algorithm for GCFGs does not have to return anything more than a chart.

If we have translated a context-free GF grammar into MCFG using algorithm 1+2, it is straight-forward to translate back a chart for the MCFG into a chart for the original grammar. Each item  $(\hat{f}, \phi, \phi_1, \dots, \phi_\delta)$ , where  $\hat{f} = f[\hat{A} \rightarrow \hat{A}_1 \dots \hat{A}_\delta]$  and  $\hat{A}_i = A_i[\pi_i]$ , can be converted to the item  $(f, \phi \cup \pi, \phi_1 \cup \pi_1, \dots, \phi_\delta \cup \pi_\delta)$ . Back-translation of trees are even simpler; just strip off the extra information from the nodes – each tree node  $\hat{f} = f[\dots]$  is converted to  $f$ .

If we have converted using algorithm 3+4, back-translation is only slightly more complicated; if there is a coercion  $\hat{A}_k \rightarrow \hat{c}(\hat{A}_k)$ , where  $\hat{A}'_k = A_k[\pi'_k]$ , use the linearization  $\phi_k \cup \pi'_k$  instead of  $\phi_k \cup \pi_k$ .

#### 6.4.1 Dependent categories

If we have a GF grammar with dependent categories, there is a straight-forward two-step parsing process for that grammar. First we simply remove all dependencies from the abstract syntax, thereby getting a grammar with a context-free backbone. This grammar is over-generating, so when parsing we get a chart containing all parse trees we want, but perhaps also some unwanted trees.

The second step is to convert the chart into Horn clauses, which can be solved by any proof search, e.g. standard Prolog. This conversion is done one item at the time; suppose the following chart item:

$$(f, \phi, \phi_1, \dots, \phi_\delta)$$

where  $f$  has the following abstract typing:

$$\begin{aligned} f & : (x_1 : A_1) \rightarrow \dots \rightarrow (x_\delta : A_\delta(x_1, \dots, x_{\delta-1})) \\ & \rightarrow A(x_1, \dots, x_\delta) \end{aligned}$$

From this we can create the following Horn clause (where  $t : A[\phi]$  is just syntactic sugar for a 3-tuple):

$$\begin{aligned} f(x_1, \dots, x_\delta) : A(x_1, \dots, x_\delta)[\phi] & :- \\ x_1 : A_1[\phi_1], \dots, x_\delta : A_\delta(x_1, \dots, x_{\delta-1})[\phi_\delta] & \end{aligned}$$

Finally, the query  $:- x : S[\phi]$ , where  $x$  is a logic variable, will result

in all possible parse trees  $x$  of category  $S$ , linearizing to the input string  $\phi$ .

### 6.4.2 Functional categories

In full GF, arguments to functions can themselves be functions. This gives rise to the question of how to linearize an “incomplete” category  $B_1 \rightarrow \dots \rightarrow B_\delta \rightarrow B$ . This is solved in GF by pairing the linearization of the result category  $B$  with linearizations of the *variable bindings* representing objects of category  $B_1, \dots, B_\delta$ .

Formally, each occurrence of a function category  $B_1 \rightarrow \dots \rightarrow B_\delta \rightarrow B$  as an argument in a typing is replaced by the new category  $\hat{B}$ , with linearization type

$$[[\hat{B}]] = [[B]] \times [[\mathbf{Var}]]^\delta$$

where  $\mathbf{Var}$  is a unique category for recognizing variable bindings, specified by the grammar. In GF, the default linearization type of variables is  $[[\mathbf{Var}]] = \mathbf{Str}$ , and it recognizes strings looking like ordinary mathematical variables (“ $x$ ”, “ $y$ ”, “ $z$ ”, ...).

For each new category  $\hat{B}$  we also need a coercion  $\hat{b}$ :

$$\begin{aligned} \hat{b} &: B \rightarrow \mathbf{Var} \rightarrow \dots \rightarrow \mathbf{Var} \rightarrow \hat{B} \\ \hat{b}(x, y_1, \dots, y_\delta) &= (x, y_1, \dots, y_\delta) \end{aligned}$$

This conversion shows that adding function arguments to abstract typings does not change the expressive power of GF, and that they are possible to handle with the parsing algorithms described in this paper.

## References

- Dymetman, Marc, Veronica Lux, and Aarne Ranta. 2000. XML and multilingual document authoring: Convergent trends. In *COLING, Saarbrücken, Germany*, pages 243–249.
- Hähne, Reiner, Kristofer Johannisson, and Aarne Ranta. 2002. An authoring tool for informal and formal requirements specifications. In R.-D. Kutsche and H. Weber, eds., *Fundamental Approaches to Software Engineering*, vol. 2306 of *LNCS*, pages 233–248. Springer.
- Hallgren, Thomas and Aarne Ranta. 2000. An extensible proof text editor. In M. Parigot and A. Voronkov, eds., *LPAR-2000*, vol. 1955 of *LNCS/LNAI*, pages 70–84. Springer.
- Pollard, Carl. 1984. *Generalised Phrase Structure Grammars, Head Grammars and Natural Language*. Ph.D. thesis, Stanford University.
- Ranta, Aarne. 2004. Grammatical Framework, a type-theoretical grammar formalism. *Journal of Functional Programming* 14(2):145–189.
- Ranta, Aarne and Robin Cooper. 2004. Dialogue systems as proof editors. *Journal of Logic, Language and Information* To appear.

- Seki, Hiroyuki, Takashi Matsumara, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88:191–229.
- Vijay-Shanker, K. David Weir, and Aravind Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *25th meeting of Association for Computational Linguistics*.



# Elliptical Constructions, Multiple Frontings, and Surface-Based Syntax

STEFAN MÜLLER

Kathol (1995, 1997, 2000, 2001) developed a theory of German clause types that is based on the Topological Fields model known from descriptive linguistics (Drach, 1937, Reis, 1980, Höhle, 1986, Askedal, 1986). He suggests relating the clause type of sentences to serialization patterns of overtly realized material. In (Kathol, 1997), he used learnability arguments to argue for a non-abstract syntax, i.e. a syntax where surface order plays a crucial role and the reference to abstract syntactic objects such as functional heads is avoided in favour of observationally accessible properties (p. 89).

In this paper, I show that an entirely surface-based conception of syntax is not tenable and that Kathol's proposal faces problems with certain elliptical constructions.

In the first section, I very briefly repeat his key assumptions. In Section 7.2, I will discuss problematic aspects of the proposal like verbless clauses, and declarative sentences that do not fit the pattern suggested by Kathol. I then suggest an analysis that does not rely on the surface order of constituents for the classification of clause types, but on the relations expressed by immediate dominance schemata.

*Proceedings of Formal Grammar 2004.*

Gerhard Jäger, Paola Monachesi, Gerald Penn, and Shuly Winter (eds.).

Copyright © 2004, the individual authors.

## 7.1 Constructional Constraints and Topological Fields

The examples in (9) show various linearization patterns that are attested in German clauses:

- (1) a. daß Lisa eine Blume gepflanzt hat  
       that Lisa a flower planted has  
       ‘that Lisa planted a flower.’  
 b. was Lisa gepflanzt hat  
       what Lisa planted has  
 c. Hat Lisa eine Blume gepflanzt?  
       has Lisa a flower planted  
       ‘Did Lisa plant a flower?’  
 d. Eine Blume hat Lisa gepflanzt.  
       a flower has Lisa planted  
       ‘Lisa planted a flower.’

(8a) is an example for sentences that are introduced by a complementizer and (8b) is an example for embedded interrogative sentences. Both sentences are verb-final. (8c–d) are verb-initial sentences: (8c) is a yes/no question and (8d) is a declarative sentence. Declarative sentences usually differ from yes/no questions in that one constituent fills the position before the finite verb.

### 7.1.1 Topological Fields, Linearization Rules, and Uniqueness Constraints

Kathol (2001, p. 50) gives the following division into topological fields for the sentences in (9):

	Vorfeld ‘initial field’ 1	linke Satzklammer ‘left bracket’ 2	Mittelfeld ‘middle field’ 3	rechte Satzklammer ‘right bracket’ 4
Vfinal		daß	Lisa eine Blume	gepflanzt hat
		was	Lisa	gepflanzt hat
V1		hat	Lisa eine Blume	gepflanzt
V2	eine Blume	hat	Lisa	gepflanzt

This is the classical terminology with additional labels 1–4 to refer to the respective positions.

He then formulates the following linearization constraints:

- (2) Topological Linear Precedence Constraint  
 $1 < 2 < 3 < 4$

- (3) Topological Uniqueness Conditions
- a.  $1 < 1$
  - b.  $2 < 2$

The first constraint ensures that all elements that are assigned to the field 1 are serialized before 2 and so on. The second is a trick from the GPSG literature (Gazdar, Klein, Pullum, and Sag, 1985, p. 55) to rule out multiple occurrences of elements assigned to the fields 1 or 2. Since constraint (8a) requires that all elements with the field 1 have to precede the other elements assigned to field 1 this constraint is necessarily violated if there is more than one element assigned to 1.

### 7.1.2 A Hierarchy of Clause Types

Kathol follows Reape (1996, 1992, 1994), who introduced linearization domains into the HPSG framework. Daughters which are combined by the usual dominance schemata may be non-adjacent. The daughters are inserted into a domain list named DOM. The elements of this list may be permuted in any order provided no LP constraint is violated. The order of the elements corresponds to the surface order. This makes it possible to assign both sentences in (9) the dominance structure in (10).

- (4) a. der Mann das Buch liest  
 b. Liest der Mann das Buch?

- (5)  $[_V \text{ der Mann } [_V \text{ das Buch liest}]]$

The sentences differ only in the order of the elements in their linearization domain. In the analysis of (7a) the verb is serialized finally and in the analysis of (7b) it is serialized initially.

Kathol (2001) defines clause types with reference to elements in the constituent order domains. He assumes that all clauses are subtypes of the following three types:

- (6) a. *V1-clause*  $\rightarrow$  
$$\left[ \begin{array}{l} S[fin] \\ \text{DOM} \quad \left\langle \left[ \begin{array}{c} 2 \\ V[fin] \end{array} \right], \dots \right\rangle \end{array} \right]$$
- b. *V2-clause*  $\rightarrow$  
$$\left[ \begin{array}{l} S[fin] \\ \text{DOM} \quad \left\langle [ 1 ], \left[ \begin{array}{c} 2 \\ V[fin] \end{array} \right], \dots \right\rangle \end{array} \right]$$

$$c. \text{ subord-clause} \rightarrow \left[ \begin{array}{l} S[fin] \\ \text{DOM} \left\langle \dots, \left[ \begin{array}{l} 2 \\ \text{HEAD} \neg V[fin] \end{array} \right], \dots \right\rangle \end{array} \right]$$

These types impose restrictions on possible orderings of elements in constituent order domains or stipulate that finite verbs may not appear in the field 2 in subordinated clauses. The first type states that a verb first clause has a finite verb as the first element in its domain list and the second states that there is an element in 1 (the *Vorfeld*) before the finite verb in 2.

Kathol cross-classifies the types in (8) with the types *declarative*, *wh-interrogative*, and *polar*. He provides the hierarchy shown in Figure 1.

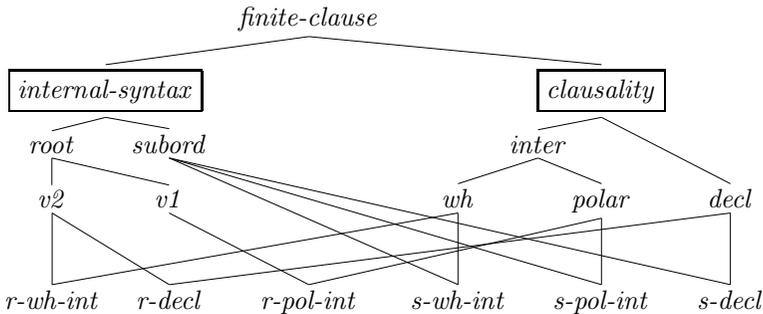


FIGURE 1 Clausal Types

Such a linearization-based approach to clause type determination would be very attractive if there were a one-by-one mapping from the surface order of constituents to clause types, but as I will show in Section 7.2, this is not the case.

### 7.1.3 Competition of Complementizer and Finite Verb

Kathol follows ideas by Thiersch (1978) and den Besten (1983) and assumes that the complementizer and the finite verb compete for the position in the left sentence bracket. If no complementizer is present, the verb may move into the left sentence bracket. If the left sentence bracket is occupied, it has to stay in the right sentence bracket.

Kathol enters verbs into the lexicon with a specification of the potential topological fields they may appear in. He specifies finite verbs for the fields 2 or 4. Complementizers are always located in the left sentence bracket: They are specified for 2. If a linearization domain contains a complementizer, the Topological Uniqueness Condition b

ensures that no other element can be serialized in 2, hence the field 4 is the only option for the finite verb.

## 7.2 Problematic Aspects of this Approach

In the following section I want to discuss four problematic aspects of this proposal.

### 7.2.1 Verbless Clauses

There are main clauses in German that consist of a predicate and a clause that depends on this predicate, but no verb (see also (Paul, 1919, p. 41) for more examples).

- (7) a. Doch egal, was noch passiert, der Norddeutsche  
but never.mind what still happens the North.German  
Rundfunk steht schon jetzt als Gewinner  
broadcasting.company stands already now as winner  
fest.<sup>1</sup>  
PART  
‘But never mind what happens, it is already certain that  
the Norddeutscher Rundfunk (North German broadcasting  
company) will be the winner.’
- b. Interessant, zu erwähnen, daß ihre Seele völlig in  
interesting to mention that her soul completely in  
Ordnung war.<sup>2</sup>  
order was  
‘It is interesting to point out that she was completely sane.’
- c. Ein Treppenwitz der Musikgeschichte, daß die  
a stair.joke of.the music.history that the  
Kollegen von Rammstein vor fünf Jahren noch im  
colleagues of Rammstein before five years still in.the  
Vorprogramm von Sandow spielten.<sup>3</sup>  
before.program of Sandow played  
‘It is an irony of musical history that the colleagues from (the  
band) Rammstein were still playing as the support group of  
Sandow a few years ago.’

In the sentences in (8) the copula *sein* (‘be’) has been omitted. The sentences in (8) correspond to the sentences in (9).

<sup>1</sup>Spiegel, 12/1999, p. 258

<sup>2</sup>Michail Bulgakow, *Der Meister und Margarita*. München: Deutscher Taschenbuch Verlag. 1997, p. 422

<sup>3</sup>Flüstern & Schweigen, taz, 12.07.1999, p. 14

- (8) a. Doch was noch passiert, ist, egal, ...  
 but what still happens is never.mind
- b. Zu erwähnen, daß ihre Seele völlig in Ordnung war,  
 to mention that her soul completely in order was  
 ist interessant.  
 is interesting
- c. Daß die Kollegen von Rammstein vor fünf Jahren  
 that the colleagues of Rammstein before five years  
 noch im Vorprogramm von Sandow spielten ist ein  
 still in.the before.program of Sandow played is a  
 Treppenwitz der Musikgeschichte.  
 stair.joke of.the music.history

The copula as used with adjectives does not contribute semantically, it merely provides agreement information and the verbal features that may be needed by other predicates that embed the copula construction (Paul, 1919, p. 41). As the examples in (7) show, the copula may be omitted. The result are clauses without a finite verb.

The examples in (7) are declarative sentences, i.e. they should have the pattern in (6b). (9) is an example for a question. The sentence corresponds to a verb first sentence with the copula in initial position, i.e. it should correspond to the pattern in (6a).

- (9) Niemand da?<sup>4</sup>  
 nobody there  
 ‘Is anybody there?’

In order to save the clause type determination one could stipulate a phonologically empty verb.<sup>5</sup> However, Kathol (1995, Chapter 5.4.1) explicitly rules out the option of domain elements with empty phonology values. In (Kathol, 2001, p. 38) he argues against empty elements so that for him the necessity to stipulate an empty element seems to be an unwanted consequence of his proposal. In any case, empty elements are highly abstract entities which have no place in his conception of non-abstract syntax.

### 7.2.2 Topic Drop

While the cases of copula ellipsis can be found in novels, news papers, magazines, and everyday speech, a construction, which is called *Vorfeldellipse* or Topic Drop or Pronoun Zap is more restricted to a certain

<sup>4</sup>Paul (1919, p. 13)

<sup>5</sup>See Sag, Wasow, and Bender, 2003, p. 464 for the suggestion of an empty verbal copula for African American Vernacular English.

register/style. Huang (1984), Fries (1988), and Hoffmann (1997) discuss this construction in some detail. Topic drop is also problematic for Kathol's approach: Sentences with Topic Drop look like polar questions at the surface. If an obligatory complement is dropped, the sentence is distinguishable from questions since the complement is missing in the *Mittelfeld* (9a). If optional complements or adjuncts are dropped, the form of the sentence is absolutely identical to the form of yes/no questions (9b).

- (10) a. Hab' ich auch gekannt.  
           have I   also known  
           'I also knew him/her/it.'
- b. Hab' ich auch gegessen.  
           have I   also eaten  
           'I also eat him/her/it.' or (with different intonation) 'Did I  
           also eat?'

Such topic drop utterances and polar questions differ only in intonation and not in the sequence of elements.

In order to save the clause type determination one could assume a phonologically empty element in the *Vorfeld*. As was discussed in Section 7.2.1, Kathol explicitly rejects empty elements.

Alternatively one could stipulate just one more type that constrains the domain list to contain a slashed verb, as was suggested by a reviewer of HPSG 2002. While this is technically possible, the commonalities of sentences with a filled *Vorfeld* and those that are the result of Topic Drop would not be captured.

### 7.2.3 Sentential Complements

Kathol (2000, p. 152) assumes that in (9) the V2 clauses in brackets are complement clauses:

- (11) a. Otto glaubt [die Erde sei flach].  
           Otto believes the earth is flat  
           'Otto believes that the earth is flat.'
- b. die Überzeugung / der Glaube / ... [die Russen würden  
           the conviction       the belief               the Russians would  
           nicht in Polen eingreifen]  
           not in Poland intervene  
           'the conviction/belief/...that the Russians would not inter-  
           vene in Poland.'

On page 153 he formulates a Head-V2-Complement Schema that combines a head that takes a finite unmarked clause as complement with that complement. The schema restricts the clause type of the complement to be *root-decl*, i.e., a sentence with the verb in second position. Kathol's clausal types are subtypes of the type *sign*. They refer to the domain values of a sign which are represented at the outermost level of a feature structure and therefore the clause types could not be subtypes of *synsem* or other types inside of the feature structures contained under SYNSEM and hence the clause type of complements cannot be selected by governing heads. Therefore Kathol is forced to encode this combinatorial property in the immediate dominance schemata. In order to avoid spurious ambiguities Kathol has to restrict the general head argument schema so that it does not apply when the Head-V2-Complement Schema applies.

A grammar that uses sufficient subcategorization information and one head argument schema instead of stipulating several special schemata is more general than what is suggested by Kathol and should therefore be regarded the better alternative.

#### 7.2.4 Multiple Constituents in the *Vorfeld*

As far as learnability and non-abstractness are concerned the following data pose a problem for Kathol:<sup>6</sup>

- (12) a. [Nichts] [mit derartigen Entstehungstheorien] hat es  
 nothing with those.kinds.of creation.theories has it  
 natürlich zu tun, wenn ...<sup>7</sup>  
 of.course to do when  
 'Of course it has nothing to do with that kind of creation  
 theory when ...'
- b. [Trocken] [durch die Stadt] kommt man am  
 dry through the town comes one at.the  
 Wochenende auch mit der BVG.<sup>8</sup>  
 weekend also with the BVG  
 'The BVG (Berlin public transport system) will also get you  
 about town on the weekend without getting wet.'

<sup>6</sup>(9b-c) are quoted from (Müller, 2002b).

<sup>7</sup>K. Fleischmann, *Verbstellung und Relieftheorie*, München, 1973, p. 72. quoted from (van de Velde, 1978, p. 135).

<sup>8</sup>taz berlin, 10.07.1998, p. 22

- c. [Alle Träume] [gleichzeitig] lassen sich nur selten  
 all dreams simultaneously let self only rarely  
 verwirklichen.<sup>9</sup>  
 realize  
 ‘All dreams can seldom be realized at once.’

These examples seem to violate the V2 constraint. In a purely surface-based model without any abstract entities, there is no way to explain sentences like (8). One could stipulate constructions that combine the elements before the finite verb so that they form a constituent and the V2 constraint is saved. However, the data discussed in Müller, 2003 shows that various combinations of material in the *Vorfeld* are possible. For instance, we have a depictive secondary predicate and a directional PP argument in (12b) and an argument and an adverbial in (12c). This means that the stipulation of several constructions would be necessary in order to provide the correct meaning for the combination of material in front of the finite verb.

If one uses one abstract entity, an empty verbal head as suggested by Müller (2002b), a stipulation of several constructions would be unnecessary. The empty verbal head is related to a verb in the remaining clause by a non-local dependency, which constraints the elements that can appear together in the *Vorfeld* and makes possible a compositional assignment of meaning to the sentence. Müller (2002b) uses a linearization-based model of the Reape/Kathol style to account for verb-initial and verb-final sentences. In such a model the use of an empty head is a stipulation. If one returns to a verb movement analysis as suggested for instance by Kiss and Wesche, 1991, Kiss, 1995 the empty head that is used for verb movement in general can also be used for the multiple fronting cases in (8). The details of the multiple fronting analysis for (8) together with a verb movement analysis can be found in Müller, In Press.

### 7.3 An Alternative Proposal

In the discussion above, I already hinted at possible solutions to the problems. For copula less sentences I will assume an empty copula, for sentential complements of nominal heads, I assume the standard selectional mechanisms and a normal combination of head and argument via the head-argument-schema. Since the information that is relevant as far as clause types are concerned is represented under SYNSEM, it can be selected and no additional ID schemata are necessary. For the

---

<sup>9</sup>Brochure from Berliner Sparkasse, 1/1999

linearization of the finite verb, I assume a verb movement analysis (Müller, In Press) and the empty head that is used in this analysis can also account for the multiple frontings as explained by Müller (2002b, In Press). What is still missing is an explanation of the distribution of complementizer and verb, the analysis of topic drop, and the clause type determination. These issues are dealt with in the following subsections.

### 7.3.1 Complementizer and Finite Verb

To account for the distribution of complementizer and finite verb, I suggest returning to the old analysis where verbs have a binary feature INV that marks whether the verb is serialized head-finally (INV−) or head-initially (INV+) (Uszkoreit, 1987, Pollard, 1996). Following Pollard (1996, p. 292), I assume INV to be a head feature. I assume that the complementizer selects for a sentence with the verb in final position, i.e., for a maximal projection of an INV− verb:

- (13) daß [der Mann den Roman schreibt].  
           that the man the novel writes

See Kiss, 1995, S. 55–57 for an argumentation for the head status of complementizers in German.

### 7.3.2 Topic Drop

The sentences from Huang (1984) in (9) show that both subjects and objects can be dropped.

- (14) a. [Ihn] hab' ich schon gekannt.  
           him have I yet known  
           'I knew him.'  
       b. [Ich] hab' ihn schon gekannt.  
           I have him yet known

The material in brackets may be omitted.

(9) shows that adjuncts can also be omitted:

- (15) Die (die Pinguine) kommen so nah ran, daß man sie hätte  
       streicheln können. Zum Fotografieren zu nah – und zu schnell,  
       unmöglich da scharf zu stellen.

[Da/Hier] Kann man ewig rumkucken.<sup>10</sup>  
       there/here can one eternally around.look

'The penguins come so close that one could stroke them. One can look around eternally.'

---

<sup>10</sup>In an Email report from the south pole.

The generalization is that things that can be fronted can also be dropped in the *Vorfeldellipse*.<sup>11</sup> This is captured by the following schema:

This schema projects a projection of a finite verb in initial position with an element in SLASH and binds off this element in SLASH: Pollard and Sag's nonlocal feature principle ensures that the INHERITED|SLASH value of the resulting projection is the empty set. The semantic/discourse effects of this rule are ignored for the moment.<sup>12</sup>

The schema is similar to the head-filler-schema that was suggested by other authors for German verb second sentences (Pollard, 1996, p. 293; Müller, 1999, p. 97). The only difference is that there is no non-head-daughter since the *Vorfeld* is not filled. The commonalities of the two schemata are captured in the hierarchical organization of dominance schemata without the reference to surface linearization.

Alternatively one could follow Huang (1984) and use an empty operator that occupies the *Vorfeld*. In such an approach, it has to be ensured that this empty element does not occur in other positions.

### 7.3.3 Clause Types

So far, we can distinguish between verb final and verb initial clauses by making reference to the value of INV. Since verb first and verb second sentences are both INV+, we need a further feature to be able to distinguish these clause types. I suggest naming this feature v2. Normal verbal projections have the v2 value – and projections that are the result of the head-filler-schema or the topic-drop-schema are v2+.

Since the v2 feature is located inside of the SYNSEM value of a sign, nouns like those in (11b) can select for verb second sentences.

## 7.4 Empty Elements and Grammars

In this section, I want to discuss the relation of grammars with empty elements to those without empty elements. This will enable us to compare my solution with an empty copula to a solution without empty elements.

Consider for example the following German sentences:

- (16) a. Er hat nur die interessanten Bücher gelesen.  
           he has only the interesting books read

---

<sup>11</sup>This is a simplification: More oblique arguments drop less easily. Space limitations prevent me from going into a detailed discussion, but see the cited references.

<sup>12</sup>This was criticized by an anonymous reviewer of FG, but it is fully legitimate, since it is clear where the additional constraints would be located in a fully specified grammar: The constraints would be attached to the schema above.

‘He only read the interesting books.’

b. Er hat nur die interessanten gelesen.

he has only the interesting read

‘He only read the interesting ones.’

As (8b) shows, nouns may be omitted. This could be captured by the following simplified phrase structure grammar for NPs.<sup>13</sup>

- (17)  $np \rightarrow det, n'$      $det \rightarrow die$   
        $n' \rightarrow adj, n'$      $adj \rightarrow interessanten$   
        $n' \rightarrow n$              $n \rightarrow Bücher$   
                                $n \rightarrow \epsilon$

As is known from the literature on formal properties of phrase structure grammars (Bar-Hillel, Perles, and Shamir, 1961, p. 153, Lemma 4.1), such grammars can be transformed into grammars without epsilons: We eliminate all epsilon productions and add new rules for all rules where elements on the right hand side could be rewritten as the empty string. For our example this yields:

- (18)  $np \rightarrow det, n'$      $det \rightarrow die$   
        $np \rightarrow det$          $adj \rightarrow interessanten$   
        $n' \rightarrow adj, n'$      $n \rightarrow Bücher$   
        $n' \rightarrow adj$   
        $n' \rightarrow n$

The example shows that the transformation of a grammar into an epsilon free grammar may increase the number of rules. Similar techniques of epsilon elimination can be applied to HPSG grammars and in fact there are processing systems that do such grammar conversion automatically (Meurers, Penn, and Richter, 2002). The grammar in (7) and the corresponding HPSG equivalent directly encode the claim that the noun can be omitted, while this information is only implicitly contained in the rules in (8). The same would be true for a grammar that

---

<sup>13</sup>The grammar predicts that all bare determiners can function as full NPs, which is not empirically correct:

- (i) a. Ich helfe den Männern.  
       I help the men  
       b. \*Ich helfe den.  
       I help the  
       c. Ich helfe denen.  
       I help those

accounts for copulaless sentences by stipulating several constructions for questions and declarative sentences with a missing finite verb.

Using grammar transformations to get epsilon-free linguistic descriptions can yield rather complicated rules that do not capture the facts in an insightful way. This is especially true in cases where two or more empty elements are eliminated by grammar transformation. While this is not a problem for computational algorithms that deal with formally specified grammars, it is a problem for linguistic specifications. For more discussion see Müller, 2002a, Chapter 6.2.5.1, In Press.

## 7.5 Conclusion

I have shown that a theory that requires positions to be filled for certain clause types is problematic. It cannot cope with elliptic patterns where no finite verb is present or where an element in the *Vorfeld* is omitted. The only possibility to get the data described in such models is to stipulate several constructions that correspond to the observable patterns. The number of constructions that had to be stipulated in a construction-based approach would be higher than the number of empty heads that are needed in more traditional approaches and generalizations regarding combinations of syntactic material would be missed.

As an alternative, I suggested that clause types are determined with reference to features that get instantiated in immediate dominance schemata. Furthermore I provided an HPSG analysis for copulaless sentences and Topic Drop in German.

The discussion showed that an entirely surface-based syntax cannot capture regularities that can be observed in the data in an insightful way. I therefore suggest returning to more traditional approaches to German clausal syntax.

## Acknowledgements

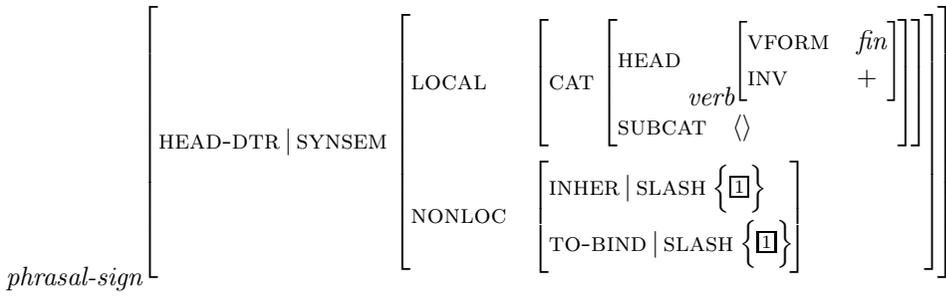
I thank three anonymous reviewers of Formal Grammar and two anonymous reviewers of HPSG 2002 for comments.

## References

- Askedal, John Ole. 1986. Zur vergleichenden Stellungsfelderanalyse von Verbalsätzen und nichtverbalen Satzgliedern. *Deutsch als Fremdsprache* 23:269–273 and 342–348.
- Bar-Hillel, Yehoshua, M. Perles, and E. Shamir. 1961. On formal properties of simple phrase-structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung* 14(2):143–172.

- Bunt, Harry and Arthur van Horck, eds. 1996. *Discontinuous Constituency*. No. 6 in Natural Language Processing. Berlin, New York: Mouton de Gruyter.
- den Besten, Hans. 1983. On the interaction of root transformations and lexical deletive rules. In W. Abraham, ed., *On the Formal Syntax of the Westgermania*, pages 47–131. Amsterdam, Philadelphia: John Benjamins Publishing Company.
- Drach, Erich. 1937. *Grundgedanken der deutschen Satzlehre*. Darmstadt: Wissenschaftliche Buchgesellschaft. 4., unveränderte Auflage 1963.
- Fries, Norbert. 1988. Über das Null-Topik im Deutschen. Forschungsprogramm Sprache und Pragmatik 3, Germanistisches Institut der Universität Lund, Lund.
- Gazdar, Gerald, Evan Klein, Geoffrey K. Pullum, and Ivan Sag. 1985. *Generalized Phrase Structure Grammar*. Cambridge, Massachusetts: Harvard University Press.
- Hoffmann, Ludger. 1997. Zur Grammatik von Text und Diskurs. In H.-W. Eroms, G. Stickel, and G. Zifonun, eds., *Grammatik der deutschen Sprache*, vol. 7.1, pages 98–591. Berlin, New York: Walter de Gruyter.
- Höhle, Tilman N. 1986. Der Begriff „Mittelfeld“, Anmerkungen über die Theorie der topologischen Felder. In W. Weiss, H. E. Wiegand, and M. Reis, eds., *Akten des 7. Internationalen Germanisten-Kongresses, Göttingen 1985*, pages 329–340. Tübingen: Max Niemeyer Verlag.
- Huang, C.-T. James. 1984. On the distribution and reference of empty pronouns. *Linguistic Inquiry* 15(4):531–574.
- Kathol, Andreas. 1995. *Linearization-Based German Syntax*. Ph.D. thesis, Ohio State University.
- Kathol, Andreas. 1997. Concrete minimalism of German. In F.-J. d’Avis and U. Lutz, eds., *Zur Satzstruktur im Deutschen*, no. 90 in Arbeitspapiere des SFB 340, pages 81–106. Tübingen: Eberhard-Karls-Universität Tübingen.
- Kathol, Andreas. 2000. *Linear Syntax*. Oxford University Press.
- Kathol, Andreas. 2001. Positional effects in a monostratal grammar of German. *Journal of Linguistics* 37(1):35–66.
- Kiss, Tibor. 1995. *Infinite Komplementation*. Tübingen: Max Niemeyer Verlag.
- Kiss, Tibor and Birgit Wesche. 1991. Verb order and head movement. In O. Herzog and C.-R. Rollinger, eds., *Text Understanding in LILOG*, pages 216–242. Springer-Verlag.
- Meurers, Walt Detmar, Gerald Penn, and Frank Richter. 2002. A web-based instructional platform for constraint-based grammar formalisms and parsing. In D. Radev and C. Brew, eds., *Effective Tools and Methodologies for Teaching NLP and CL*, pages 18–25. Proceedings of the Workshop held at 40th Annual Meeting of the Association for Computational Linguistics. Philadelphia, PA.

- Müller, Stefan. 1999. *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*. Tübingen: Max Niemeyer Verlag.
- Müller, Stefan. 2002a. *Complex Predicates: Verbal Complexes, Resultative Constructions, and Particle Verbs in German*. No. 13 in *Studies in Constraint-Based Lexicalism*. Stanford: CSLI Publications. <http://www.cl.uni-bremen.de/~stefan/Pub/complex.html>. 14.06.2004.
- Müller, Stefan. 2002b. Multiple frontings in German. In G. Jäger, P. Monachesi, G. Penn, and S. Winter, eds., *Proceedings of Formal Grammar 2002*, pages 113–124. Trento.
- Müller, Stefan. 2003. Mehrfache Vorfeldbesetzung. *Deutsche Sprache* 31(1):29–62. <http://www.cl.uni-bremen.de/~stefan/Pub/mehr-vf-ds.html>. 14.06.2004.
- Müller, Stefan. In Press. Zur Analyse der scheinbar mehrfachen Vorfeldbesetzung. *Linguistische Berichte* 199. <http://www.cl.uni-bremen.de/~stefan/Pub/mehr-vf-lb.html>. 14.06.2004.
- Paul, Hermann. 1919. *Deutsche Grammatik. Teil IV: Syntax*, vol. 3. Halle an der Saale: Max Niemeyer Verlag. 2nd unchanged edition 1968, Tübingen: Max Niemeyer Verlag.
- Pollard, Carl J. 1996. On head non-movement. In Bunt and van Horck (1996), pages 279–305. Published version of a Ms. dated January 1990.
- Reape, Mike. 1992. *A Formal Theory of Word Order: A Case Study in West Germanic*. Ph.D. thesis, University of Edinburgh.
- Reape, Mike. 1994. Domain union and word order variation in German. In J. Nerbonne, K. Netter, and C. J. Pollard, eds., *German in Head-Driven Phrase Structure Grammar*, pages 151–198. Stanford: CSLI Publications.
- Reape, Mike. 1996. Getting things in order. In Bunt and van Horck (1996), pages 209–253. Published version of a Ms. dated January 1990.
- Reis, Marga. 1980. On justifying topological frames: ‘positional field’ and the order of nonverbal constituents in German. *Documentation et Recherche en Linguistique Allemande Contemporaine* 22/23:59–85.
- Sag, Ivan A., Thomas Wasow, and Emily M. Bender. 2003. *Syntactic Theory: A Formal Introduction*. No. 152 in CSLI Lecture Notes. Stanford: CSLI Publications, 2nd edn.
- Thiersch, Craig L. 1978. *Topics in German Syntax*. Dissertation, M.I.T.
- Uszkoreit, Hans. 1987. *Word Order and Constituent Structure in German*. No. 8 in CSLI Lecture Notes. Stanford: CSLI Publications.
- van de Velde, Marc. 1978. Zur mehrfachen Vorfeldbesetzung im Deutschen. In M.-E. Conte, A. G. Ramat, and P. Ramat, eds., *Wortstellung und Bedeutung*, pages 131–141. Tübingen: Max Niemeyer Verlag.



---

# Type-Logical HPSG

CARL POLLARD

## 8.1 Introduction

Pullum and Scholz (2001) bifurcate 20th-century syntactic research frameworks into two principal paradigms: *model-theoretic syntax* (MTS, e.g. arc pair grammar, construction grammar, and HPSG) and *generative-enumerative syntax* (GES, e.g. transformational grammar and categorial grammar, including type-logical grammar (TLG)). Pullum and Scholz argue on empirical grounds for the superiority of MTS over GES. Although I think their arguments are vulnerable to criticism on a number of counts, my purpose here is not to criticize MTS but rather to argue that one need not choose between MTS and GES. More precisely, I propose a framework, *higher-order grammar* (hereafter HOG) which at once embodies both model-theoretic and proof-theoretic aspects. To put it another way, HOG is in the intersection of the MTS and GES paradigms; its MTS and GES aspects are not in competition, but rather complementary.

The comparison between TLG (Morrill, 1994, Moortgat, 1997) and HPSG (Pollard and Sag, 1994) is of particular interest because, among all the widely employed syntactic frameworks, they have been especially committed to explicit formalization of linguistic theory within logic. Given this shared concern with formal precision (to say nothing of other commonalities such as lexicocentrism and concern with computational tractability), one might well wonder why the research communities associated with these two frameworks have not merged into a single community. The principal scientific reason for the separation is that the logical foundations of the two frameworks are seemingly

incompatible.

Leaving aside for the moment semantic interpretation, in TLG words (thought of as prosodic/phonological entities) are assigned to types—formulas in a resource-sensitive logic (usually an elaboration of Lambek’s 1958, 1961 syntactic calculus) and then the assignment is extended to word strings by well-known proof-theoretic means. In HPSG, by contrast, one starts with a set of “candidate” structures (in the terminology of Carpenter (1992), the totally well-typed, sort-resolved, inequation-resolved feature structures over a given signature of sorts and features), and then discards the ones that fail to satisfy the grammar, a set of axioms (grammatical constraints) written in a (quite idiosyncratic) classical propositional logic, viz. RSRL (Richter, 2000). TLG, then, is a GES framework because the derivations are (at least) recursively enumerable, whereas HPSG is a MTS framework because the well-formed structures are models of the logical theory axiomatized by the constraints. It would appear, then, that the logics underlying the two frameworks bear no interesting relationship. But as will be shown presently, linguistic type logics and linguistic constraint logics can be seen to be intimately related, if viewed from a higher-order perspective.

HOG is an outgrowth of research over the past several years aimed at solving some of HPSG’s foundational problems, which mostly arise from the absence of functional types. Once we opt for a classical logic with functional types for expressing linguistic constraints, some form or other of higher-order logic (HOL) naturally suggests itself. In fact, the use HOL for linguistic description has been advocated by others (e.g., Moshier, 1999, Ranta, In press, Penn and Hoetmer, 2003). The present proposal, however, is unique in two respects. First, HOG employs a single HOL for all three of syntax, semantics, and phonology (as well as the syntax-semantics and syntax-phonology interfaces). Correspondingly, as in HPSG, syntactic, semantic, and phonological entities inhabit a model of a grammar. And second, HOG exploits the formal parallel between, on the one hand, the cartesian type constructors (product ( $\times$ ) and exponential ( $\Rightarrow$ )) of the typed lambda calculus that underly HOL and, on the other hand, the tensor type constructors (tensor product and directional slashes) of the Lambek calculus, thereby enabling analogs of linguistic analyses originating within TLG to be developed in an MTS setting. (A third difference, namely that HOG is *categorical* in the sense that the types and entities of a given natural language are construed, respectively, as objects and arrows of a category, with phonological and semantic interpretation as endofunctors, is discussed in (Pollard, 2004).)

## 8.2 HOG and HPSG

To clarify the connection with HPSG, a HOG can be characterized roughly as follows. (1) The grammar is written in a classical HOL rather than in RSRL. Thus HOG is free of the idiosyncrasies of RSRL such as chains (Richter, 2000) and the concomitant undecidability of finite model-checking (Kepser, 2001). (2) The types of the HOL replace HPSG's feature structure types. More specifically, feature structure types become indexed product types (Barr and Wells, 1999) HPSG partitions of a type are expressed as coproducts (model-theoretically, disjoint unions), and types SET-OF[A] are realized as functional types  $A \Rightarrow \text{Bool}$ . (3) A sign is modelled not by a feature structure but rather by the denotation (in a model of the grammar) of a closed term. (4) Unlike HPSG, HOG does not have a type Sign; instead signs are of many types (namely the ones where the interpretive endofunctors are defined). (5) "Unsaturated" signs (in HPSG terms, signs with non-null valence features) have functional types. Hence there are no valence features, so the perennial problem of how to instantiate undischarged valence features (e.g. the subject of the infinitive in *to err is human*) to satisfy total well-typedness does not arise. (6) Semantic interpretation, treated in HPSG as just another feature (CONTENT) of signs, is treated in HOG as a (schematic polymorphic) function from signs to semantic entities, that is, a type-indexed family of functions each of whose domains is one of the sign types. Since the HOL is already a lambda calculus, there is no need to encode Ty2 terms as feature structures or lambda conversion as an RSRL relation (Richter and Sailer, 2003). (7) Analogously, phonological interpretation, also treated in HPSG as a feature (PHONOLOGY) of signs, is a (schematic polymorphic) function from signs to phonological entities. Constraints on phonological entities (e.g. phonotactic constraints) can then be expressed directly as nonlogical axioms of the grammar. (8) Since there is just equality *simpliciter* rather than a distinction between type identity and token identity, the framework is free of Höhle's Problem (that a sentence containing two occurrences of the same sign is spuriously ambiguous as to whether the occurrences are type-identical or merely token identical). Some of these points will be elaborated below; others are discussed elsewhere (as specified).

## 8.3 HOG resolves the MTS-GES dichotomy

The HOG architecture provides insight into the relationship between TLG's type logic and HPSG's constraint logic, since it has analogs of both. The HOG analog of TLG's type logic is just the HOL's type

system, which, as for any typed lambda calculus ((Curry and Feys, 1958, Howard, 1980) forms an intuitionistic propositional logic with the type constructors as the logical connectives (though not a resource-sensitive one as in TLG). And the HOG analog of HPSG's constraint logic (RSRL) is just the higher-order logic of terms: both are quantificational logics with all the familiar boolean connectives, and both are used to impose well-formedness constraints on linguistic entities. TLG, however, lacks an analogue of the constraint logic. In HPSG, on the other hand, what is missing is the type logic. But in HOG there is both a type logic and a constraint logic, and the latter is the proof term calculus of the former. Thus any grammar will be a theory written in the HOL of choice, and in a model of that theory, any entity of a given type will satisfy all the constraints that the grammar imposes on entities of that type. In this respect HOG is an MTS framework (like HPSG). But at the same time, any one of the family of equivalent terms denoting that entity encodes (as per Curry-Howard) a natural-deduction proof of its type. So as long as grammars are written in such a way to ensure that the set of signs of type S (which is in one-to-one correspondence with the set of normalized proofs of type S) is recursively enumerable, HOG is also a GES framework.

## 8.4 The Logic

HOL (with two types, here called Bool (truth values) and Ent (entities), and the single type constructor  $\Rightarrow$ ), was first placed on a firm footing in the form of Church's 1940 simple theory of types (STT), which moved the term equivalence of the simply-typed lambda calculus into the object language and added constants to serve as logical connectives and quantifiers. Henkin (1950) reaxiomatized STT, added the axiom of propositional extensionality, and proved completeness with respect to the class of models which now bear his name, viz. general Henkin models (here 'general' means that there need only be enough functions to interpret all closed functional terms). Gallin (1975) showed that Ty2 (obtained by adding a type World to Henkin's HOL) was equivalent in a clearly defined sense to Montague's intensional logic IL (equipped with a suitable proof theory).

As pleasant to work with (and familiar to linguists) as Ty2 is, it is somewhat too blunt an instrument to serve as a general linguistic formalism, even with the addition of arbitrarily many basic types. Experimentation over the past several years points to the need for the following features absent in Ty2:

**Indexed Products.** The (cartesian) product type constructor  $\times$  (conjunction, in terms of the type logic), together with corresponding projection terms and pairing term constructor, is a standard feature of many HOLs and functional programming languages. This makes currying of functions an option rather than a necessity. Even better is the addition of *indexed* products, which allow the factors in product types to be indexed by arbitrary sets of labels (feature names) rather than just by natural numbers (so that the indexed projection functions are the features). Indexed products are a more standard way of doing what linguists do with feature structures.

**Coproducts.** The (cartesian) coproduct type constructor  $+$  (disjunction in the type logic, with corresponding injection functions and co-pairing term constructor) is interpreted as disjoint union in the models and is therefore ideally suited for partitioning types, as in HPSG type hierarchies.

The addition of product and coproduct (including nullary 1 and 0 respectively) to the original exponential ( $\Rightarrow$ ) constructor makes the type logic into a full intuitionistic propositional logic.

**Separation types.** Separation types (so-called by analogy with the set-theoretic axiom of separation), are subtypes defined by restricting a given type by an open boolean term, e.g. given  $S$  as a primitive type, we can define the subtype of finite sentences as the separation type

$$\text{Sfin} =_{def} [x \in S \mid \text{VFORM}(x) = \text{fin}]$$

The (separation) subtypes of any given type should form a boolean algebra. For example, to implement the now-standard analysis of case syncretism (Bayer and Johnson, 1995), given types  $\text{NPacc}$  and  $\text{NPgen}$ , we would like to define the type of syncretic accusative-genitive noun phrases as

$$\text{NPnom\_acc} =_{def} [x \in \text{NP} \mid \text{CASE}(x) = \text{nom\_acc}] = \text{NPnom} \cap \text{NPacc}$$

**Natural Number Type.** Incorporation of a natural number type  $\text{Nat}$  is another standard feature, which in effect builds an analog of the set-theoretic axiom of infinity into the logic and, inter alia, makes the Kleene- $*$  (string) type constructor definable. That is, for each type  $A$  there is a type  $A^*$  with the expected behavior of strings (see (Pollard and Hana, 2003) for linguistic motivation and application to the analysis of coordination).

**Schematic polymorphism.** From its inception HPSG has employed, at least informally, some notion of parametric polymorphism (e.g. for

sets or lists all of whose members are to be of the same type  $A$ ). In HOG we already have strings ( $A^*$ ) and sets ( $A \Rightarrow \text{Bool}$ ), but there is still a need for limited polymorphism, e.g. to define the semantic and phonological interpretation functions across the kind of sign types. Experience thus far suggests that schematic (or abbreviatory) polymorphism is sufficient; that is, no new types (or quantification over types) are introduced, but a family of functions can be defined schematically across a family of types (here, the sign types).

The higher-order categorical logic of Lambek and Scott (1986) provides a good point of departure for satisfying the foregoing desiderata: it has products, (definable) coproducts, natural number type, and (separation) subtypes. Moreover the subtypes of a given type form a Heyting algebra, which becomes boolean once the boolean axiom is imposed. This is still a bit too general, because the type of truth values (usually called  $\Omega$ ) can be an arbitrary Boolean algebra (so there can be truth values other than `true` and `false`); so in order to get bivalence it is also necessary to impose the type identity  $\Omega = 1 + 1$ , with  $\Omega$  being the truth value type and `true` and `false` being the canonical injections. (This then justifies the renaming of  $\Omega$  to `Bool`.) The models of the resulting logic are categories (abstract mathematical universes) called *bivalent boolean toposes* abstract models of typed lambda calculus with enough additional structure to interpret a propositional type and associated logical constants. Henkin models (when so augmented) are special cases of these. (Readers unfamiliar with category theory can just think of Henkin models augmented with cartesian products and lambda-definable subtypes without being led seriously astray.)

## 8.5 Syntax

A higher-order grammar is given by specifying three things: (1) the basic types; (2) the basic nonlogical constants (including their types); and (3) the constraints (nonlogical axioms).

**Basic types.** For purposes of discussion we present a HOG for a simple fragment of English (with noun and verb as the only parts of speech), starting with the following basic types: `Phon` (phonemes); `S` (sentences); `NP` (noun phrases, for the moment limited to nonquantificational ones); `N` (common noun(phrase)s, setting aside the the question of whether `N` should be analyzed as the head of `NP`); `Prop` (propositions, the semantic interpretations of declarative sentences); `Ind` (individual concepts, the semantic interpretations of noun phrases); and `Ent` (entities, the kinds of things that can be the extensions of individual concepts). (Note that the type `Bool` of truth values, the extensions

of propositions, is already supplied by the logic.) For notational convenience we also provide types for the values of what are treated in HPSG as nonboolean head features in HPSG, such as Vform (verb inflected form), Case (case), and Agr (noun agreement).

**Basic constants.** Next, we add the basic nonlogical constants. For example, to say that **nom** is a case value we include in the grammar the basic constant  $\text{nom} : 1 \rightarrow \text{Case}$ , which for familiarity we write as

$$\text{nom} \in \text{Case}$$

This means that in a model, **nom** is interpreted as a member of the set that interprets the type symbol Case (more precisely, as a function whose codomain is that set and whose domain is the singleton set  $\{0\}$ ).

Next we add functions which play the same role in HOG that head features play in HPSG, e.g.

$$\begin{aligned} \text{CASE} &\in (\text{NP} \Rightarrow \text{Case}) \\ \text{AGR} &\in (\text{NP} \Rightarrow \text{Agr}) \\ \text{VFORM} &\in (\text{S} \Rightarrow \text{Vform}) \\ \text{AUX} &\in (\text{S} \Rightarrow \text{Bool}) \\ \text{INV} &\in (\text{S} \Rightarrow \text{Bool}) \end{aligned}$$

We turn next to the specification of the lexicon. For example, to include the word *she* as a nominative third-singular-feminine nominative noun, we add the specification

$$\text{she} \in \text{NP}_{\text{nom}}/3\text{fs}$$

where the target type is defined as follows:

$$\text{NP}_{\text{nom}}/3\text{fs} =_{\text{def}} [x \in \text{NP} \mid \text{CASE}(x) = \text{nom} \wedge \text{AGR}(x) = 3\text{fs}]$$

Note that the definition does not bring the defined type into existence! Its existence is a consequence of the existence of the basic type NP, together with the subtyping provided by the logic; the definition merely provides a handy abbreviation. It is important to be aware that the entity that interprets the constant **she** is to be thought of as modelling the word *she* qua syntactic entity; it is not a phonological entity. (So far we have said nothing about what the syntactic word **she** sounds like). The view of signs (syntactic words and phrases) as inhabitants of syntactic types originates with Lambek's 1988, 1999 categorical view of his own syntactic calculus. The principal difference between Lambek's approach and the one advocated here is that they employ different type logics: Lambek calculus vs. intuitionistic propositional logic.

Next we add a couple of finite third-singular verbs to the lexicon (the definitions of the various subtypes of S and NP employed should be obvious):

$$\text{swims} \in \text{VP}_{3\text{s}}/\text{main} =_{\text{def}} (\text{NP}_{\text{nom}}/3\text{s} \Rightarrow \text{S}_{\text{fin}}/\text{main})$$

$\text{sees} \in \text{TVP3s/main} =_{\text{def}} (\text{NP} \Rightarrow (\text{NPnom}/3\text{s} \Rightarrow \text{Sfin/main}))$

These lexical items parallel lexical type assignments in TLG, but there are (at least) three important differences. First, the product and exponential type constructors involved are cartesian, not tensor. Second, there is no mention of phonology (no pairing of word strings with types). And third, words (and phrases) actually inhabit their types, rather than just being assigned to them. In the model, this means that, e.g., the word *swims* (i.e. the interpretation of the constant *swims*) is a member of the set of third-singular main (i.e. nonauxiliary) verb(phrase)s; in Curry-Howard terms, it means that *swims* encodes a (one-line) derivation (proof) of the formula  $\text{VP3s/main}$ .

Note that the lexical entry for *sees* above assigns it a curried type. But we could just as well have given the lexical entry as

$\text{sees}^\dagger \in ((\text{NP} \times \text{NPnom}/3\text{s}) \Rightarrow \text{Sfin/main})$

where the antecedent type is now suggestive of an HPSG SUBCAT list. In fact, because of the adjoint relationship between  $\times$  and  $\Rightarrow$ , either lexical entry implies the existence of the other:

$\text{sees}^\dagger = \text{uncurry}(\text{sees})$

$\text{sees} = \text{curry}(\text{sees}^\dagger)$ ;

thus whether lexical entries are curried or uncurried is strictly a matter of convenience.

In HPSG (as in relational grammar and lexical-functional grammar), grammatical functions (such as subject and complement) are taken as theoretical primitives rather than defined (as has usually been done in categorial grammar) in terms of order of functional application. In HOG, primitive grammatical functions are naturally implemented as *contrafeatures*, i.e. indices of indexed product types that occur as the antecedent in an implicative type:

$\text{TVP3s/main} =_{\text{def}} ((\text{COMP} : \text{NP}, \text{SUBJ} : \text{NPnom}/3\text{s}) \Rightarrow \text{Sfin/main})$

This development is discussed further in the full paper; for now we just mention that by using natural generalizations of currying and application to the case of indexed products, it is easy to show that the analogs of standard HPSG constraints (specifically the Head Feature Principle and the Valence Principle) are just instances of *modus ponens* with respect to the type logic. Suitably refined, this technique is also applicable to constructors corresponding to HPSG features for handling various types of unbounded dependencies such as SLASH (for “*wh*-movement” gaps, including parasitic gaps), REL (for pied-piped relative pronouns), and QSTORE (for unscoped quantificational NPs), etc.

To illustrate, a simple example is provided of how phrasal signs come about (ignoring morphosyntactic features in order to simplify the exposition). Assuming we are given the three lexical entries  $\text{kim} \in \text{NP}$ ,  $\text{sandy} \in \text{NP}$ , and  $\text{sees} \in \text{TVP}$ , we can form the term  $\text{sees}(\text{sandy})(\text{kim}) \in \text{S}$  by successive functional application. In a model, this term denotes a certain sentence (i.e. member of the set that interprets the type S). This sentence will be mapped by the semantic interpretation functor to a certain proposition (member of the set that interprets the type Prop), and by the phonological interpretation functor to a certain string of phonological words (member of the set that interprets the type Phoneme<sup>\*\*</sup>). In terms of the type logic, this term corresponds to a certain intuitionistic proof that uses *modus ponens* twice to prove the atomic formula S from the premises NP, NP, and  $\text{NP} \Rightarrow (\text{NP} \Rightarrow \text{S})$ . Thus the relationship between the derivation of the sentence and the sentence itself is that, literally, the latter is the model-theoretic interpretation of the former. Thus, in the HOG setting, the Curry-Howard isomorphism resolves the distinction between type-logical and constraint-based grammar.

**Syntactic constraints (nonlogical axioms).** As noted above, some well-established HPSG constraints, such as the Head Feature Principle and the Valence Principle, whose essential purpose is to simulate functional application, come for free. Others, such as the constraints that govern the “discharge” of NONLOCAL features, can presumably be absorbed into the general machinery for handling the corresponding type constructors (as is done in TLG), but others may have to be stated as nonlogical axioms, e.g. the English constraints on nested dependencies, which differ from (say) the Swedish ones; constraints on the distribution of parasitic gaps; the constraint that QUE-binding is possible at infinite VP or finite S but nowhere else; the pan-Germanic (but not pan-Slavic) constraint that SLASH-binders (or *Vorfeld* occupants) must be “constituents” (i.e. cannot be of cartesian product types).

One type of syntactic constraint that is straightforwardly dealt with in HOG is feature co-occurrence restrictions. For example the constraint that in English inverted sentences must be headed by a finite auxiliary can be expressed as a nonlogical axiom:

$$\forall x(\text{INV}(x) \Rightarrow (\text{AUX}(x) \wedge \text{VFORM}(x) = \text{fin}))$$

where  $x$  is a variable of type S. To take another example, consider the hypothesis that, in Polish, nominative and accusative case never syncretize. This can be expressed by the nonlogical axiom

$$\neg \exists x(x = x)$$

where  $x$  is a variable of type NP<sub>nom\_acc</sub>.

## 8.6 Semantics

The HOG approach to semantic interpretation follows up a suggestion due to (Montague, 1974, 263). Recall that in PTQ, translation is treated as a relation between English expressions (in the sense of strings of basic expressions) and terms of Montague’s intensional logic IL. Montague’s suggestion is to revise the grammar architecture so translation becomes a function from *derivations* (PTQ analysis trees) to IL terms. But our sign-denoting terms can be thought of as encoding proofs, which are analogous to PTQ-style analysis trees; so we implement Montague’s suggestion by treating semantic interpretation as a *translation* from syntactic terms to semantic terms. Following Lambek and Scott (1986), here *translation* means that: (1) each sign type translates to a semantic type; (2) cartesian products translate to cartesian products; (3) basic terms of a given sign type translate to closed terms of the corresponding semantic type; (4) the translation extends uniquely to all terms by translating lambda abstraction to lambda abstraction, application to application, and pairing to pairing. In short, translation preserves all lambda-calculus constructs. Additionally, semantic interpretation is required to be a *logical* translation (i.e. to preserve all logical constants). This is a very strong hypothesis about the nature of the syntax-semantics interface, and one that is not easily expressible in HPSG.

The details of the semantic translation are discussed elsewhere. For now we just note that the proposed semantics is *hyperintensional* in the sense of being finer-grained than the usual intensional semantics; i.e. two signs can have interpretations whose denotations coincide in every world, yet are distinct. The trick is to take propositions as primitive, and entailment as (an appropriately axiomatized) preorder on propositions (i.e. a constant of type  $(\text{Prop} \times \text{Prop}) \Rightarrow \text{Bool}$ ) and then use subtyping to *define* the type of worlds as the type of all subsets of the set of propositions which are ultrafilters (maximal consistent sets) relative to the entailment preorder. The hyperintensionality is a consequence of the fact that entailment is only a preorder, not an order (so e.g. two distinct propositions can entail each other). See (Pollard, in preparation) for details.

The upshot is that semantic interpretation is a (schematic polymorphic) function  $\text{sem}_A$  whose domain is the sign types, with NP and S mapping to Ind and Prop (propositions) respectively. The semantic types for the translations of signs belonging to nonbasic sign types is then determined by the requirement that semantic interpretation be a logical translation (as described above). The semantic interpretations

of lexical signs are assigned by constraints such as:

$$\begin{aligned} \text{sem}(\text{kim}) &= \text{kim}' \\ \text{sem}(\text{sandy}) &= \text{sandy}' \\ \text{sem}(\text{sees}) &= \text{see}' \end{aligned}$$

which in concert with the logical translation condition, uniquely determines semantic interpretation for all signs. For example:

$$\begin{aligned} \text{sem}(\text{sees}(\text{sandy})(\text{kim})) &= (\text{sem}(\text{sees}))(\text{sem}(\text{sandy}))(\text{sem}(\text{kim})) = \\ &= \text{see}'(\text{sandy}')(\text{kim}') \end{aligned}$$

## 8.7 Phonology

HOG phonology can be summarized in one sentence: phonological interpretation, like semantic interpretation, is a logical translation. This entails, *inter alia*, the following things: (1) It is impossible to tell by looking at (the term denoting) a sign what it sounds like. (2) Phonological entities are in the model and denoted by lambda terms. (3) Phonological interpretation is a translation from terms that denote signs to terms that denote phonological entities. (4) Such a translation is specified by defining it on lexical signs; the extension to phrases is uniquely determined by the requirement that phonological interpretation be a logical translation. (5) The HOL can be used to express phonotactic constraints.

This may sound like a radical program for phonology, but a good deal of it is historically grounded. The first point is closely connected with Curry's 1961 version of type-logical syntax, which insisted on a clean separation between abstract syntactic combinatorics (in Curry's term, *tectogrammar*), and the concrete realizations of syntactic entities (*phenogrammar*); in fact, Curry faulted Lambek's calculus for failing to maintain this distinction. The second point is prefigured in categorial phonology (Wheeler, 1981). The third point has a precursor in (Oehrle, 1994) (however, Oehrle's language of phonological terms did not form a logic).

The following sketch of HOG phonology is necessarily simplified and limited to segmental phonology. Our point of departure is the basic type Phoneme, so that phonological words have type  $\text{Phonword} =_{def} \text{Phoneme}^*$  and the phonological interpretations of syntactically saturated signs are strings of phonological words (type  $\text{Phonword}^*$ ). Phonological features for phonemes can be handled formally on a par with head features for saturated signs, and natural classes can be defined as subtypes of the type Phoneme. Phonotactic constraints can be expressed as nonlogical axioms, but first the phonological ontology has to

be enriched to include the kinds of entities to be constrained (e.g. syllables).

As with semantic interpretation, the value of the phonological interpretation functor **phon** on signs is uniquely determined by the values on the lexical signs (which are specified by grammatical constraints) in concert with the condition that phonological interpretation be a logical translation. As noted above, for the saturated sign types NP and S, the corresponding phonological type is Phonword\*, and so, for example, the phonological interpretation of a sign of type VP =<sub>def</sub> (NP ⇒ S) (disregarding coordinate structures) is of type Phonword\* ⇒ Phonword\*, and the phonological interpretation of a sign of type TVP is of type Phonword\* ⇒ (Phonword\* ⇒ Phonword\*).

A few examples should suffice to make this concrete. To enhance readability, we omit the type subscript on polymorphically typed constants. Additionally, we employ the standard notational abuse whereby what should denote a phonological word actually denotes a string of phonological words of length one, e.g. /kim/ instead of ⟨/kim/⟩; in fact, we compound the abuse by writing, e.g. /siz, kim/ instead of ⟨/siz/, /kim/⟩. Also, e<sub>A</sub> denotes the null A-string and  $\hat{A}$  denotes the polymorphically typed concatenation operator

$$\hat{A} \in (A^* \Rightarrow (A^* \Rightarrow A^*))$$

These are subject to the following (type-schematized) monoid constraints (with the variables all of type A):

$$\begin{aligned} \forall x (e \hat{\ } x &= x) \\ \forall x (x \hat{\ } e &= x) \\ \forall x, y, z ((x \hat{\ } y) \hat{\ } z &= x \hat{\ } (y \hat{\ } z)) \end{aligned}$$

Note that these are nonlogical axioms of the grammar, not a metalinguistically imposed term equivalence as in (Oehrle, 1994).

Phonological interpretations are assigned to lexical signs by nonlogical axioms such as the following:

$$\begin{aligned} \text{phon}(\text{kim}) &= /kim/ \\ \text{phon}(\text{sandy}) &= /sændi/ \\ \text{phon}(\text{sees}) &= \lambda x \lambda y. y \hat{\ } /siz/ \hat{\ } x \end{aligned}$$

Just as with semantic interpretation, phonological interpretation of nonlexical signs is uniquely determined by logical functoriality. For example:

$$\text{phon}(\text{sees}(\text{sandy})(\text{kim})) = (\text{phon}(\text{sees}))(\text{phon}(\text{sandy}))(\text{phon}(\text{kim})) = (\lambda x \lambda y. y \hat{\ } /siz/ \hat{\ } x)(/sændi/)(/kim/) = /kim, siz, sændi/$$

Note that the lexical entry for *sees* ensures that the first and second syntactic arguments (object and subject respectively) are phonologi-

cally realized to the right and to the left of /siz/. This is why there is no need to split the  $\Rightarrow$  constructor into  $\backslash$  and  $/$ : the directionality of combination is moved out of the syntax and into the phonology (and its interface with syntax). More generally, the resource sensitivity of language is relocated from syntax (where TLG has it) into the phonology; thus the syntactic type logic is not a Lambek calculus but just an ordinary intuitionistic propositional logic with all three of the structural rules (contraction, interchange, and weakening).

This point is perhaps best conveyed in an intuitive, nontechnical way as follows: in TLG, the syntax has to keep track of word order and word occurrences, hence the need for a directional and linear syntactic type theory. But in HOG, the syntax is freed of this responsibility, because every time a syntactic word is used in a derivation, an occurrence of its phonological interpretation shows up, appropriately linearized, in the string of phonological words. Adapting the sort of economic metaphor favored by linear logicians: logical constants come for free, but every time you use a nonlogical constant in a syntactic proof, you have to pay for it with a spoken word (by saying its name out loud).

As is well known (Zaenen and Karttunen, 1984, Sag et al., 1985, Pullum and Zwicky, 1986) any syntactic theory must distinguish between ambiguity (two or more signs with the same phonology) and something else variously known as neutrality, nondistinctiveness, syncretism, indeterminacy, or underspecification. Here we sketch the HOG treatment of this distinction, starting with ambiguity. In the simplest case (so-called argument ambiguity), we have two distinct words with the same phonological interpretation, e.g., *bank* ‘riverside’ and *bank* ‘financial institution’:

$\text{bank}_1 \in N$

$\text{bank}_2 \in N$

General properties of the cartesian product (and the associated projection terms and pairing term constructor) ensure that the presence of these two lexical entries is equivalent to the presence of the single conjunctive specification

$(\text{bank}_1, \text{bank}_2) \in N \times N$

Of more interest is so-called functor ambiguity, where the two signs in question both have implicative types with the same consequent, e.g. main verb *can* and modal *can*:

a. I can tuna.

b. I can get a better job if I want to.

c.\*I can tuna and get a better job if I want to.

In this case the pair of ambiguous words has type (ignoring morphosyntactic features)

$$(\text{NP} \Rightarrow \text{VP}) \times (\text{VP} \Rightarrow \text{VP})$$

which, by the intuitionistically valid law of disjunctive syllogism and its converse, is equivalent (in the sense that the functions denoted by the proofs in both directions are each other's inverses) to the type

$$(\text{NP} + \text{VP}) \Rightarrow \text{VP}$$

Intuitively, since  $+$  (cartesian coproduct) is disjunction in the type logic, this says that *can* can take as its complement something which is either an NP or a VP (but not, as will be shown below, a coordination of an NP and a VP, which is neither). Formally, this treatment parallels the standard TLG treatment of ambiguity in Morrill (1990), which also employs cartesian coproduct (written  $\vee$  in the Lambek calculus setting, where in terms of the type logic it is linear additive disjunction, in spite of being incorrectly characterized as boolean in most of the relevant TLG literature). Unfortunately, the standard TLG treatment of coordination (Morrill, 1990, Bayer and Johnson, 1995, Bayer, 1996, hereafter MBJ), which builds on Steedman's polymorphic typing of coordinate conjunctions as  $A \setminus A / A$ , wrongly generates such ungrammatical examples side by side with grammatical examples such as

John is rich and an excellent cook.

because coordinate structures are analyzed as having disjunctive types (here  $\text{AP} \vee \text{NP}$ , with the neutral functor *is* receiving the lexical type assignment  $\text{VP} / (\text{AP} \vee \text{NP})$ ). Thus the TLG account fails to distinguish functor ambiguity from functor neutrality. To take another example, the MBJ account predicts both of the following to be grammatical:

- a. \*Mary wants to go and John to go.
- b. I would like to leave town early and for you to go with me.

As pointed out by Whitman (2002), the MBJ account also fails to distinguish between argument ambiguity and argument neutrality (which includes case syncretism as a special case), so that all instances of homophony between slots in the inflectional paradigm of a word are wrongly predicted to syncretize (see Dyla (1984) for relevant counterexamples). Moreover, the MBJ account is inconsistent with the standard TLG frame-semantics approach to phonological interpretation (Heylen, 1996, 1997, Moortgat, 1997, Carpenter, 1998). On that account, if  $S$  is the stringset that phonologically interprets AP and  $T$  is the stringset that phonologically interprets NP, then *rich and an excellent cook* should be in their union; hence it must lie in either  $S$  or in  $T$ ; but

it does not. Faced with these difficulties, Whitman suggests abandoning the syntactic distinction between ambiguity and neutrality (so that in principle neutralization is always an option, subject only to pragmatic factors). Alternatively, Morrill (p.c.) suggests the possibility of distinct phonological entities with no audible difference (more precisely, a phonological entity is not just a string, but rather an ordered pair of a string and an integer).

Some of the problems discussed above have also been addressed within recent HPSG literature, most recently by Sag (2003), which proposes a relaxation of the requirement that feature structures be sort-resolved. In the absence of a precise formalization, this proposal is hard to assess. (Note that the model theory of RSRL precludes any entities which belong to a sort without belonging to one of its maximally specific subsorts.)

Pollard and Hana (2003) propose the following HOG analysis of neutrality and coordination of unlikes. First, the treatment of syncretism (say, for a language with nominative and accusative case) follows Levine et al. (2001) in employing a nonstandard inventory of `CASE` values: `pnom` (pure nominative), `pacc` (pure accusative), and `nom_acc` (syncretic between nominative and accusative). Then `NPnom` and `NPacc` are defined as subtypes of `NP` as follows:

$$\begin{aligned} \text{NPnom} &=_{\text{def}} [x \in \text{NP} \mid \text{CASE}(x) = \text{pnom} \vee \text{CASE}(x) = \text{nom\_acc}] \\ \text{NPacc} &=_{\text{def}} [x \in \text{NP} \mid \text{CASE}(x) = \text{pacc} \vee \text{CASE}(x) = \text{nom\_acc}] \end{aligned}$$

Also we define

$$\text{NPnom\_acc} =_{\text{def}} [x \in \text{N} \mid \text{CASE}(x) = \text{nom\_acc}] = \text{NPnom} \cap \text{NPacc}$$

That is, case syncretism is handled not by the type logic's conjunction (cartesian product, which is appropriate only for non-neutralizing ambiguity) but rather by the (genuinely boolean) intersection of separation subtypes.

Pollard and Hana's analysis of coordination employs a schematic polymorphic type `GEN[A]` where `A` can be instantiated as any type of kind `Sign`. That is, for each sign type `A`, there is a type `GEN[A]` of "generalized `A`", where a generalized `A` is a sign that is either an `A` or a coordinate structure whose conjuncts are generalized `A`'s. To ensure that, for each sign type `A`, `A` is actually a subtype of `GEN[A]`, we add to the term logic a type-schematized family of constants  $gen_A \in (A \Rightarrow \text{GEN}[A])$  together with a type-schematized set of constraints which ensure that in any model, each sign type is embedded in a one-to-one fashion into its generalization. (In the model, each of these constants is interpreted as a function that maps each sign of a certain type into a string of signs of length one). All that remains to complete the analysis

of coordination is to add type-schematized conjunctions (e.g.,  $\text{and}_A$  and  $\text{or}_A$ ) to the lexicon. What drives the analysis is the polymorphic typing of these constants, which is not  $A \Rightarrow (A \Rightarrow A)$  (as would be suggested by the Steedman typing), but rather the type

$$\text{GEN}[A]^+ \Rightarrow (\text{GEN}[A] \Rightarrow \text{GEN}[A]).$$

The phonological interpretation functor will ensure that the nonempty list argument shows up to the left of the conjunction and the other argument to the right. (Note that, in order for coordinate structures to serve as arguments to other signs, we must systematically retype our unsaturated lexical entries from  $A \Rightarrow B$  to  $\text{GEN}[A] \Rightarrow B$ , e.g. VP is redefined from  $\text{NP} \Rightarrow \text{S}$  to  $\text{GEN}[\text{NP}] \Rightarrow \text{S}$ .)

## References

- Barr, M. and C. Wells. 1999. *Category Theory for Computing Science 3rd edition*. Montreal: CRM.
- Bayer, S. 1996. The coordination of unlike categories. *Language* 72(3):579–616.
- Bayer, S. and M. Johnson. 1995. Features and agreement. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, vol. 33, pages 70–76. ACL.
- Carpenter, B. 1998. *Type-Logical Semantics*. Cambridge, MA: MIT Press.
- Carpenter, R. 1992. *The Logic of Typed Feature Structures*. New York: Cambridge University Press.
- Church, Alonzo. 1940. A formulation of the simple theory of types. *Journal of Symbolic Logic* 5:56–68.
- Curry, H. 1961. Some logical aspects of grammatical structure. pages 56–68.
- Curry, H. and R. Feys. 1958. *Combinatory Logic*. Amsterdam: North-Holland.
- Dyla, S. 1984. Across-the-board dependencies and case in Polish. *Linguistic Inquiry* 15(4):701–705.
- Gallin, D. 1975. *Intensional and Higher Order Modal Logic*. Amsterdam: North Holland.
- Henkin, L. 1950. Completeness in the theory of types. *Journal of Symbolic Logic* 15:81–91.
- Heylen, D. 1996. On the proper use of booleans in categorial logic. In *Proceedings of the Conference on Formal Grammar 1996*. Prague.
- Heylen, D. 1997. Generalization and coordination in categorial grammar. In *Proceedings of the Conference on Formal Grammar 1997*. Aix-en-Provence.
- Howard, W. 1980. The formulae-as-types notion of construction. pages 479–490.
- Kepser, S. 2001. On the complexity of RSRL. In *Proceedings of FG-MOL 2001, Electronic Notes in Theoretical Computer Science 53*. Kluwer.

- Lambek, J. 1958. The mathematics of sentence structure. *American Mathematical Monthly* 65:154–169.
- Lambek, J. 1961. On the calculus of syntactic types. pages 166–178.
- Lambek, J. 1988. Categorical and categorial grammars. In R. Oehrle, E. Bach, and D. Wheeler, eds., *Categorial Grammars and Natural Language Structures*, pages 297–317. Dordrecht: Reidel.
- Lambek, J. 1999. Deductive systems and categories in linguistics. In H. J. Ohlbach and U. Reyle, eds., *Logic, Language, and Reasoning: Essays in Honour of Dov Gabbay*, pages 279–294. Dordrecht: Kluwer.
- Lambek, J. and P. Scott. 1986. *Introduction to Higher Order Categorical Logic*. Cambridge: Cambridge University Press.
- Levine, R., T. Hukari, and M. Calcagno. 2001. Parasitic gaps in English: some overlooked cases and their theoretical implications. pages 181–222. Cambridge, MA: MIT Press.
- Montague, R. 1974. The proper treatment of quantification in ordinary English. In R. Thomason, ed., *Formal Philosophy*, pages 247–270. New Haven, CT: Yale University Press.
- Moortgat, G. 1997. Categorical type logics. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language*. New York: Elsevier.
- Morrill, G. 1990. Grammar and logical types. In M. Stokhof and L. Torenvliet, eds., *Proceedings of the Seventh Amsterdam Colloquium*, pages 429–450. Amsterdam: Institute for Logic, Language, and Information.
- Morrill, G. 1994. *Type Logical Grammar: Categorical Logic of Signs*. Dordrecht: Kluwer.
- Moshier, M.A. 1999. HPSG as type theory. In J. Ginzburg, L. Moss, and M. de Rijke, eds., *Logic, Language, and Computation*, vol. 2. Stanford, CA: CSLI.
- Oehrle, R. 1994. Term-labelled categorial type systems. *Linguistics and Philosophy* 17:633–678.
- Penn, G. and K. Hoetmer. 2003. *In search of epistemic primitives in the English Resource Grammar (or Why HPSG can't live without higher-order datatypes)*. East Lansing.
- Pollard, Carl. 2004. Higher-order categorial grammar. In *Submitted for Categorical Grammar 2004*.
- Pollard, Carl. in preparation. Hyperintensions in higher-order categorial logic Submitted for LoLa 2004.
- Pollard, Carl and Jiri Hana. 2003. Ambiguity, neutrality, and coordination in higher-order grammar. In *Proceedings of Formal Grammar 2003*.
- Pollard, C. and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago and CSLI, Stanford.
- Pullum, G. and B. Scholz. 2001. On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In P. de Groote, G. Morrill, and C. Retoré, eds., *LACL 2001, LNAI 2099*, pages 17–43. Berlin: Springer-Verlag.

- Pullum, G. and A. Zwicky. 1986. Phonological resolution of syntactic feature conflict. *Language* 62(4):751–773.
- Ranta, A. In press. Grammatical Framework: a type-theoretical grammar formalism. *Journal of Functional Programming* .
- Richter, F. 2000. *A Mathematical Formalism for Linguistic Theories with Application to Head-Driven Phrase Structure Grammar and a Fragment of German..* Ph.D. thesis, University of Tübingen.
- Richter, F. and M. Sailer. 2003. Basic concepts of lexical resource semantics. Course materials.
- Sag, I. 2003. Coordination and underspecification. In J.-B. Kim and S. Wechsler, eds., *The Proceedings of the 9th International Conference on HPSG*, pages 267–291. Stanford: CSLI.
- Sag, I., G. Gazdar, T. Wasow, and S. Weisler. 1985. Coordination and how to distinguish categories. *Natural Language and Linguistic Theory* 3:117–171.
- Wheeler, D. 1981. *Aspects of a Categorical Theory of Phonology*. Ph.D. thesis, University of Massachusetts.
- Whitman, P. 2002. *Category Neutrality: A Type-Logical Investigation*. Ph.D. thesis, Department of Linguistics, The Ohio State University.
- Zaenen, A. and L. Karttunen. 1984. Morphological non-distinctiveness and coordination. In *Proceedings of the First Eastern States Conference on Linguistics*, pages 309–320.

---

# About Spilled Beans and Shot Breezes: A New Word-level Approach to Idioms<sup>1</sup>

JAN-PHILIPP SOEHN

## 9.1 Motivation

Idioms are omnipresent in everyday language. Nonetheless, they have been widely neglected by linguists developing grammar fragments. And even where an account for idioms has been given, most approaches have their shortcomings (cf. Riehemann, 2001, ch. 4).

In this contribution we want to focus on decomposable and non-decomposable idioms<sup>2</sup> and discuss technical aspects of an HPSG analysis. For reasons of space we will have to neglect detailed linguistic corpus data. By “idiom” we mean idiomatic expressions that do not form complete sentences as would be the case for e. g. *His bark is worse than his bite*.

(33) *make waves* (“cause trouble”)

(34) *spill the beans* (“divulge a secret”)

The expressions in (33) and (34) are instances of decomposable idioms, i. e. their meaning can be derived from the idiom parts. Note that idiom

---

<sup>1</sup>The research to the paper was funded by the *Deutsche Forschungsgemeinschaft*. I am grateful to Stefan Müller, Christine Römer, Manfred Sailer, Adrian Simpson and the reviewers of FGNancy for their comments and Michelle Wibraham for her help with English. My email address: [jp.soehn@uni-jena.de](mailto:jp.soehn@uni-jena.de)

<sup>2</sup>Cf. Nunberg et al. (1994), e. g., for this distinction.

parts are not necessarily to be understood literally. In (33), e. g., we can attribute the meaning “cause” to *make* and “trouble” to *waves*. The idiomatic meaning of the whole idiom consists of the idiomatic meanings of its parts.

Where this is not the case, an idiom is non-decomposable: the meaning of the whole phrase has nothing to do with the meaning of the words the idiom consists of. Consider (35) and (36):

(35) *saw logs* (“snore”)

(36) *shoot the breeze* (“chat”)

It is not clear how to assign the meaning “snore” to the words *saw* and *logs*, the same holds for “chat”.

After providing the prerequisites for a revised approach to idioms in the next section, we analyse instances of these idiom classes and discuss previous approaches. Then we briefly sketch how our proposal fits into the overall architecture of HPSG and illustrate this by examining how to merge it with a fronting analysis of German idioms.

## 9.2 Lexemes and Listemes

Before we present our analysis, we point out a way that enables us to select a specific word. This forms a prerequisite of our approach.

Idioms often consist of particular words which cannot be substituted by semantically equivalent terms. It seems in general that each word has a unique “identity” with an idiosyncratic behavior. The possibility to select a particular word would, thus, be a useful feature. Up to now, there has been a discussion about the necessity of having such kind of selection. One could argue that any data in question are to be handled as Constructions or collocations. But why impose such a “heavy thing” on an expression like *to furrow one’s brow*? Would it not be plausible that the verb *furrow* simply selects a word of the form *brow*? For perfect tense in German a main verb has to be combined with the right auxiliary (*haben/sein*; in HPSG with the attribute AUXF, cf. Heinz and Matiasek, 1994, p. 222). Here one does nothing other than to select a particular lexeme.

Krenn and Erbach (1994) made an important contribution to idiom analysis within the HPSG framework. They suggested selecting particular lexemes via their feature LEXEME below CONTENT INDEX. This idea of having lexeme information in the CONTENT is questionable. A lexeme combines phonetic, morphological, syntactic and semantic properties all together, not only semantic information. Besides, their approach had several technical shortcomings (cf. Soehn and Sailer, 2003). We therefore propose that the LEXEME approach has to be discarded.

A different concept that can help to distinguish between individual words is that of a listeme<sup>3</sup>. As the concept holds the characteristic of listedness in a lexicon, we use it in our grammar to identify a particular word or phrase. Thus, we insert LISTEME into the feature geometry below CATEGORY, emphasizing the morpho-syntactic character of information. More precisely, we put it below HEAD. This has two consequences: firstly, it is available for selection, as a HEAD value is below SYNSEM. Secondly, the LISTEME value of a projection is the same as the one of the head, as all HEAD features “percolate” according to the HEAD-FEATURE-PRINCIPLE. For our *furrow*-example that means that a modified direct object *his heavy brow* still has the same LISTEME value as *brow* alone.

A third question to address is the handling of pronominalization. It is necessary that pronouns have the same LISTEME value as their antecedent.<sup>4</sup> In Krenn and Erbach’s approach this was the major motivation of putting the LEXEME feature in the INDEX. To emulate this quality, we propose a constraint ensuring that each pronoun which is co-indexed with an antecedent takes over its LISTEME value. In the lexical entries of pronouns that value would be left underspecified in that way, that it consists of a disjunction of an identifying value (*she, her, etc.*) and a wildcard. In case of co-indexation the wildcard is identical to the LISTEME value of the antecedent and – by virtue of the constraint – becomes the actual and concrete LISTEME value of the pronoun. An informal description of such a pronoun constraint is illustrated in (37).

(37) PRONOUN-LISTEME-CONSTRAINT:

If a pronoun is co-indexed with an antecedent, it takes over the LISTEME value of that antecedent. Otherwise the LISTEME value of this pronoun is that of the other disjunct.

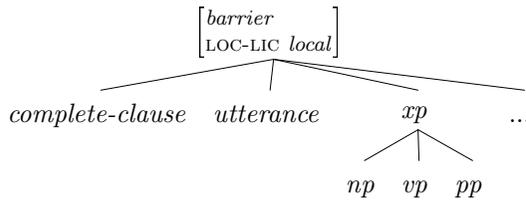
The value of LISTEME is an atomic sort as *brow, heavy, furrow, take, she* etc. In order to identify listemes for the same words having different meanings, we use numeric indices just as in a dictionary.

In summary, discarding the LEXEME approach, we propose a more adequate solution for the problem of selecting particular words, at least with respect to terminology, technical feasibility and the feature geometry. We introduce a feature LISTEME which is appropriate for the sort *head* taking atomic sorts as its value.

---

<sup>3</sup>This term has been introduced by Di Sciullo and Williams (1988) for a sign that is listed in the lexicon.

<sup>4</sup>E. g. in the phrase *He furrowed it.* the pronoun has the same LISTEME value as its antecedent, satisfying the subcategorizational requirement of the verb.

FIGURE 1 Sort hierarchy for *barrier*

### 9.3 Licensing Contexts

Getting to the analysis, we have to define a second attribute in the feature geometry. We declare objects of sort *sign* to bear a list-valued feature COLL (Context Of Lexical Licensing), first introduced by Richter and Sailer (1999). The COLL list may contain objects of sort *barrier*. These *barriers* are particular nodes in the syntactic configuration, like XPs, complete clauses or utterances (a complete clause with an illocutionary force). The concept of barriers is borrowed from the tradition of generative grammar, where these form boundaries for government and binding principles. We avail ourselves of this concept and use similar barriers for the restriction of distributional phenomena.

*barrier* objects have an attribute LOCAL-LICENSER (LOC-LIC) which has a value of sort *local*. In the lexical entry of an idiomatic word one can thus specify a *barrier* on its COLL list with a specific *local* configuration. Subsorts of *barrier* are illustrated in figure 1. The subsorts of *barrier* correspond to nodes in the syntactic tree with particular properties. The following relations identify the nodes which relate to the barriers *complete-clause* and *vp*, respectively.<sup>5</sup>

$$\forall \mathbb{I} \left( \text{is\_complete-clause}(\mathbb{I}) \leftrightarrow \left[ \begin{array}{l} \mathbb{I} \text{ phrase} \\ \text{SS} \left[ \begin{array}{l} \text{STATUS } complete \\ \text{LOC CAT} \left[ \begin{array}{l} \text{HEAD } verb \\ \text{SUBCAT } elist \end{array} \right] \end{array} \right] \end{array} \right] \right)$$

$$\forall \mathbb{I} \left( \text{is\_vp}(\mathbb{I}) \leftrightarrow \left[ \begin{array}{l} \mathbb{I} \text{ phrase} \\ \text{SS} \left[ \begin{array}{l} \text{STATUS } incomplete \\ \text{LOC CAT} \left[ \begin{array}{l} \text{HEAD } verb \\ \text{SUBCAT } nelist \end{array} \right] \end{array} \right] \end{array} \right] \right)$$

The LICENSING-PRINCIPLE (informally in 38) makes sure that if there is a barrier specified on a word's COLL list, there is an actual barrier in the phrase our word occurs in. This barrier must fulfill the *local* requirements and it has to be minimal, i. e., there is no other potential barrier of the same kind between the word and the actual

<sup>5</sup>Cf. (Richter, 1997, pp. 68f) for the STATUS feature.

barrier.

(38) LICENSING-PRINCIPLE (LIP):

For each *barrier* object on the COLL list of a sign  $x$  and for each phrase  $z$ :

the LOCAL value of  $z$  is identical with the LOC-LIC value,  
 iff  $z$  dominates  $x$ ,  $z$  can be identified as the barrier specified<sup>6</sup> and  
 $z$  dominates no sign  $y$  which in turn dominates  $x$  and forms an  
 equivalent barrier.

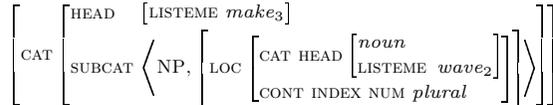
Hence, a word for which a barrier is defined cannot occur elsewhere; its distribution is already specified in the lexical entry.

This concludes the description of technical requirements for our approach to idioms. Note that we have defined a very small number of new sorts and attributes to be included in the signature. All idiosyncratic information comes from the lexicon, as we will see in the next section.

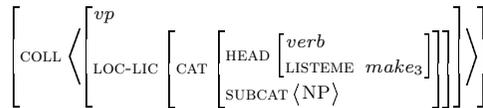
9.3.1 Decomposable Idioms

Let us show how a decomposable idiom can be analysed with our proposal. Take for instance the idiom in (33) *make waves*<sup>7</sup>. We can assign the meanings “cause” and “trouble” to *make* and *waves* and assume that there are two lexical entries for the idiomatic usage of these words.<sup>8</sup> The meaning of the whole idiom can be calculated in a regular compositional way.

The idiomatic *make* subcategorizes for a plural noun with the word form *wave* (the idiomatic version) creating a VP with the meaning “cause trouble”.



*wave*<sub>2</sub> for its part bears a non-empty COLL list which looks as follows:



The distribution of the idiomatic noun *waves* is restricted in that it must be the complement of idiomatic *make*. The LIP makes sure that the specified *vp* on the COLL list is identical to the actual VP containing

<sup>7</sup>as in “Italian film makes waves” from <http://news.bbc.co.uk/1/hi/entertainment/film/3171907.stm> (All weblinks were found by Google on 01-27-2004)

<sup>8</sup>Another meaning of the idiom is “call attention” or “attract interest”.

*make* and *waves*. Defining the barrier as a VP correctly implies that passivization of this idiom is not possible.<sup>9</sup>

Our example *spill the beans*<sup>10</sup> can be analysed analogously. As we assume regular syntactic composition to be in force, we predict that different specifiers (*some beans*) or modifications (as *some very compromising beans*) are grammatical.

A special case of the idiom not occurring in its canonical form is that of pronominal reference. In fact, pronominalization is quite hard to handle in idiom analysis. Cf. the following example:

- (39) *Eventually she spilled all the beans. But it took her a few days to spill them all.*<sup>11</sup>

Here the pronoun *them* refers back to the idiomatic *beans*. As described in section 2 a pronoun has the same LISTEME value as its antecedent, so *them* gets its correct meaning. This being the case, the subcategorization requirements of idiomatic *spill* in both clauses are satisfied. The antecedent of *them* in turn is licensed by its own COLL value stating that the idiomatic *beans* can only occur together with the verb *spill* in its idiomatic use. The barrier is a *complete-clause* which allows e. g. passive or relative constructions. Thus, our proposal can handle pronominalization data, too.

### 9.3.2 Non-decomposable Idioms

For idioms that have a non-decomposable meaning we define phrasal lexical entries (PLE), according to Sailer (2003) and following the idea of Gazdar et al. (1985). PLEs are lexical entries for syntactically complex expressions. Thus, they have properties of both words and phrases. As words, they are licensed by their lexical entry. As phrases, lexical rules cannot apply to them and syntactic operations like topicalization can be excluded by defining structural requirements in their DTRS attribute.

The semantics of a non-decomposable idiom is defined in its PLE. The parts of such an idiom are licensed by their ordinary lexical entries. In the syntactic structure of a sentence containing such an idiom there is a node where all necessary idiom parts are present. This node is either licensed by regular compositional principles or by a PLE. If a

---

<sup>9</sup>Riehemann found 5 examples out of 243 (2%) where the idiom parts do not occur within the same VP. If one wants to account for those (including passivization and a relative clause) the barrier is simply to be set accordingly.

<sup>10</sup>as in “Tom Cruise has spilled the beans on Nicole Kidman’s relationship with US musician Lenny Kravitz.” from <http://www.smh.com.au/articles/2003/11/29/1070081589377.html?from=storyrhs>

<sup>11</sup>Riehemann (2001), p. 207

PLE is applied, it replaces the semantics computed so far with its own meaning.

According to standard HPSG assumptions we adopt Immediate Dominance Schemas that license ordinary phrasal signs. In order to exclude the application of ID-Schemas to a phrase licensed by a PLE we can redefine the ID-PRINCIPLE in the following way:

$$\left[ \begin{array}{l} \textit{phrase} \\ \text{COLL } e\text{-list} \end{array} \right] \rightarrow \left( \begin{array}{l} \text{HEAD-COMPLEMENT-SCHEMA} \vee \text{HEAD-ADJUNCT-SCHEMA} \vee \\ \text{HEAD-MARKER-SCHEMA} \vee \text{HEAD-FILLER-SCHEMA} \end{array} \right)$$

Accordingly, we have to change all principles of grammar that are concerned with regular combination of signs (like the HEAD-FEATURE-PRINCIPLE or the SEMANTICS-PRINCIPLE) in such a way that they only apply to phrases bearing an empty COLL list. This can simply be done by adding a line in the antecedent (remember that the principles consist of an implication) stating [COLL *e-list*].

In order to specify which lexical entries must have an empty COLL list, we introduce subsorts of *listeme*, namely the sorts *coll\_listeme* and *no\_coll\_listeme*, and make the following constraint:

$$\left[ \begin{array}{l} \textit{sign} \\ \text{SS LOC CAT HEAD LISTEME } no\_coll\_listeme \end{array} \right] \rightarrow [\text{COLL } e\text{list}]$$

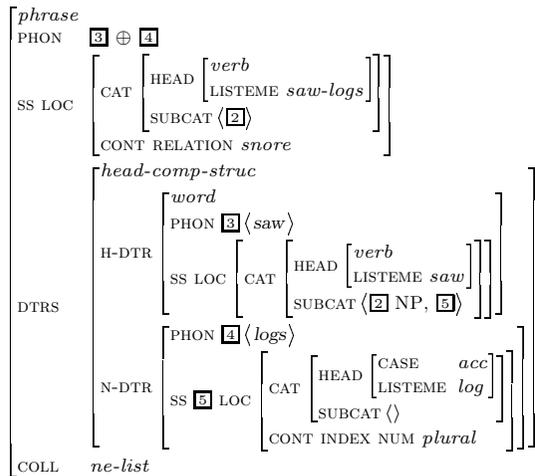
Note that all lexical entries have different values of LISTEME and, conversely, the set of all LISTEME values covers the entirety of lexical entries.

We have now made a distinction between regular phrasal signs which have an empty COLL list and non-regular or idiomatic phrases having a non-empty COLL list.<sup>12</sup> Thus, in a PLE of an idiom like (35) *saw logs*<sup>13</sup> we define its COLL list as non-empty. Besides, this idiom cannot be passivized without losing its idiomatic reading. Passivization is already excluded by the nature of the PLE itself: an object in accusative case is required and thus, *logs* can never occur as the subject.

---

<sup>12</sup>The distribution of COLL values could be easily constrained by another principle which we omit here for reasons of space.

<sup>13</sup>as in “Two young boys stand by their mother’s bed while she saws logs in her sleep.” from [http://www.collegestories.com/filmfrat/igby\\_goes\\_down.html](http://www.collegestories.com/filmfrat/igby_goes_down.html)



In defining a non-empty COLL value, we provide a unified way to treat decomposable and non-decomposable idioms, marking their quality of being idiomatic. Parts of decomposable idioms bear a non-empty COLL list, which restricts their occurrence to certain contexts. Nondecomposable idioms also have a non-empty COLL list, exempting them from regular syntactic and semantic principles.

In addition, the occurrence of nondecomposable idioms can be restricted to certain contexts via the same feature. This is important for idiomatic intensifiers, among others, like *as a sandboy* in *to be happy as a sandboy* or *as a kite* in *to be high as a kite*.

#### 9.4 How does it fit in?

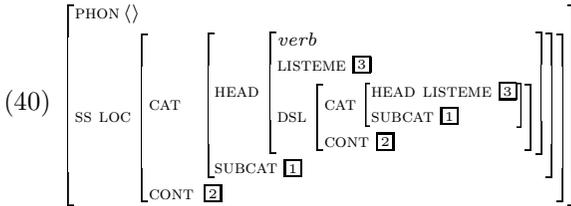
As we have introduced a (rather manageable) number of new sorts, principles and features, and explained how to adapt existing grammar principles to our approach, it is not difficult to imagine how our proposal fits into the overall HPSG Grammar. Non-idiomatic signs bear an empty COLL list and thus are exempt from any consequence of our analysis. In addition, the fact that each word has now a value of its LISTEME feature is only important where an access to LISTEME is needed. At the same time our proposal can be combined smoothly with other modules of grammar concerning idioms.

In order not to settle for this mere claim, we will illustrate it by means of an idiom-independent part of grammar – topicalization and verb movement, for instance.

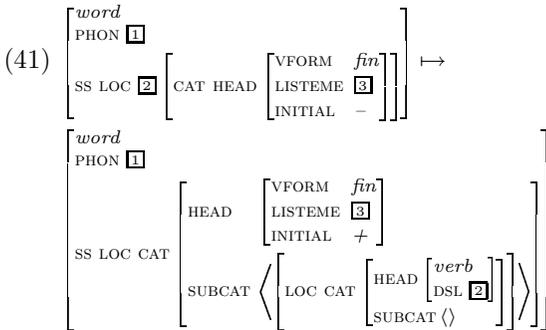
In a declarative sentence in German the finite verb occupies the second position. Such sentences can be derived from verb-last-clauses by movement of the verb. The position in front of the verb is called

*Vorfeld* and can contain an argument or an adjunct. The positioning of an element in the *Vorfeld* is usually analyzed as a nonlocal dependency, which accounts for the vast majority of declarative clauses. A verb movement analysis by Borsley and Kiss, which is discussed in Müller (2004), can account for this phenomenon.

First, a special lexical entry for a verbal trace (40) is introduced, where verbal movement is treated as a local phenomenon. Thus, for a verbal trace there is no application of the standard unbounded dependency analysis (cf. Pollard and Sag (1994)). The *local*-valued feature DSL allows the valence structure of the moved verb to be available along the head projection line. The lexical entry of the verbal trace (slightly adapted from Müller, 2004, p. 22) looks as follows:



Secondly, Müller outlines a lexical rule (41) for a special version of the finite verb which is moved. The verb being licensed by this rule takes the projection of the verbal trace as its argument. A phrase is only grammatical if the valence properties of the verb are identical to those of the verbal trace. The rule takes a verb in non-initial position as input and outputs a verb in initial position (marked by the verb feature INITIAL). The lexical rule for initial verb position (V1-LR, slightly adapted from Müller, 2004, p. 23) is defined in the following way:



To illustrate this analysis together with our approach, we take the German idiom *jemandem den Garaus machen* (to cook so.'s goose, 'to kill someone'). We can roughly state its decomposable meaning as "to

put an end to”.

- (42) Zucker, Salz, chemische Aromen und Geschmacksverstärker  
 sugar salt chemical flavors and flavor enhancers  
**machen** dem natürlichen Geschmacksempfinden **den Garaus**.  
 make the natural taste the end  
 ‘Sugar, salt, chemical flavorings and flavor enhancers destroy our  
 natural taste.’<sup>14</sup>
- (43) **Den Garaus macht** den Seglern die Langleinenfischerei.  
 the end makes the sailors the long line fishing  
 ‘Long line fishing encroaches upon sailors.’<sup>15</sup>

The sentence in (42) shows that the NP can appear detached from the verb. In (43) the NP is fronted and has been extracted from the original VP. The structure of (43) is the following:

- (44) [Den Garaus]<sub>i</sub> macht<sub>j</sub> den Seglern die Langleinenfischerei <sub>-i -j</sub>.

We can analyse this sentence by means of the standard unbounded dependency analysis, the approach in Müller (2004) and our proposal. Figure 2 shows the syntax tree of (44) including all relevant information.

Let us begin to explain this figure with the extraction trace. The unbounded dependency analysis involves a lexical entry for a trace where the local value ( $\square$ ) of the extracted element is identical to an element in the INHER SLASH list. The NONLOCAL-FEATURE-PRINCIPLE guarantees that this value “percolates” up the tree. By the HEAD-FILLER-SCHEMA this SLASH value is bound via TO-BIND SLASH because the LOCAL value of the filler is the same as the SLASH value.

Further, the verbal trace attracts the arguments of the moved verb and puts it onto both its SUBCAT list and its DSL CAT SUBCAT list. The HEAD-FEATURE-PRINCIPLE enforces the presence of the DSL value along the head projection line. The verb *machen* for its part is input to the V1-LR (41). The output subcategorizes for a sign ( $\square$ ) whose DSL value is identical to the LOCAL value of the input ( $\square$ ).

This concludes the pure fronting analysis. So far as our proposal is concerned, the important parts are the values of COLL and LISTEME respectively.

The filler *den Garaus* has a COLL value defining a barrier<sup>16</sup> bearing

<sup>14</sup><http://www.erichlutz.de/publikationen/globus.html>

<sup>15</sup>Salzburger Nachrichten, 15.11.2000; Found with COSMAS II by IDS Mannheim

<sup>16</sup>We do not specify the barrier to be a VP deliberately. If *den Garaus* occurs *in situ*, a *vp* would be correct, but in this case we have a *complete-clause* (if not even an *utterance*) as barrier. In order to not exclude such fronting cases, the barrier has

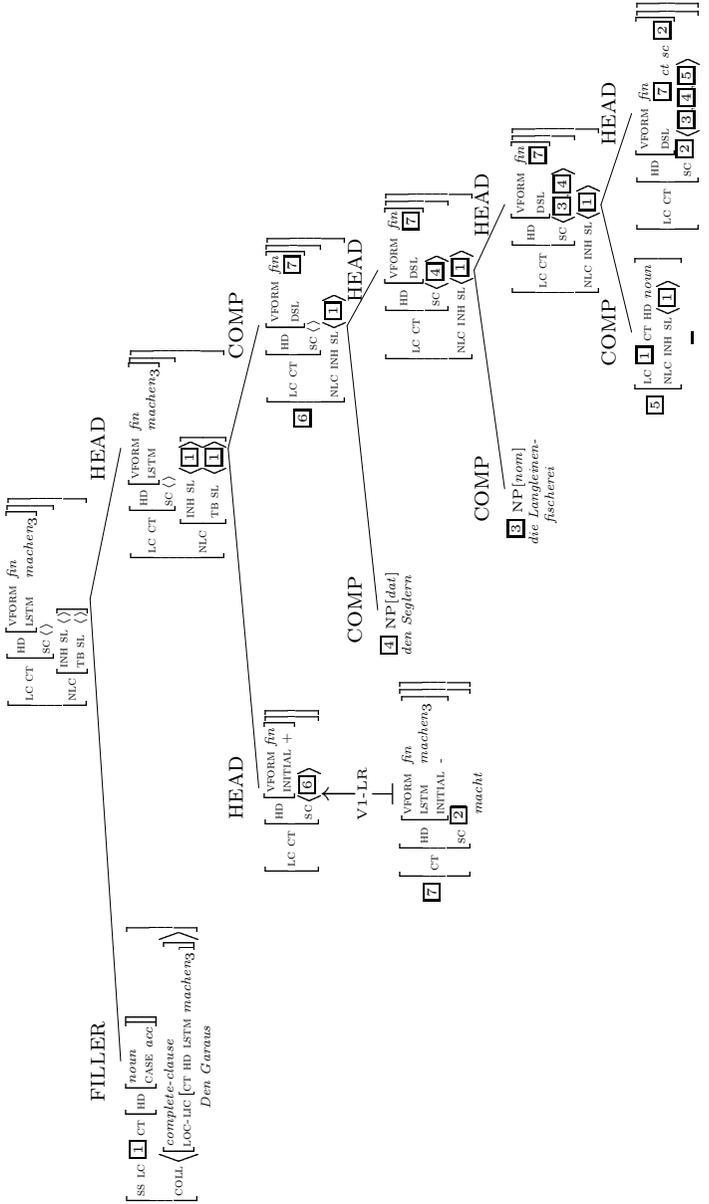


FIGURE 2 Analysis of *Den Garaus macht den Seglern die Langleinenfischerei*

the LISTEME value *machen*<sub>3</sub>. In the structure we have a node (S), which is the minimal barrier above the filler and whose head is exactly of the required LISTEME value. Thus, our LICENSING-PRINCIPLE is satisfied.

By the sort *machen*<sub>3</sub> we refer to an idiomatic verb *machen* with the meaning “to put”. This verb subcategorizes for an NP (with SYNSEM value [5]) bearing the LISTEME value *garaus* (not depicted in figure 2). As a LOCAL value includes LISTEME, and the LOCAL values of the filler and the gap are identical, the subcategorization requirements of *machen*<sub>3</sub> are satisfied, even though *Garaus* has been extracted. Note that by the identity of the LOCAL value [7] and the DSL value of [6], the verbal trace has the correct LISTEME value.

## 9.5 Alternative Analyses

### 9.5.1 A Different COLL Mechanism

The analysis we suggest here is an enhancement of a proposal by Richter and Sailer (1999). However, in Sailer (2003) the author described a variant of the COLL mechanism: In this thesis, the value of COLL is a singleton list that may contain a sign. That sign is the overall expression in which the idiomatic word occurs. Take for example the idiom *spill the beans*: in the lexical entry of the idiomatic word *beans* its COLL value is specified as a sign containing the semantic contributions of a definite article, the idiomatic word *spill* and *beans* itself in the right scopal relations. Sailer defines the so-called COLL-PRINCIPLE ensuring that the sign specified in a COLL list dominates the sign bearing that list. As a consequence, information of the overall utterance is available at lexical level and, conversely, local information is available on each node in the structure.

Thus, even though Sailer introduces only one new attribute, this approach is very unrestrictive and if one taps its full potential, nearly all grammatical phenomena can be described, even if they have nothing to do with collocations. Selection, e. g., would only be a special case of a collocation. Because of this power and unrestrictedness, that version of COLL is to be met with criticism.

### 9.5.2 A Constructional Approach

Riehemann (2001) makes another concrete proposal for the analysis of idioms. She adopts many ideas of Construction Grammar and carries them forward to the HPSG framework. Her approach re-

---

thus, on the one hand, to be general enough. On the other hand, we can exclude ungrammatical cases of extraction beyond clausal boundaries.

quires a complex machinery of new sorts and attributes to cover not only the amount of existing idioms but also their occurrences in different syntactic configurations. She has to assume, e. g., distinct subsorts of a *spill\_beans\_idiom\_phrase* for the idiom occurring in a *head-subject-phrase* (*Dana spilled the beans.*), in a *head-filler-structure* (*Who did Kim claim spilled the beans?*) or in a *head-specifier-structure* (*the beans that Dana spilled*). Even if the existence of sorts for different constructions is well established in Construction Grammar, it is questionable to assume different subclasses of linguistic signs, only because they contain idiomatic items in different syntactic structures. In other words, why assume different sorts for one single idiom only because it occurs in different constructions?

Moreover, Riehemann herself has to admit that her approach cannot handle cases of pronominal reference like (39), because idiomatic *spill* is not licensed as it seems to appear by itself and not within a *spill\_beans\_idiom\_phrase*.

In summary, it seems to us that a lexical approach is to be preferred over a structural one. Nevertheless, her arguments in favor of a constructional analysis of non-decomposable idioms are convincing. Our counterpart to that are phrasal lexical entries which we assume for this kind of idiomatic expressions.

## 9.6 A Modular Approach: Prospects

We have proposed one way of analyzing idioms and similar lexical idiosyncrasies. It can handle distributional characteristics of idiomatic words and even difficult cases like pronominalization. We have demonstrated by means of topicalization and verb movement that our proposal merges smoothly with other modules of grammar.

We decided to take a word-level collocation-based account using the COLL feature. This approach is modular in two ways. Firstly, the barriers can be adjusted “vertically” according to the range (XP, complete clause or utterance) needed for a particular idiomatic expression. Secondly, by the LOC-LIC feature we can specify any characteristics within the local information. We could now go on and define other attributes of *barrier* like PHON-LIC to define any requirements of the phonetic string of that barrier. In that way our approach is also horizontally modular.

An application of such a PHON-LIC feature would be the modelling of occurrence restrictions of the English indefinite article *an*. This phenomenon together with other cases of sandhi is discussed by Asudeh and Klein (2002). With our approach, we define the lexical entry of *an* as follows: the PHON-LIC value of the barrier *np* on the COLL list is the phonetic string ⟨*an*⟩ + a phonetically realized vowel.

Thus, with a quite general approach to idioms using the COLL feature, we can handle very particular phenomena, too. We leave it to further research to explore the possibilities that our approach offers.

## References

- Asudeh, Ash and Ewan Klein. 2002. Shape Conditions and Phonological Context. In F. van Eynde, L. Hellan, and D. Beermann, eds., *Proceedings of the 8th International HPSG Conference*, pages 20–30. CSLI Publications. <http://csli-publications.stanford.edu/HPSG/2/>.
- Di Sciullo, Anna-Maria and Edwin Williams. 1988. *On the Definition of Word*. Linguistic Inquiry Monographs. MIT Press, Cambridge, Mass, 2nd edn.
- Gazdar, Gerald, Ewan Klein, Geoffrey Pullum, and Ivan Sag. 1985. *Generalized Phrase Structure Grammar*. Cambridge, Mass.: Harvard University Press.
- Heinz, W. and J. Matiassek. 1994. Argument Structure and Case Assignment in German. In J. Nerbonne, K. Netter, and C. Pollard, eds., *German in Head-Driven Phrase Structure Grammar*, pages 199–236. CSLI Publications. Lecture Notes 46.
- Krenn, Brigitte and Gregor Erbach. 1994. Idioms and Support Verb Constructions. In J. Nerbonne, K. Netter, and C. Pollard, eds., *German in Head-Driven Phrase Structure Grammar*, pages 365–396. CSLI Publications. Lecture Notes 46.
- Müller, Stefan. 2004. Zur Analyse der scheinbar mehrfachen Vorfeldbesetzung. *To appear in Linguistische Berichte* URL: <http://www.cl.uni-bremen.de/~stefan/Pub/mehr-vf-lb.html>; 16.01.2004.
- Nunberg, Geoffrey, Ivan A. Sag, and Thomas Wasow. 1994. Idioms. *Language* 70:491–538.
- Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Stanford University: CSLI/The University of Chicago Press.
- Richter, Frank. 1997. Die Satzstruktur des Deutschen und die Behandlung langer Abhängigkeiten in einer Linearisierungsgrammatik. Formale Grundlagen und Implementierung in einem HPSG-Fragment. In E. Hinrichs, D. Meurers, F. Richter, M. Sailer, and H. Winhart, eds., *Ein HPSG-Fragment des Deutschen, Teil 1: Theorie*, no. 95 in Arbeitspapiere des SFB 340, pages 13–187. Universität Tübingen.

- Richter, Frank and Manfred Sailer. 1999. LF conditions on expressions of Ty2: An HPSG analysis of negative concord in Polish. In R. D. Borsley and A. Przepiórkowski, eds., *Slavic in Head-Driven Phrase Structure Grammar*, pages 247–282. Stanford: CSLI Publications.
- Riehemann, Susanne Z. 2001. *A Constructional Approach to Idioms and Word Formation*. Ph.D. thesis, Stanford University, Stanford, CA.
- Sailer, Manfred. 2003. Combinatorial Semantics and Idiomatic Expressions in Head-Driven Phrase Structure Grammar. Phil. Dissertation (2000). Arbeitspapiere des SFB 340. 161, Eberhard-Karls-Universität Tübingen.
- Soehn, Jan-Philipp and Manfred Sailer. 2003. At first blush on tenterhooks. about selectional restrictions imposed by nonheads. In G. Jäger, P. Monachesi, G. Penn, and S. Wintner, eds., *Proceedings of Formal Grammar 2003*, pages 149–161.



## 10

---

# Resumption in Persian Relative Clauses: An HPSG Analysis

MEHRAN TAGHVAIPOUR

### 10.1. Introduction

Persian is a null-subject verb final language with SOV word order that allows personal pronouns to be used resumptively in relative clause (RC) constructions. RCs in Persian are head-modifying constituents, all typically introduced by the invariant complementizer *ke*. Persian RCs are Unbounded Dependency Constructions (UDCs), containing either a gap or a resumptive pronoun (RP). The gap or the RP is linked to and licensed by the NP modified by the RC. In some positions only gaps are allowed, and in other positions only RPs. There are also some positions where both gaps and RPs are alternatively allowed. Illustrating the striking similarities between Persian gaps and RPs, I will provide an HPSG unified approach to take care of the long distance dependency between the licensing structure and the gap or the RP in Persian restrictive RCs with a truly single feature-based mechanism, using only the SLASH feature.

### 10.2 The Data

Example (1) shows a Persian sentence containing a RC. The RC is put in brackets.

*Proceedings of Formal Grammar 2004*

Gerhard Jäger, Paola Monachesi, Gerald Penn and Shuly Wintner (eds.).

Copyright © 2004, the individual authors

(1)

*mærd-i*<sup>1</sup> [ke *piræn-e* *zærd* *pu\_ideh*] *Dr. Bayat-eh*  
 man-RES COMP shirt-EZ yellow wear-PP-3sg Dr. Bayat is  
 ‘The man who is wearing a yellow shirt is Dr. Bayat.’

Example (2a) shows another Persian RC in which the gap is shown by \_\_\_\_\_. Example (2b) represents the same RC with a resumptive pronoun. The pronoun *u*, i.e. ‘he’, is used resumptively in (2b). Example (2c) shows the cliticized form of the pronoun ‘*u*’.

(2a)

*mærd-i* [ke *oma* \_\_\_\_\_ *diruz* *molaqat kærdid*]...  
 man-RES COMP you Ø yesterday meet-PAST-2pl ...  
 ‘The man whom you met yesterday...’

(2b)

*mærd-i* [ke *oma u* *ra*<sup>2</sup> *diruz* *molaqat kærdid*]...  
 man-RES COMP you **he** RA yesterday meet-PAST-2pl ...  
 ‘The man whom you met (**\*him**) yesterday...’

(2c)

*mærd-i* [ke *oma diruz* *molaqat-æ* *kærdid* ...]  
 man-RES COMP you yesterday meet-**him** do-PAST-2pl...  
 ‘The man whom you met (**\*him**) yesterday...’

It is not always possible to replace a gap with a RP. For instance, if we replace the gap in (1) above with a RP, the result will be example (3), which is ungrammatical.

(3)

*mærd-i* [ke *u* *piræn-e* *zærd* *pu\_ideh*] *Dr. Bayat-eh*  
 man-RES COMP **he** shirt-EZ yellow wear-PP-3sg Dr. Bayat is  
 ‘The man who (**\*he**) is wearing a yellow shirt is Dr. Bayat.’

The pattern of distribution of RPs and gaps in Persian RCs depends on two factors. The first factor is their position inside the RC and the second factor is whether the RC is restrictive or nonrestrictive.

<sup>1</sup> Particle *-i* (-RES in gloss) is a suffix that attaches to the nouns modified by restrictive RCs.

<sup>2</sup> This particle (whose colloquial form is *ro*) is a specificity marker in Persian and is shown, henceforth, by RA in gloss.

	Subject	Object of Prep.	Genitive	Direct Object
Gap is allowed?	Yes	No	No	Yes
RP is Allowed?	No	Yes	Yes	Yes

Table 1: Distribution of Gaps or Resumptive Pronouns in Persian restrictive RCs

Table 1 above shows the pattern of distribution of RPs and gaps in restrictive RCs in Persian. Table 2 below shows this pattern in nonrestrictive RCs in this language. A comparison between the two tables shows that it is not possible to use gaps or RPs alternatively in direct object positions in nonrestrictive RCs in Persian.

	Subject	Object of Prep.	Genitive	Direct Object
Gap is allowed?	Yes	No	No	No
RP is Allowed?	No	Yes	Yes	Yes

Table 2: Distribution of Gaps or Resumptive Pronouns in Persian restrictive RCs

While examples like (2) above showed the possibility of alternative options in restrictive RCs when the relativized position is direct object, examples like (4) below show lack of this possibility in non-restrictive RCs in this language. In non-restrictive RCs, RPs are obligatory if the the relativized position is direct object.

(4a)

*Omid, ke shoma u ra molaqat+kærdid, daee-ye mæn æst.*  
 Omid, COMP you **he** RA meet-PAST-2pl, uncle-EZ I is  
 ‘Omid, who(m) you met (**\*him**) yesterday, is my uncle.’

(4b)

*\*Omid, ke shoma \_\_\_ molaqat+kærdid, daee-ye mæn æst.*  
 Omid, COMP you \_\_\_ meet-PAST-2pl, uncle-EZ I is  
 ‘Omid, who(m) you met \_\_\_ yesterday, is my uncle.’

Persian Gaps and RPs show striking similarities. I will provide a variety of evidence in favour of this similarity to conclude that Persian RCs contain traces, rather than null constituent gaps.

A strong argument in support of the fundamental similarity of RPs and gaps are comes from coordinate structures. Example (5) shows that in Persian a RP can be used with a gap in coordinate structures in unbounded

dependencies. In fact, it is possible to have gaps in both conjuncts, RPs in both, or a gap in one conjunct and a RP in the other (in any order).

(5)

*mærd-i ke \_\_\_\_\_ pirahæn-e zærd pu\_ideh+bud væ*  
 man-RES COMP Ø shirt-EZ yellow wear-PRESPART-3sg and

*shoma diruz az u pul qærz+gereftid Ali bud.*  
 you yesterday from **him** money borrow-PAST-2pl Ali was

‘The man who \_\_\_\_\_ was wearing a yellow shirt and you borrowed money from (**\*him**) was Ali.’

The second argument that supports the similarity between Persian RPs and gaps comes from parasitic gaps. Persian data shows that RPs, like gaps, can license parasitic gaps. I will bring examples (6a) and (6b) to illustrate this possibility. In (6a) there are two gaps, the second of which is parasitic. (6b) shows a sentence in which the second gap is still parasitic but licensed by the RP *un*.

(6a)

*in ketab-i-ye ke Yasmin bedun in*  
 this book-RES-is COMP Yasmin without this

*ke \_\_\_\_\_ bexuneh \_\_\_\_\_ xærid.*  
 COMP Ø read-3sg Ø bought-3sg.

‘This is the book that Yasmin bought \_\_\_\_\_ without reading \_\_\_\_\_’

(6b)

*in ketab-i-ye ke Yasmin bedun in*  
 this book-RES-is COMP Yasmin without this

*ke un ro bexuneh \_\_\_\_\_ xærid.*  
 COMP **it** RA read-3sg Ø bought-3sg.

‘This is the book that Yasmin bought (**\*it**) without reading \_\_\_\_\_’

Another piece of supporting evidence for the similarity of Persian gaps and RPs is the sensitivity of RPs, like gaps, to certain islands. This is unlike what we see in Hebrew, for instance (see Vaillette (2001)). As an example, Persian gaps are sensitive to Subject Condition as shown in (7).

(7a)

[*in ede'a ke Ali Hæmid ra dideh*]  
 this claim COMP Ali Hamid RA seen

*Yasmin ra narahat+kærd.*  
 Yasmin RA annoyed

‘The claim that Ali has seen Hamid annoyed Yasmin.’

(7b)

\**mærd-i ra ke [in ede'a ke Ali \_\_\_/u ra dideh]*  
 man-RES RA COMP [this claim COMP Ali \_\_\_/him RA seen]

*Yasmin ra narahat+kærd.*  
 Yasmin RA annoyed.

‘The man that the claim that Ali has seen \_\_\_/him annoyed Yasmin.’

Thus, Persian gaps and RPs are strikingly similar: they have the same status within conjuncts, they can both license parasitic gaps; and, they are both sensitive to some island constraints. Based on this similarity, I will propose that they are both signs associated with the SLASH feature.

## 10.3 An HPSG Analysis

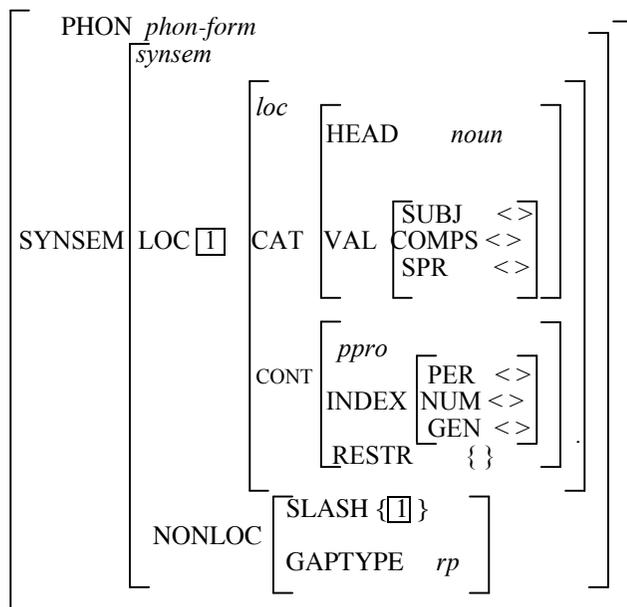
### 10.3.1 Bottom

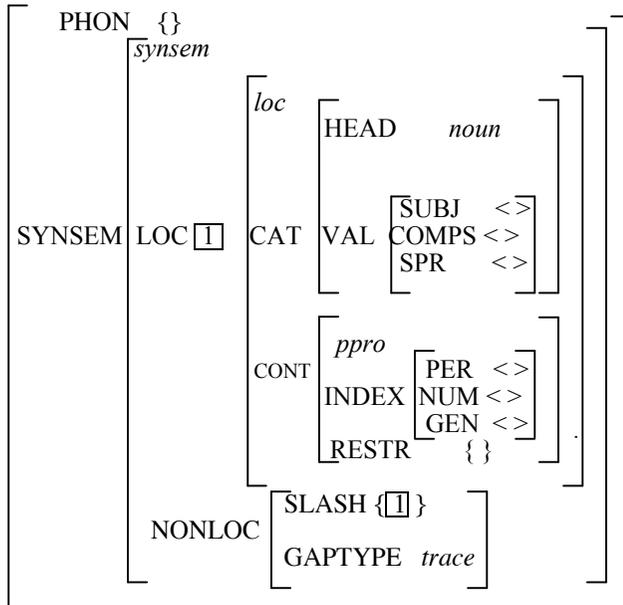
I will assume that the unbounded dependency in Persian RCs appear at the bottom of the dependency by a special sign that has a nonempty value for the SLASH feature. This special sign is either a trace or a RP. The nonempty SLASH feature encodes the information that there is a dependency between the trace/RP and the NP modified by the RC.

I will propose the lexical entry in (8) for RPs and the one in (9) for traces. The lexical entries in (8) and (9) are the same except in two respects. Firstly, the value of the PHON feature in traces is an empty set. This means that RPs as overt elements have phonology but traces do not. The second

difference between these two lexical entries is that the value of their GAPTYPE features is different.

(8) Lexical Entry for a **resumptive pronoun**



(7) Lexical Entry for a **trace**

GAPTYPE is a feature that I have introduced in order to capture the distributional properties of RPs and traces. GAPTYPE is a non-local feature whose value can be either *trace* or *rp*, for traces and RPs, respectively. The reason for distinguishing traces and RPs with a NONLOCAL feature is that this is not reflected within the value of SLASH and hence it is possible for a single unbounded dependency to be associated with a trace and an RP.

As for the pattern of distribution of RPs and traces, I will, first prevent RPs from appearing in subject position. I will propose the constraint in (10) to deal with this.

(10)  
 $[SUBJ < [1] >] \rightarrow \sim ([1] = [SYNSEM|NONLOC|GAPTYPE \quad rp])$

The effect of this constraint is that if an element is in subject position, then the value of its GAPTYPE feature cannot be *rp*. In other words, if an element is a RP whose value of the GAPTYPE feature is *rp*, then it cannot come in subject position.

The second constraint, I will propose here, is to prevent traces from appearing in the positions of object of prepositions and possessors (i.e., in positions of the complements of non-verbs). This constraint is proposed in (11) below.

(11)

$$\left[ \begin{array}{l} \text{HEAD} \quad [1] \\ \text{COMPS} \langle \dots, [\text{GAPTYPE } \textit{trace}], \dots \rangle \end{array} \right] \rightarrow [1] = \textit{verb}$$

The effect of (11) is that if there is a trace as a complement of a head, then that head has to be a verb. Therefore, as in the case of object of preposition and genitive cases (possessors), the head is not a verb, we will not have a trace therein.

### 10.3.2 Middle

In the middle of the dependency, I will follow Sag (1997). The SLASH is inherited by two constraints: Lexical Amalgamation of SLASH, and SLASH Inheritance Principle, given in (12) and (13) below.

(12) Lexical Amalgamation of SLASH

$$\textit{word} \Rightarrow \left[ \begin{array}{l} \text{BIND} \quad \boxed{0} \\ \text{ARG-ST} \langle [\text{SLASH} \boxed{1}], \dots, [\text{SLASH} \boxed{n}] \rangle \\ \text{SLASH} (\boxed{1} + \dots + \boxed{n}) - \boxed{0} \end{array} \right]$$

According to (12), all words, except SLASH binding elements like *tough*, specify empty value for the feature BIND. That is, in most cases nothing is subtracted from the disjoint union of the argument's SLASH values. Therefore, if a non-head-daughter is slashed so should the head daughter.

(13) SLASH Inheritance Principle (SLIP):

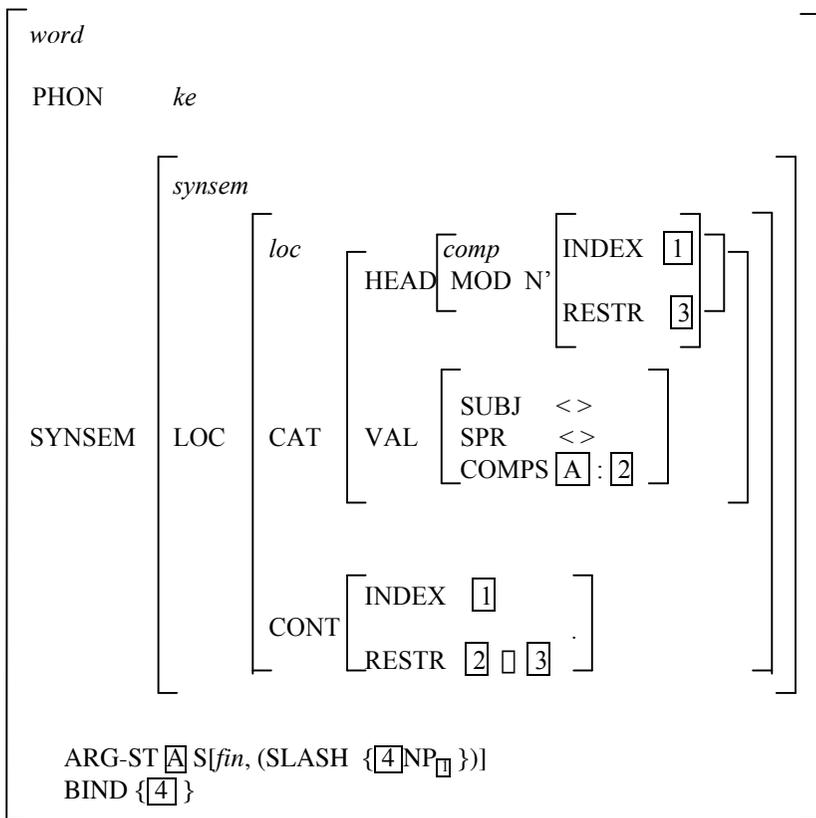
$$\textit{hd-nexus-ph} \Rightarrow \left[ \begin{array}{l} \text{SLASH} / \boxed{1} \\ \text{HD-DTR} / [\text{SLASH} \boxed{1}] \end{array} \right]$$

The constraint in (13) guarantees that the SLASH value of a phrase (of the type head-nexus-phrase) is the SLASH value of its head-daughter. In this way, any SLASH inheritance is mediated by the head-daughter, whose SLASH value contains that of the relevant non-head daughter.

### 10.3.3 Top

At the top of the dependency, I will need some way to bind the SLASH feature. In other words, I will need a way to ensure that the non-empty SLASH value stops at an appropriate point. This appropriate point, in Persian RCs, is the complementizer *ke*. I will propose the lexical entry in (14) for *ke* in RCs (i.e.,  $ke_{RC}$ ).

(14) Lexical Entry for  $ke_{RC}$



The lexical entry for *ke* specifies some lexical information that ensures that the index of the N' (the NP modified by the RC) is identical to the SLASH value of *ke*. This structure-sharing, which is shown by tag  $\square$ , relates the trace or the RP to the NP modified by the RC. In addition, (12) also ensures that *ke* requires a sentential complement, shown by tag  $\square_A$ . Tag  $\square_A$  is the only member of *ke*'s ARG-ST list that stands for a finite sentence, containing a trace or a RP. The lexical binding of SLASH is accounted for by the feature BIND, which has a non-empty set as value for *ke*. This is shown by tag  $\square_4$ . The BIND feature will ensure that the trace or the RP is not amalgamated into the SLASH value of *ke* itself.

## 10.4 The Open Issue

In Section 1, I noted that the pattern of distribution of resumptive pronouns in non-restrictive relative clauses is different. That is, while the resumptive pronoun or gap can be used alternatively in restrictive RCs (as shown in (2) above), the two cannot substitute one another in non-restrictive counterparts (as shown in (4) above).

My account for RPs at its present state cannot provide any analysis for non-restrictive clauses.

## References

- Pollard C., and I. Sag, 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, USA.
- Sag, I., 1997. English Relative Clause Constructions. *Journal of Linguistics* 33:431-484.
- Vaillette, N. 2001. Hebrew Relative Clauses in HPSG. *Proceedings of the 7th International Conference on Head-Driven Phrase Structure Grammar*, CSLI Publications.

# A Hierarchy of Mildly Context-Sensitive Dependency Grammars

ANSSI YLI-JYRÄ AND MATTI NYKÄNEN

*Dept. of General Linguistics and Dept. of Computer Science,  
University of Helsinki, Finland*

*Email: Anssi.Yli-Jyra@ling.helsinki.fi, Matti.Nykanen@cs.helsinki.fi*

**ABSTRACT.** The paper presents Colored Multiplanar Link Grammars (CMLG). These grammars are reducible to extended right-linear  $S$ -grammars (Wartena 2001) where the storage type  $S$  is a concatenation of  $c$  pushdowns. The number of colors available in these grammars induces a hierarchy of Classes of CMLGs. By fixing also another parameter in CMLGs, namely the bound  $t$  for *non-projectivity depth*, we get  $c$ -Colored  $t$ -Non-projective Dependency Grammars (CNDG) that generate acyclic dependency graphs. Thus, CNDGs form a two-dimensional hierarchy of dependency grammars. A part of this hierarchy is mildly context-sensitive and non-projective.

## 11.1 Introduction

This paper proposes non-projective, polynomially parseable lexicalized grammars capable of describing scrambling and long-distance dependencies up to a bounded *nested crossing depth* and a bounded *non-projectivity depth*.

In terms of dependency grammar (DG) (Tesnière 1959), word-order is distinct from the dependency tree that analyzes the structure of the sentence. However, Hays (1964) and Gaifman (1965) have formalized Tesnière's ideas so that their DGs describe only *projective* linearisations.

What we would like to have is a mildly context-sensitive (MCS) (Joshi 1985) superclass of the Hays-Gaifman DGs that captures also

*Proceedings of Formal Grammar 2004.*

Gerhard Jäger, Paola Monachesi, Gerald Penn, and Shuly Winter (eds.).

Copyright © 2004, the individual authors.

non-projective dependencies, e.g. scrambling, the possibility of the elements of a sentence to lie in arbitrary permutations. However, it seems that grammars that capture unrestricted scrambling fail to be mildly context-sensitive, *cf.* Global Index Grammar (GIG) (Castaño 2003). Limited scrambling is, however, captured by linear context-free rewriting systems (LCFRSs) (Vijay-Shanker et al. 1987) that are currently the best characterisation for MCS grammars.

Linear Indexed Grammar (LIG) (Gazdar 1988) is a LCFRS that represents nested non-local dependencies through an *index pushdown* that is associated with nodes in derivation trees. The additional power of some other LCFRSs is based on replacing the index pushdowns with *index storages* of a more general type  $\mathcal{S}$ . We will base our investigations on extended right-linear  $\mathcal{S}_{\text{pd}}^c$ -grammars (ERL- $\mathcal{S}$ -Gs) Wartena (2001) whose storage type consists of  $c$  pushdowns.

In this paper, some new restrictions on ERL- $\mathcal{S}_{\text{pd}}^k$ -Gs are developed. Through these restrictions we obtain various classes of DGs. The obtained DGs can be used to describe restricted non-projective dependencies and restricted scrambling, and they contain the Hays-Gaifman DGs as a subclass.

The important contribution of this paper is to show that when we set bounds for *nested crossing depth* and *non-projectivity depth*, we obtain classes of DGs that are mildly context-sensitive. The length of a longest chain ( $\frown \dots \smile$ ) of crossing dependencies constitute a lower bound for the *nested crossing depth* that is defined, in this paper, as the number of concatenated pushdowns  $c$  needed in derivation. By the *non-projectivity depth*, we mean the number of times dependency links climb from a projective position to a non-projective one along a path of directed dependencies.

The paper is structured as follows. Section 11.2 defines Context-Free Linear  $\mathcal{S}_{\text{pd}}^{c,\Gamma}$ -Grammars with Extended Domain of Locality. Section 11.3 eliminates ambiguity that is related to the storage allocation. In Section 11.4, we introduce *Colored Multiplanar Link Grammars* (CMLG), and discuss their properties in Section 11.5. In Section 11.6 we enforce a sufficient condition for acyclicity in  $c$ -Colored  $t$ -Non-projective Dependency Grammars (CNDG) that form a sub-hierarchy among CMLGs. The conclusion is in Section 11.8.

## 11.2 The Basic Machinery

### 11.2.1 Storage Type

**Definition 7** A *storage type* is a tuple  $\mathcal{S} = (C, C_i, C_f, \Phi, \Pi, m)$ , where

- $C$  is the set of *configurations*, and  $C_i, C_f \subseteq C$  are respectively the

- sets of *initial* and *final* configurations,
- $\Phi$  and  $\Pi$  are respectively the sets of *instructions* and *predicates*,
  - $m$  is the *meaning function*. It associates to each  $\pi \in \Pi$  the corresponding function  $m(\pi): C \rightarrow \{\text{TRUE}, \text{FALSE}\}$ , and to each  $\phi \in \Phi$  the corresponding partial function  $m(\phi): C \rightarrow C$ .

The meaning function  $m$  is extended to Boolean combinations of the predicates  $\Pi$  in the natural way and to nonempty strings  $\phi = (\Phi \cup (\mathcal{B}^\Pi \times \Phi^+))^+$  and pairs  $(\pi, \phi) \in (\mathcal{B}^\Pi \times \Phi^+)$  by defining  $m(\phi_1\phi_2)(\kappa) = m(\phi_2)(m(\phi_1)(\kappa))$ , where  $\kappa \in C$  and by defining

$$m((\pi, \phi)(\kappa) = \begin{cases} m(\phi)(\kappa), & \text{if } m(\pi)(\kappa) = \text{TRUE}, \\ \kappa, & \text{otherwise.} \end{cases}$$

Wartena (2001) defines a trivial (memoryless) storage  $\mathcal{S}_{\text{triv}}$ , an ordinary pushdown  $\mathcal{S}_{\text{pd}}$  and concatenations on storage types. The concatenation w.r.t. writing is denoted as  $\circ_w$ .

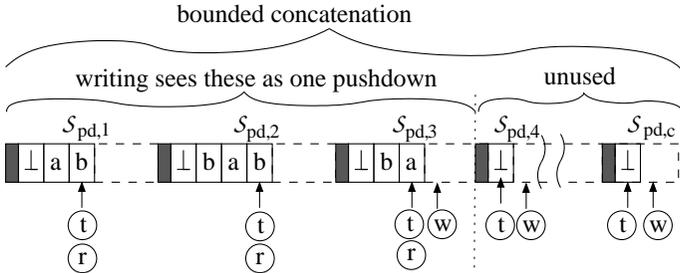


FIGURE 1 Example of a possible configuration of a storage  $(\dots((\mathcal{S}_{\text{triv}} \circ_w \mathcal{S}_{\text{pd},1}) \circ_w \mathcal{S}_{\text{pd},2}) \circ_w \mathcal{S}_{\text{pd},3}) \dots) \circ_w \mathcal{S}_{\text{pd},c}$  with marking of possibilities of applying operations TOP (t), POP (r) and PUSH (w).

### 11.2.2 Concatenating Storage Type

We restrict our attention to storages of type  $(\dots((\mathcal{S}_{\text{triv}} \circ_w \mathcal{S}_{\text{pd},1}) \circ_w \mathcal{S}_{\text{pd},2}) \circ_w \mathcal{S}_{\text{pd},3}) \dots) \circ_w \mathcal{S}_{\text{pd},c}$  that is intuitively a tuple  $\langle \mathcal{S}_{\text{pd},1}, \mathcal{S}_{\text{pd},2}, \mathcal{S}_{\text{pd},3}, \dots, \mathcal{S}_{\text{pd},c} \rangle$  of  $c$  pushdowns  $\mathcal{S}_{\text{pd},p}$  (Figure 1) with the restriction that writing new elements into pushdown  $\mathcal{S}_{\text{pd},p}$  is permitted only if all the succeeding pushdowns  $\mathcal{S}_{\text{pd},p+1}, \mathcal{S}_{\text{pd},p+2}, \mathcal{S}_{\text{pd},p+3}, \dots, \mathcal{S}_{\text{pd},c}$  are empty. Otherwise each pushdown  $\mathcal{S}_{\text{pd},p}$  can be used independently of the others. More formally, this storage type is defined as follows:

**Definition 8** A *writing-concatenating tuple of  $c$  pushdowns over a stack alphabet  $\Gamma$*  is the following storage type  $\mathcal{S}_{\text{pd}}^{c,\Gamma} = (C, C_i, C_f, \Phi, \Pi, m)$ :

- The configurations are  $C = ((\Gamma \cup \{\#, b, \natural\})^* \perp)^c$ , where  $\perp \notin \Gamma$  is a special symbol denoting the bottom of the pushdown, and  $\#, b, \natural \notin \Gamma$  are special *semaphore* symbols reserved for the restriction of normalized  $\mathcal{S}_{\text{pd}}^{c, \Gamma}$  that is introduced in Section 11.3,
- the unique initial and final configuration is  $C_i = C_f = \{\perp\}^c$ ,
- the predicates are  $\Pi = \{\text{TOP}_p(a) \mid 1 \leq p \leq c, a \in \Gamma \cup \{\perp, \#, b, \natural\}\}$ ,
- the instructions are  $\Phi = \{\text{ID}, \text{UNDEF}\} \cup \{\text{PUSH}_p(\beta) \mid 1 \leq p \leq c, \beta \in (\Gamma \cup \{\#, b, \natural\})^+\} \cup \{\text{POP}_p(\beta) \mid 1 \leq p \leq c, \beta \in (\Gamma \cup \{\#, b, \natural\})^+\}$ .

The predicates  $\Pi$  and instructions  $\Phi$  have the following basic meanings:

$$\begin{aligned} m(\text{ID})(\langle \alpha_1, \dots, \alpha_c \rangle) &= \langle \alpha_1, \dots, \alpha_c \rangle \\ m(\text{TOP}_p(a))(\langle \alpha_1, \dots, \alpha_{p-1}, \beta b, \alpha_{p+1}, \dots, \alpha_c \rangle) &= (a = b) \\ m(\text{PUSH}_p(\beta))(\langle \alpha_1, \dots, \alpha_p, \perp, \dots, \perp \rangle) &= \langle \alpha_1, \dots, \alpha_{p-1}, \alpha_p \beta, \perp, \dots, \perp \rangle, \\ m(\text{POP}_p(\beta))(\langle \alpha_1, \dots, \alpha_{p-1}, \alpha_p \beta^r, \alpha_{p+1}, \dots, \alpha_c \rangle) &= \langle \alpha_1, \dots, \alpha_c \rangle \end{aligned}$$

where  $\beta^r$  is the reverse of any string  $\beta \in (\Gamma \cup \{\#, b, \natural\})^+$  and the functions  $m(\phi) : C \rightarrow C$ ,  $\phi \in \Phi$ , remain undefined for all other cases.

### 11.2.3 Context-free Linear- $\mathcal{S}$ -Grammars

**Definition 9** Let  $\mathcal{S} = (C, C_i, C_f, \Phi, \Pi, m)$  be a storage type. A *context-free linear  $\mathcal{S}$ -grammar with extended domain of locality (CFL-EDL- $\mathcal{S}$ -G)* is a tuple  $G = (V_N, V_T, P, S, \kappa_0)$ , where

- the pairwise disjoint finite sets  $V_N$  and  $V_T$  are the *nonterminal* and *terminal alphabets*, respectively,
- $S \in V_N$  is the *start symbol*, and  $\kappa_0 \in C_i$  is the *start configuration*, and
- $P$  is a finite set of *productions* of the form

$$X\phi_1 \rightarrow \text{if } \pi \text{ then } \zeta_1 Y\phi_2 \zeta_2 \quad (11.9)$$

$$X \rightarrow \text{if } \pi \text{ then } w \quad (11.10)$$

where  $X, Y \in V_N$ ,  $\phi_1 \in (\mathcal{B}^\Pi \times \Phi^+)^+$ ,  $\phi_2 \in \Phi^+$ ,  $\pi \in \mathcal{B}^\Pi$ ,  $\zeta_1, \zeta_2 \in (V_N \cup V_T)^*$ , and  $w \in V_T^*$ .

The set  $\mathbf{S} = ((V_N \times C) \cup V_T)^*$  is called the set of *sentential forms*, and  $\sigma \in \mathbf{S}$  is said to *derive*  $\tau \in \mathbf{S}$  if and only if  $\sigma = \alpha(X, \kappa)\beta$  and  $\tau = \alpha\gamma\beta$  for some  $\alpha, \beta, \gamma \in \mathbf{S}$  and  $P$  contains either

- a production of the type (11.9) for which  $m(\phi_1(\neg\pi, \text{UNDEF})\phi_2)(\kappa)$  is defined and  $\gamma = \zeta'_1(Y, m(\phi_1\phi_2)(\kappa))\zeta'_2$ , where  $\zeta'_1$  and  $\zeta'_2$  are obtained from  $\zeta_1$  and  $\zeta_2$  respectively by replacing every nonterminal  $D$  by  $(D, \kappa_0)$ , or,
- a production of the type (11.10) for which  $m(\pi)(\kappa) = \text{TRUE}$  and  $\gamma = w$ .

The initial sentential form is  $\langle(S, \kappa_0)\rangle$ . The *derivations* and the *generated language* of grammar  $G$  are defined in a usual way.

**Definition 10** A CFL-EDL- $\mathcal{S}$ -G is a *right-linear  $\mathcal{S}$ -grammar* with extended domain of locality (RL-EDL- $\mathcal{S}$ -G), if its productions of the form (11.9) are such that  $\zeta_1 \in V_T^*$  and  $\zeta_2 \in \epsilon$ .

A *context-free linear  $\mathcal{S}$ -grammar* (CFL- $\mathcal{S}$ -G) (Weir 1994) is a CFL-EDL- $\mathcal{S}$ -grammar, whose productions of the form (11.9) are such that  $\phi_1 = \epsilon$  and  $\phi_2 \in \Phi$ .

A *right linear  $\mathcal{S}$ -grammar* (RL- $\mathcal{S}$ -G) is an RL-EDL- $\mathcal{S}$ -G, whose productions of the form (11.9) are such that  $\phi_1 = \epsilon$  and  $\phi_2 \in \Phi$ .

An *extended right-linear  $\mathcal{S}$ -grammar* (ERL- $\mathcal{S}$ -G) (Wartena 2001) is a CFL- $\mathcal{S}$ -G, whose productions of the form (11.9) are such that  $\zeta_1 \in V_N$  and  $\zeta_2 \in V_T \cup \{\epsilon\}$ .

**Theorem 6** *CFL-EDL- $\mathcal{S}$ -Gs and CFL- $\mathcal{S}$ -Gs generate the same languages, and RL-EDL- $\mathcal{S}$ -Gs and RL- $\mathcal{S}$ -Gs generate the same languages.*

*Proof.* The inclusions  $\mathcal{L}(\text{CFL-}\mathcal{S}\text{-G}) \subseteq \mathcal{L}(\text{CFL-EDL-}\mathcal{S}\text{-G})$  and  $\mathcal{L}(\text{RL-EDL-}\mathcal{S}\text{-G}) \subseteq \mathcal{L}(\text{RL-}\mathcal{S}\text{-G})$  follows from the definition of the grammars. To show that the reverse inclusions hold, we replace productions of the form (11.9) by expanding them syntactically into

$$X \rightarrow \text{if TRUE then } \zeta_1 Q_{\phi_1 \pi \phi_2}^Y \zeta_2$$

and define the productions for new nonterminals  $Q_\omega^Y$  inductively as

$$\begin{aligned} Q_\epsilon^Y &\rightarrow \text{if TRUE then } Y \text{ ID} \\ Q_{\langle \pi', \phi' \rangle \omega}^Y &\rightarrow \text{if } \pi' \text{ then } Q_{\phi' \omega}^Y \text{ ID} \\ Q_{\langle \pi', \phi' \rangle \omega}^Y &\rightarrow \text{if } \neg \pi' \text{ then } Q_\omega^Y \text{ ID} \\ Q_{\pi' \omega}^Y &\rightarrow \text{if } \pi' \text{ then } Q_\omega^Y \text{ ID} \\ Q_{\phi' \omega}^Y &\rightarrow \text{if TRUE then } Q_\omega^Y \phi' \end{aligned}$$

where  $\pi' \in \mathcal{B}^\Pi$ ,  $\phi' \in \Phi^+$ , and where  $\omega$  denotes suffixes of the three-part string  $\phi_1 \pi \phi_2$ . All the productions of the form (11.9) in the expanded grammar contain only productions where  $\phi_1 = \epsilon$  and  $\phi_2 \in \Phi$ . The expanded grammar recognizes the language of the original grammar.  $\square$

Note that a derivation step of the original CFL-EDL- $\mathcal{S}$ -G may corresponds to multiple steps in the resulting CFL- $\mathcal{S}$ -G.

**Theorem 7** *Every RL- $\mathcal{S}$ -G can be reduced to an ERL- $\mathcal{S}$ -G.*

*Proof.* A new nonterminal and a new production are created for each non-empty prefix of  $\zeta_1 \in V_T^+$ . This allows replacing  $\zeta_1 \in V_T^+$  in the

original productions with  $\zeta'_1 \in V'_N$  in the productions of the ERL- $\mathcal{S}$ -G.  $\square$

The following two properties of ERL- $\mathcal{S}$ -Gs are needed in Section 11.5:

**Proposition 8** *ERL- $\mathcal{S}$ -G is a linear context-free rewriting system. Thus, it is polynomially parseable and has linear growth property.*

### 11.3 Normalized Storage

If the number of pushdowns is larger than the number of nested crossing dependencies in derivations, CFL-EDL- $\mathcal{S}_{pd}^{c,\Gamma}$ -Gs have some freedom in allocation of different pushdowns for different stack symbols. An example of this is shown in Figure 2. Some ways to allocate pushdowns

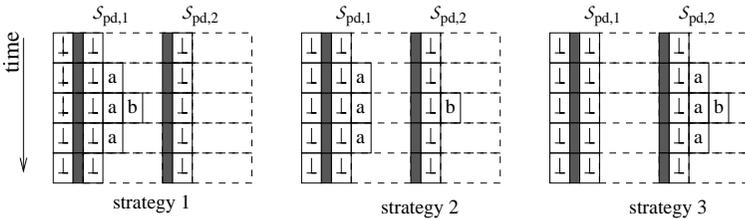


FIGURE 2 An example of ambiguity in storage allocation.

have already been banned by the the following restrictions that are imposed by the concatenating storage type  $\mathcal{S}_{pd}^{c,\Gamma}$ :

1. the LIFO discipline applies separately to each pushdown, and
2. the pushdowns are concatenated with respect to writing.

In addition to the effect of these two restrictions, we want to eliminate strategies 2 and 3 shown in Figure 2. For this purpose, we propose the following restrictions that eliminate this kind of ambiguity in allocation of pushdowns:

3. An operation that writes to an empty pushdown  $\mathcal{S}_{pd,p}$ ,  $p \geq 2$ , is allowed only if  $\mathcal{S}_{pd,p-1}$  is nonempty.
4. An operation that writes to an empty pushdown  $\mathcal{S}_{pd,p}$ ,  $p \geq 2$ , is allowed only if the current configuration contains a stack symbol in  $\mathcal{S}_{pd,p-1}$  that will be read before  $\mathcal{S}_{pd,p}$  becomes empty again.

Let  $\Phi'$  be an extended set of instructions on the normalized storage type  $\text{Norm}\mathcal{S}_{pd}^{c,\Gamma}$ . It is the union of  $\Phi$  and  $\{\text{RPUSH}_p(\beta) \mid 1 \leq p \leq c, \beta \in \Gamma^+\} \cup \{\text{RPOP}_p(a) \mid 1 \leq p \leq c, a \in \Gamma\}$ , where the new RPUSH and RPOP instructions obey the constraints 3 and 4, while the old PUSH and POP instructions do not.

The meanings of the new instructions are defined by means of semaphore symbols: when an  $\text{RPUSH}_p$  instruction to an empty pushdown  $p$ ,  $p > 1$ , takes place, a semaphore symbol  $\flat$  is written on the bottom of  $\mathcal{S}_{\text{pd},p}$  and two semaphore symbols  $\sharp$  and  $\natural$  are written respectively on the top of pushdowns  $\mathcal{S}_{\text{pd},p-1}$  and  $\mathcal{S}_{\text{pd},p}$ . The symbol  $\natural$  is kept always on the top of  $\mathcal{S}_{\text{pd},p}$ . Later, when a normal symbol is being read from the pushdown  $\mathcal{S}_{\text{pd},p-1}$  with  $\text{RPOP}$ , these semaphores  $\sharp$  and  $\natural$  are first removed respectively from  $\mathcal{S}_{\text{pd},p-1}$  and  $\mathcal{S}_{\text{pd},p}$  if they have not yet been removed. When the semaphore symbol  $\flat$  is read from  $\mathcal{S}_{\text{pd},p}$ , this must happen immediately after reading a non-semaphore symbol (rather than  $\natural$ )  $\mathcal{S}_{\text{pd},p}$ . If the semaphores cannot be read in this way, there are no other ways to read them. Thus, the derivation will be stuck in the cases where the restrictions 3 and 4 cannot be satisfied. More formally, the meaning functions are:

$$\begin{aligned}
m(\text{RPUSH}_p(\beta))(\kappa) &= \\
&\begin{cases} m(\text{PUSH}_{p-1}(\sharp) \text{PUSH}_p(b\beta\natural))(\kappa), & \text{if } p \geq 2 \vee \text{TOP}_p(\perp)(\kappa); \\ m(\text{POP}_p(\natural) \text{PUSH}_p(\beta\natural))(\kappa), & \text{if } p \geq 2 \vee \text{TOP}_p(\natural)(\kappa); \\ m(\text{PUSH}_p(\beta))(\kappa), & \text{otherwise,} \end{cases} \\
\text{and } m(\text{RPOP}_p(a))(\kappa) &= \\
&\begin{cases} m(\text{POP}_p(a))(\kappa), & \text{if } p = 1; \\ m((\text{TOP}_p(\sharp) \wedge \text{TOP}_{p+1}(\natural), \text{POP}_p(\sharp) \text{POP}_{p+1}(\natural)) \\ (\text{TOP}_p(\natural), \text{POP}_p(\natural a) \text{PUSH}_p(\natural)) (\neg \text{TOP}_p(\natural), \text{POP}_p(a)) \\ (\text{TOP}_p(b), \text{POP}_p(b)))(\kappa), & \text{otherwise.} \end{cases}
\end{aligned}$$

**Theorem 9** *CFL-EDL- $\mathcal{S}_{\text{pd}}^{c,\Gamma}$ -Gs using  $\text{RPUSH}$  and  $\text{RPOP}$  instructions can be reduced to CFL-EDL- $\mathcal{S}_{\text{pd}}^{c,\Gamma}$ -Gs that don't use these instructions.*

*Proof.* These new instructions can be regarded as shorthand notations which extend the transformation in Theorem 6 as follows:

Nonterminals  $Q_{\text{RPUSH}_1(\beta)\omega}^Y$ ,  $Q_{\text{RPOP}_1(\beta)\omega}^Y$  and  $Q_{\text{RPOP}_p(\beta)\omega}^Y$  where  $p \geq 2$  are replaced respectively with nonterminals  $Q_{\text{PUSH}_1(\beta)\omega}^Y$ ,  $Q_{\text{POP}_1(\beta)\omega}^Y$  and  $Q_{(\text{TOP}_p(\sharp) \wedge \text{TOP}_{p+1}(\natural), \text{POP}_p(\sharp) \text{POP}_{p+1}(\natural)) (\text{TOP}_p(\natural), \text{POP}_p(\natural a) \text{PUSH}_p(\natural)) (\neg \text{TOP}_p(\natural), \text{POP}_p(a)) (\text{TOP}_p(b), \text{POP}_p(b))}^Y$ . Nonterminals  $Q_{\text{RPUSH}_p(\beta)\omega}^Y$ , where  $p \geq 2$ , create the grammar rules

$$\begin{aligned}
Q_{\text{RPUSH}_p(\beta)\omega}^Y &\rightarrow \text{if } \text{TOP}_p(\perp) \text{ then } Q_{\omega}^Y \text{PUSH}_{p-1}(\sharp) \text{PUSH}_p(b\beta\natural) \\
Q_{\text{RPUSH}_p(\beta)\omega}^Y &\rightarrow \text{if } \text{TOP}_p(\natural) \text{ then } Q_{\omega}^Y \text{POP}_p(\natural) \text{PUSH}_p(\beta\natural) \\
Q_{\text{RPUSH}_p(\beta)\omega}^Y &\rightarrow \text{if } \neg \text{TOP}_p(\perp) \wedge \neg \text{TOP}_p(\natural) \text{ then } Q_{\omega}^Y \text{PUSH}_p(\beta). \quad \square
\end{aligned}$$

**Definition 11** A *Context-free linear normalized*  $\mathcal{S}_{\text{pd}}^{c,\Gamma}$  *grammar with extended domain of locality* (CFL-EDL-Norm $\mathcal{S}_{\text{pd}}^{c,\Gamma}$ -G) is a CFL-EDL- $\mathcal{S}_{\text{pd}}^{c,\Gamma}$ -grammar whose rules do not directly use PUSH and POP instructions, but use RPUSH and RPOP instead.

**Theorem 10** CFL-EDL-Norm $\mathcal{S}_{\text{pd}}^{c,\Gamma}$ -Gs allocate pushdowns for different stack symbols so that they conform the restrictions 1 - 4.

*Proof.* The proof is omitted for brevity. □

Dependencies are binary relations between string positions of the generated string. They are described by symbols that are written to a pushdown at one string position and read from that pushdown at another position. The reason for having multiple pushdowns in the storage Norm $\mathcal{S}_{\text{pd}}^{c,\Gamma}$  is to enable capturing *nested crossing dependencies*. For any set of dependency links whose starting and finishing times are disjoint, there is only one way to allocate Norm $\mathcal{S}_{\text{pd}}^{c,\Gamma}$  for these links. Yli-Jyrä (2003) has experimented with a data structure equivalent to Norm $\mathcal{S}_{\text{pd}}^{c,\Gamma}$  and shown that dependency trees of a small dependency tree-bank can be represented as a sequence of configurations of such a data structure. This motivates introduction of the grammar formalisms of Sections 11.4 and 11.6.

### 11.4 Colored Multiplanar Link Grammar (CMLG)

**Definition 12** A *nonterminal-free Norm $\mathcal{S}_{\text{pd}}^{c,\Gamma}$ -grammar* is a *RL - EDL* Norm $\mathcal{S}_{\text{pd}}^{c,\Gamma}$  - G  $G = (V_N, V_T, P, S, \kappa_0)$  for which  $V_N = \{S, X\}$  and whose productions are of the following forms:

$$S \quad \rightarrow \text{if } \text{TRUE} \quad \text{then } X\phi_{2,1} \quad (11.11)$$

$$X\phi_{1,r} \rightarrow \text{if } \bigwedge_{1 \leq p \leq c} \text{TOP}_p(\perp) \quad \text{then } \epsilon \quad (11.12)$$

$$X\phi_{1,r} \rightarrow \text{if } \pi \quad \text{then } aX\phi_{2,1} \quad (11.13)$$

$$X\phi_{1,r} \rightarrow \text{if } \pi \quad \text{then } X\phi_{2,r+1} \quad (11.14)$$

where  $a \in V_T$ ,  $\pi \in \mathcal{B}^\Pi$ ,  $1 \leq r \leq c$ ,  $\phi_{1,r} \in \{\text{RPOP}_p(a) \mid 1 \leq p \leq r, a \in \Gamma\}^*$ , and  $\phi_{2,s} \in \{\text{RPUSH}_q(\alpha) \mid s \leq q \leq c, \alpha \in \Gamma^*\}$ .

**Definition 13** A *colored multiplanar link grammar (CMLG)* is a structure  $G = \langle V_T, \Lambda_D, \Lambda_H, c, \Psi \rangle$  where  $V_T$  is the set of *terminal symbols*,  $\Lambda_D$  and  $\Lambda_H = \{\bar{a} \mid a \in \Lambda_D\}$  are respectively the sets of *dependent* and *governor labels*,  $c$  is the *number of colors*, and  $\Psi$  is the set of

colored rules of the forms

$$*(Y_1 \dots Y_m) \tag{11.15}$$

$$(p_1/V_1 \dots p_n/V_n)* \tag{11.16}$$

$$a(p_1/V_1 \dots p_n/V_n * q/Y_1 Y_2 \dots Y_m) \tag{11.17}$$

$$0(p_1/V_1 \dots p_n/V_n * q/Y_1 Y_2 \dots Y_m) \tag{11.18}$$

where  $a \in V_T$ ,  $0 \notin V_T$ , and  $V_1, V_2, \dots, V_n, Y_1, Y_2, \dots, Y_m \in \Lambda_D \cup \Lambda_H$ ,  $p_1, p_2, \dots, p_n, q \in [1, 2, \dots, c]$  and  $p_i \leq p_{i+1}$  for  $1 \leq i \leq n$ . Moreover, in rules of type (11.18)<sup>1</sup> it holds that  $\max\{p_1, p_2, \dots, p_n\} + 1 \leq q \leq c$ .

The semantics of the grammar  $G$  is defined by reducing it to a nonterminal-free Norm $\mathcal{S}_{pd}^{c,\Gamma}$ -grammar  $G = (V_N, V_T, P, S, \kappa_0)$ , with stack alphabet  $\Gamma = \{(\overleftarrow{x}, \_), (\overrightarrow{x}, \_) \mid x \in \Lambda_D\}$  and set  $P$  containing the following productions:

$$S \rightarrow \text{if TRUE then } X \text{ PUSH}_1(y_m y_{m-1} \dots y_1)$$

for each rule of type (11.15);

$$X \text{ POP}_{p_n}(v_n) \dots \text{POP}_{p_1}(v_1) \rightarrow \text{if } \wedge_{1 \leq p \leq c} \text{TOP}_p(\perp) \text{ then } \epsilon$$

for each rule of type (11.16); and respectively

$$X \text{ POP}_{p_n}(v_n) \dots \text{POP}_{p_1}(v_1) \rightarrow \text{if TRUE then } aX \text{ PUSH}_q(y_m y_{m-1} \dots y_1),$$

$$X \text{ POP}_{p_n}(v_n) \dots \text{POP}_{p_1}(v_1) \rightarrow \text{if TRUE then } X \text{ PUSH}_q(y_m y_{m-1} \dots y_1)$$

for each rule of type (11.17) and (11.18), where  $v_i \in \lambda(V_i)$  and  $y_i \in \rho(Y_i)$  and functions  $\lambda, \rho : (\Lambda_D \cup \Lambda_H) \rightarrow 2^\Gamma$  are defined as follows:

$$\begin{aligned} \lambda(\overline{x}) &= \{(\overleftarrow{x}, \_)\} & \rho(\overline{x}) &= \{(\overleftarrow{x}, \_)\} \\ \lambda(x) &= \{(\overrightarrow{x}, \_)\} & \rho(x) &= \{(\overrightarrow{x}, \_)\}, \end{aligned}$$

### 11.5 Mild Context-Sensitivity of CMLGs

The notion of mild context-sensitivity is an attempt by Joshi (1985) to express the formal power needed to define natural languages.

**Definition 14** A class of grammars is *mildly context-sensitive* if the grammars of this class are (i) polynomial time parseable and (ii) they capture multiple dependencies, limited crossing dependencies and the copy language and its grammars generate (iii) a proper superclass of context-free languages where (iv) all the languages have linear growth property.

**Theorem 11**  $\mathbf{G}_c$ , where  $c \geq 2$ , is mildly context-sensitive.

*Proof.* We will now prove that  $\mathbf{G}_c$  has the properties (i) - (iv).

---

<sup>1</sup>If we drop the rules of the form (11.18), we will not be able to express the copy language as required for MCS grammars.

- (i) As shown previously, each CMLG  $G \in \mathbf{G}_c$  can be reduced to a polynomial size ERL- $\mathcal{S}_{\text{pd}}^{c,\Gamma}$ -G, and the latter is known to be polynomially parseable. The class  $\mathbf{G}_c$  are, thus, polynomially parseable.
- (ii) The capability to describe multiple dependencies is usually represented as an ability to describe languages  $L_1 = \{a^n b^n c^n \mid n \geq k\}$ ,  $L_2 = \{a^n b^m c^n m^n \mid m, n \geq k\}$ , and  $L_3 = \{uw \mid w \in \{a, b\}^i, i \geq k\}$ , where  $k$  is a positive integer.

For the language  $L_1 = \{a^n b^n c^n \mid n \geq 2\}$ , we construct a CMLG  $G = \langle V_T, \Lambda_D, \Lambda_H, 2, \Psi \rangle$  where  $V_T = \{a, b, c\}$ ,  $\Lambda_D = \{A, B, C, b, c\}$  and with the rules

$$\begin{array}{lll} *(A) & (1/\overline{C})* & a(1/\overline{A} * 1/A b) \\ a(1/\overline{A} * 1/B b) & b(1/\overline{b} 1/\overline{B} * 2/B c) & b(1/\overline{b} 2/\overline{B} * 2/C c) \\ c(2/\overline{c} 2/\overline{C} * 2/C) & c(2/\overline{c} 2/\overline{C} * 1/C) & \end{array}$$

For the language  $L_2 = \{a^n b^m c^n d^m \mid m, n \geq 1\}$ , we construct a CMLG  $G = \langle V_T, \Lambda_D, \Lambda_H, 2, \Psi \rangle$  where  $V_T = \{a, b, c, d\}$ ,  $\Lambda_D = \{A, B, C, D, c, d\}$  and with the rules

$$\begin{array}{llll} *(A) & (1/\overline{D})* & a(1/\overline{A} * 1/A c) & b(1/\overline{A} * 2/B d) \\ b(2/\overline{B} * 2/B d) & b(2/\overline{B} * 2/C d) & c(1/\overline{c} 2/\overline{C} * 2/C) & \\ c(1/\overline{c} 2/\overline{C} * 2/D) & d(2/\overline{d} 2/\overline{D} * 2/D) & d(2/\overline{d} 2/\overline{D} * 1/D) & \end{array}$$

For the language  $L_3 = \{uw \mid w \in \{a, b\}^i, i \geq 2\}$ , we construct a CMLG  $G = \langle V_T, \Lambda_D, \Lambda_H, 2, \Psi \rangle$  where  $V_T = \{a, b\}$ ,  $\Lambda_D = \{U, V, Y, a, b\}$  and with the rules

$$\begin{array}{llll} *(U) & (1/\overline{Y})* & a(1/\overline{U} * 1/a U) & \\ b(1/\overline{U} * 1/b U) & 0(1/\overline{a} 1/\overline{U} * 2/V a) & 0(1/\overline{b} 1/\overline{U} * 2/V b) & \\ 0(1/\overline{a} 2/\overline{V} * 2/V a) & 0(1/\overline{b} 2/\overline{V} * 2/V b) & a(2/\overline{a} 2/\overline{V} * 2/V a) & \\ b(2/\overline{b} 2/\overline{V} * 2/V b) & a(2/\overline{a} 2/\overline{V} * 1/Y a) & b(2/\overline{b} 2/\overline{V} * 1/Y b) & \end{array}$$

- (iii) We have already shown that grammars in  $G \in \mathbf{G}_c$ ,  $c \geq 2$ , can express non-context-free languages. Inclusion of all context-free languages is shown by reduction from the Hays-Gaifman dependency grammars. The rule set  $\Pi$  of these grammars contain rules of the following three types:

1.  $*(X)$  — gives word categories the elements of which may govern the sentence,
2.  $X(V_1 V_2 \dots V_n * Y_1 Y_2 \dots Y_m)$  — gives those categories which may derive directly from the category  $X$  and specified their relative positions, and
3.  $X : w$  — gives for word category  $X$  a word  $w$  belonging to it.

We construct a CMLG  $G = \langle V_T, \Lambda_D, \Lambda_H, c, \Psi \rangle$  where  $V_T = \{w \mid (X : w) \in \Pi\}$ ,  $\Lambda_D = \{X \mid X : w \in \Pi\}$  and with the set  $\Psi$  consisting of rules  $(*(X))$  for each  $(*(X)) \in \Psi$ , and of rules

$$w(\overline{X} \ 1/V_1 \ 1/V_2 \ \dots \ 1/V_n \ * \ 1/Y_1 \ Y_2 \ \dots \ Y_m), \text{ and}$$

$$w(1/V_1 \ 1/V_2 \ \dots \ 1/V_n \ * \ 1/Y_1 \ Y_2 \ \dots \ Y_m \ \overline{X})$$

for each  $(X(V_1 \ V_2 \ \dots \ V_n \ * \ Y_1 \ Y_2 \ \dots \ Y_m)) \in \Pi$  and  $(X : w) \in \Pi$ .

(iv) Languages of CMLGs have linear growth property because CMLGs can be reduced to ERL- $\mathcal{S}_{pd}^{c,\Gamma}$ -Gs that have this property.  $\square$

We denote the class of CMLG with  $c$  colors by  $\mathbf{G}_c$ .

CMLGs can associate to the input general dependency graphs. In fact, for any finite dependency graph attached to a string of word tokens, there is a CMLG that generates the string and associates this structure to the string:

**Theorem 12** *If  $D = (V, E)$  is a directed graphs without cycles of length 1 and  $\prec$  is an order among the vertices  $V$ , then there is a CMLG that generates a string  $v_1 v_2 \dots v_{|V|} \in V^*$ , where  $v_i \prec v_{i+1}$  for all  $1 \leq i \leq |V| - 1$ , with a derivation tree whose storage operations encode the edges  $E$ .*

*Proof.* A edges (links) of the directed graph  $D$  can be colored in a correct manner using a method that has been informally presented in Yli-Jyrä (to appear). After the links have been colored, we know the number of colors required, and it is easy to extract from the colored  $D$  a CMLG that generates the string  $v_1 v_2 \dots v_{|V|}$  and associates the desired derivation tree for the string. The details are omitted for brevity.  $\square$

However, the current definition of CMLGs has the restriction that the number of links leaving each word token is bounded by the grammar. Linguistically, having no free dependents is a severe restriction, but we believe that it is relatively easy to simulate such features in the CMLG.

**Theorem 13** *Link grammars (Sleator and Temperley 1991) without connectors that can link one or more tokens are reducible to  $G \in \mathbf{G}_1$ .*

*Proof.* Omitted for brevity.  $\square$

It should be noted, that underspecific link colors could be used in the rules of CMLGs in order to gain more flexibility when the possible linearisations are unknown. An underspecific rule can be seen as a rule-schema that is expanded to a finite set of normal rules.

## 11.6 $c$ -Colored $t$ -Non-projective Dependency Grammar

Linguistically oriented dependency grammars describe usually acyclic graphs. We are therefore interested to find a fragment of CMLG that would generate only acyclic graphs.

It is a well known that acyclicity of finite graphs is not first-order definable property. We get acyclicity neither as a by-product of derivation, because the derivation trees of the underlying non-terminal-free  $\text{NormS}_{\text{pd}}^{c,\Gamma}$ -grammars can be completely different from the link structure represented by the storage operations. There is no limit for the length of “almost cyclic” paths that can be contained in the link structure. Thus, we have to enforce acyclicity of the link structure by some other means, by making a restriction to a subclass of acyclic structures.

In the following, we will propose a parametrized solution that is based on a new complexity measure called the *non-projectivity depth* of dependency paths.

**Definition 15** A sequence of directed dependency links connecting string position  $i$  to string position  $j$  so that  $j$  is transitively governed by  $i$  is called a *dependency path*. The *non-projectivity depth* of an acyclic dependency path is the sum of the number of visited string positions for which the incoming (governor) link is shorter than the outgoing (dependent) link and the number of positions where the incoming link is stored to a pushdown whose index is greater than the index of the pushdown containing the outgoing link.

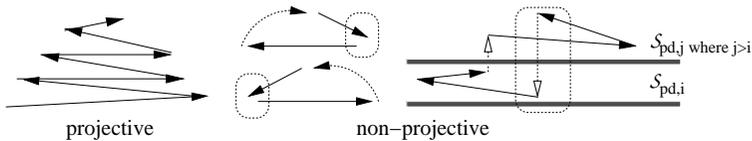


FIGURE 3 Cycles contain at least one special kind of linkage (in rounded boxes) that are not found in *acyclic projective* graphs.

Using the non-projectivity depth is motivated by the facts that (i) it is always greater than zero for cyclic dependency paths, and (ii) it is constantly zero for projective dependency trees. Moreover, if we let grammars assign each token  $a$  the maximum non-projectivity depth of a path from an independent token to token  $a$ , the grammars will fail to build cyclic dependency paths, because such the maximum is not well-defined if a path is acyclic. Based on these observations, we define a subset of CMLGs that can generate only acyclic dependency graphs (but not all them).

**Definition 16** A *colored  $t$ -non-projective dependency grammar (CNDG)* is a structure  $G = \langle V_T, \Lambda_H, \Lambda_D, c, \Psi, t \rangle$  where  $V_T, \Lambda_D, \Lambda_H, c$ , and  $\Psi$  are defined in the same way as done for CMLGs, and  $t$  is the bound for non-projectivity depth in dependency paths.

The semantics of grammar  $G$  is much like in the case of CMLGs (Definition 13). However, there are the following differences:

- The stack alphabet will be  $\Gamma = \{(\overleftarrow{x}, i), (\overrightarrow{x}, i) \mid x \in \Lambda_D, 0 \leq i \leq t\}$ .
- The functions  $\lambda, \rho : (\Lambda_D \cup \Lambda_H) \rightarrow 2^\Gamma$  are defined in such a way that

$$\begin{aligned} \lambda(\overline{x}) &= \{(\overrightarrow{x}, i) \mid i \in [0..t]\} & \rho(\overline{x}) &= \{(\overleftarrow{x}, i) \mid i \in [0..t]\} \\ \lambda(x) &= \{(\overleftarrow{x}, i) \mid i \in [0..t]\} & \rho(x) &= \{(\overrightarrow{x}, i) \mid i \in [0..t]\}. \end{aligned}$$

- From the obtained productions of the forms (11.13) and (11.14) we keep only those productions where the counters  $i$  in the pairs  $(X, i)$  (i) increase monotonically from incoming (governor) links to outgoing (dependent) links, (ii) increase strictly when an outgoing link is longer than an incoming link of the same side, (iii) increase strictly in left outgoing links with color  $p$  when there is a right incoming link with a color  $q \geq p + 1$ , and (iv) do not increase more than necessary. This is expressed formally as follows:

Let  $(a_i, l_i) = v_i$  and  $(b_j, r_j) = y_j$ , where  $1 \leq i \leq n$  and  $1 \leq j \leq m$ , be the stack symbols and  $\alpha \in V_T \cup \{\epsilon\}$  be the lexical anchor in a constructed production

$$X \text{ POP}_{p_n}(v_n) \cdots \text{POP}_{p_1}(v_1) \rightarrow \text{if TRUE then } \alpha X \text{ PUSH}_q(y_m y_{m-1} \cdots y_1).$$

This production is kept if, for all  $i, j$ ,  $1 \leq i \leq n$ ,  $1 \leq j \leq m$ , it holds that

$$\begin{aligned} a_i \in \overleftarrow{\Gamma} &\text{ implies } t_i = \max\{l_*, r_*, l_{n,i+1}, s_{p_i}\}, \text{ and} \\ b_j \in \overrightarrow{\Gamma} &\text{ implies } u_j = \max\{l_*, r_*, r_{j-1,1}\}, \end{aligned}$$

where

$$\begin{aligned} l_* &= \max\{0\} \cup \{l_i & \mid 1 \leq i \leq n, a_i \in \overrightarrow{\Gamma}\} \\ r_* &= \max\{0\} \cup \{r_j & \mid 1 \leq j \leq m, a_j \in \overleftarrow{\Gamma}\} \\ l_{n,k} &= \max\{0\} \cup \{l_i + 1 & \mid k \leq i \leq n, a_i \in \overrightarrow{\Gamma}\} \\ r_{k,1} &= \max\{0\} \cup \{r_j + 1 & \mid 1 \leq j \leq k, a_j \in \overleftarrow{\Gamma}\} \\ s_p &= \max\{0\} \cup \{r_i + 1 & \mid 1 \leq i \leq m, a_i \in \overleftarrow{\Gamma}, p < q\}. \end{aligned}$$

**Theorem 14** *Classes of CNDG with at least two colors are mildly context-sensitive.*

*Proof.* The proof can be given in a similar way as in Theorem 11. Note in particular, that the example grammars for languages  $L_1, L_2$  and  $L_3$  and the representation for the Hays-Gaifman dependency grammars given there have bounded non-projectivity depth.  $\square$

The following theorem relates CNDGs to CMLGs:

**Theorem 15** *Every CNDG can be reduced to a CMLG.*

*Proof.* Instead of constructing a nonterminal-free  $\text{Norm}\mathcal{S}_{\text{pd}}^{c,\Gamma}$ -grammar directly from each CNDG as we did above, we will now have to construct in a similar way a CMLG where counters for non-projectivity depth are visible already in the link labels and in colored rules.  $\square$

The number of productions in CNDGs can be much larger than in corresponding CMLGs. However, it is possible to parse the sentence first with a CMLG and then re-parse the parse forest using CNDGs that will filter out dependency graphs whose non-projectivity depth is greater than  $t$ . When used in this way, CNDGs may in fact provide more efficient filtering than what is generally possible by complete methods (e.g. backtracking search) for acyclicity testing.

## 11.7 Grammars for Dependency Trees

An acyclic dependency graph is a dependency tree if and only if

- it has a unique root
- all the word tokens except the root are governed by exactly one other node.

To obtain dependency grammars that assign dependency trees to the strings, we can specialize colored non-projective dependency grammars so that these two requirements are satisfied. First, we require that all the rules of the forms (11.17) and (11.18) contain exactly one governor link, and that the rules of the form (11.15) or the form (11.16) contain only one category ( $n = 1$  or  $m = 1$ ). Furthermore, the derivations where rules of both forms (11.15) and (11.16) must be discarded.

## 11.8 Conclusion

We have presented new classes of link and dependency grammars, namely the Colored Multiplanar Link Grammar (CMLG) and its subtypes, the  $c$ -Colored  $t$ -Non-projective Dependency Grammar (CNDG). The semantics of these grammars was given by reduction to Extended Right-Linear  $\mathcal{S}_{\text{pd}}^{c,\Gamma}$ -grammar Wartena (2001), which immediately relates CMLGs and CNDGs with existing families of grammars.

The important contribution of this paper is to show that CMLGs and CNDGs are mildly context-sensitive and that the number of colors  $c$  available in CMLGs induce an infinite hierarchy of classes of CMLGs. Furthermore, a sub-hierarchy of CNDGs in each class of CMLGs is obtained by restricting the non-projectivity depth of the dependency

paths. Dependency grammars that generate dependency trees up to a bounded non-projectivity depth form a subset of CNDGs.

We argue that the presented grammars have linguistic and practical relevance, because (i) the core ideas of the CMLG derivations have been tested against a small treebank (Yli-Jyrä 2003), (ii) the CMLG hierarchy provides a useful complexity measure for natural language sentences (iii) CMLGs are mildly context-sensitive and (iv) lexicalized, and (v) they give rise to finite-state approximations that assign non-projective dependency structures to strings (Yli-Jyrä 2004).

(This paper version appears in FGNancy 2004 pre-proceedings.)

## Acknowledgements

The work was partially funded by NorFA under the personal Ph.D. scholarship (ref.nr. 010529) of the first author.

## References

- Castaño, J. M. 2003. Global index grammars and descriptive power. In R. T. Oehrlé and J. Rogers, eds., *Proceedings of MOL 8, 2003*.
- Gaifman, H. 1965. Dependency systems and phrase-structure systems. *Inf. Control* 8(3):304–37.
- Gazdar, G. 1988. Applicability of indexed grammars to natural languages. In U. Reyle and C. Rohrer, eds., *Natural Language Parsing and Linguistic Theories*. Dordrecht: Reidel.
- Hays, D. G. 1964. Dependency theory: A formalism and some observations. *Language* 40:511–525.
- Joshi, A. K. 1985. Tree Adjoining Grammars: how much context-sensitivity is required to provide reasonable structural descriptions? In D. Dowty, L. Karttunen, and A. Zwicky, eds., *Natural Language Parsing*, pages 206–250. Cambridge: Cambridge University Press.
- Sleator, Daniel and Davy Temperley. 1991. Parsing english with a link grammar. Technical Report CMU-CS-91-196, Carnegie Mellon University, Computer Science.
- Tesnière, L. 1959. *Éléments de Syntaxe Structurale*. Paris: Editions Klincksieck.
- Vijay-Shanker, K., David Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalism. In *25th ACL*, pages 104–111. Stanford, CA.
- Wartena, C. 2001. Grammars with composite storages. In M. Moortgat, ed., *LACL'98*, vol. 2014 of *LNAI*, pages 266–285.
- Weir, David J. 1994. Linear iterated pushdowns. *Computational Intelligence* 10(4):422–430.

- Yli-Jyrä, A. 2003. Multiplanarity - a model for dependency structures in treebanks. In *The Second Workshop on Treebanks and Linguistic Theories*. Växjö, Sweden.
- Yli-Jyrä, A. 2004. Axiomatization of non-projective dependency trees through finite-state constraints that analyse crossing bracketings (preliminary title). In *the COLING 2004 workshop "Recent Advances in Dependency Grammar"*. University of Geneva, Switzerland.
- Yli-Jyrä, A. to appear. Coping with dependencies and word order or how to put Arthur's court into a castle. In H. Holmboe, ed., *Nordisk Sprogteknologi 2003. Årbog for Nordisk Sprogteknologisk Forskningsprogram 2000-2004*.