Proceedings of FGVienna: The 8th Conference on Formal Grammar

Gerald Penn (ed.)

August 16–17, 2003

CENTER FOR THE STUDY OF LANGUAGE AND INFORMATION

Contents

Introduction **vii**

1 k-Valued Link Grammars are Learnable from Strings **3**
Denis Béchet

2 MCSGs for Estimating Maximum Entropy Parsing Models **13**
David Chiang

3 Automating the Generation of an LFG **29**
Lionel Clément and Alexandra Kinyon

4 An Asymmetric Theory of Peripheral Sharing in HPSG **45**
Berthold Crysmann

5 Discourse Plans and Linguistic Meaning **65**
Alexander Dikovsky

6 A Learnable Class of Classical Categorial Grammars **81**
Daniela Dudau-Sofronie, Isabelle Tellier, Marc Tommasi

7 A Note on Countercyclicity and Minimalist Grammars **95**
Hans-Martin Gärtner and Jens Michaelis

8 Graph Properties of HPSG Feature Structures 111
Stephan Kepser and Uwe Mönnich

9 At First Blush on Tenterhooks 123
Jan-Philipp Soehn and Manfred Sailer

10 A New Application for Raising in HPSG: Complex Prepositions 139
Beata Trawiński

11 Regular Approximations through Labeled Bracketing 153
Anssi Yli-Jyrä

Index 167

Introduction

The Formal Grammar conference series began in 1996 as a forum for new and original linguistics research within formal frameworks, to give linguists and mathematicians an opportunity to interact, and thereby foster precise linguistic analyses along with new generations of empirically relevant models for use in characterizing the structure of natural languages. Although the representation of mainstream linguists at the European Summer School of Logic, Language and Information (ESSLLI), with which Formal Grammar is usually co-located, has indeed waned over the years, Formal Grammar has so far managed to maintain a very pleasing balance between theory and practice that continues to attract an impressive range of scholarship from around the world.

The 8th conference, held in Vienna on the 16th and 17th of August, 2003, was no exception, with contributions to syntax in the areas of grammar engineering (Clément and Kinyon), coordination and linearization (Crysmann), countercyclic adjunction (Gärtner and Michaelis), selectional restrictions by modifiers (Soehn and Sailer), and complex prepositions (Trawiński), and to formal methods in the areas of learnability (Béchet; Dudau-Sofronie, Tellier, and Tommasi), statistical parsing (Chiang), formal meaning representation (Dikovsky), finite model theory (Kepser and Mönnich), and formal language theory (Yli-Jyrä).

The newly appointed organizing committee for Formal Grammar asked the attendees of the 7th conference (Trento) for feedback on improvements they should make to the series, and there was a unanimous consensus that the proceedings of the conference should be published. On behalf of the other organizers, I am therefore very proud to present these proceedings of the 8th conference under the impressum of CSLI Publications, in the hope that this will be the first of many Formal Grammar conferences to offer them.

Acknowledgments

I thank the other members of the organizing committee, Gerhard Jaeger, Paola Monachesi, and Shuly Wintner, for their assistance with organizing the Vienna conference. A special thanks is also due to our programme committee without whose volunteer efforts such a conference would not have been possible: Gosse Bouma (Groningen), Chris Brew (Ohio State), Miriam Butt (Manchester), Philippe de Groote (Nancy), Maarten de Rijke (Amsterdam), Mark Hepple (Sheffield), Ruth Kempson (London), András Kornai (Northern Light), Geert-Jan Kruijff (Saarbrücken), Guido Minnen (Motorola), Uwe Mönnich (Tübingen), Michael Moortgat (Utrecht), Mark-Jan Nederhof (Groningen), James Rogers (Earlham), and Anoop Sarkar (Simon Fraser). We are likewise very grateful to our other reviewers for their contributions: Nissim Francez (Technion), Thilo Goetz (IBM), Nizar Habash (College Park), Sandra Kuebler (Tübingen), Jonas Kuhn (Austin), Rob Malouf (San Diego State), Philip Miller (Lille), Stephan Müller (Bremen), Ed Stabler (UCLA), and Yoad Winter (Technion).

Los Angeles, July, 2007

1

k-Valued Link Grammars are Learnable from Strings

Denis Béchet

Abstract The article is concerned with learning link grammars in the model of Gold. We show that rigid and k-valued link grammars are learnable from strings. In fact, we prove that the languages of link structured lists of words associated to rigid link grammars have finite elasticity and we show a learning algorithm. As a standard corollary, this result leads to the learnability of rigid or k-valued link grammars learned from strings.

1.1 Introduction

Link grammars, introduced in Sleator and Temperley (1991, 1993), are a formal grammatical system for natural language processing. A sentence is recognized if there is a way to connect correctly the specification of the words in the lexicon using *links*. Very close to dependency grammars (Mel'cuk, 1988), link grammars have the advantage to show directly the connections of words in a graph (not only as a tree like dependency grammars) which can be useful for a direct translation from syntax to semantics.

Since link grammars are completely lexicalized, they are well adapted to learning perspectives and an actual way of research is to determine what are the lexicalized grammars that remain learnable in the sense of Gold (1967). In fact, learning link grammars was mainly studied for probabilistic variants (Della Pietra et al., 1994, Fong and Wu, 1995, Kübler, 1998) or with a context where only one word is unknown and

Proceedings of FGVienna: The 8th Conference on Formal Grammar.
Gerald Penn (ed.).
Copyright © 2008, CSLI Publications.

has to be inferred in a sentence (Pedersen and Chen, 1995).

Here, we recall that learning consists to define an algorithm on a finite set of sentences that converge to obtain a grammar in the class that generates the examples. Let $\mathcal{G}$ be a class of grammars that we wish to learn from positive examples. Formally, let $\mathcal{L}(G)$ denote the language associated with grammar G, and let V be a given alphabet. A learning algorithm is a function ϕ from finite sets of words in V^* to $\mathcal{G}$, such that for $G \in \mathcal{G}$ with $\mathcal{L}(G) = (e_i)_{i \in \mathcal{N}}$ there exist a grammar $G' \in \mathcal{G}$ and $n_0 \in N$ such that: $\forall n > n_0,\ \phi(\{e_1, \ldots, e_n\}) = G' \in \mathcal{G}$ and $\mathcal{L}(G') = \mathcal{L}(G)$. After the initial pessimism following the unlearnability results in Gold (1967), there has been a renewed interest due to learnability of non trivial classes from Angluin (1980) and Shinohara (1990). Recent works from Kanazawa (1998) have answered the problem for different classes associated to classical categorial grammars. In the paper, we try to answer to the question for link grammars: are they learnable from strings?

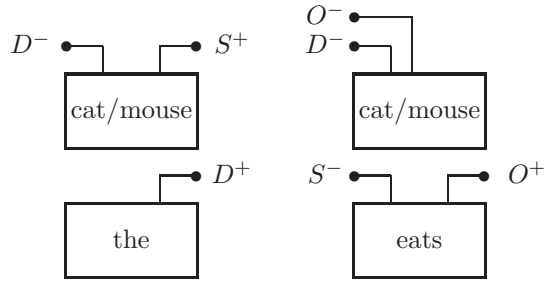
The paper is organized as follows. Section 2 presents link grammars. Section 3 reminds some useful properties on languages and learning algorithms. Section 4 gives the main lemma: rigid untyped link net grammars have finite elasticity and thus is learnable. This section ends with a learning algorithm for this class of structured languages. Section 5 extends this result and proves that k-valued link grammars are learnable from strings. Section 6 concludes and gives some perspectives.

1.2 Link Grammar

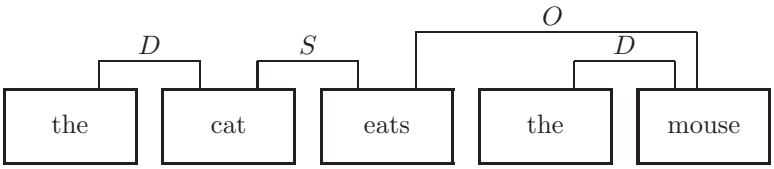
A link grammar connects the words of a sentence by links so as to satisfy:

- **Planarity**: The links do not cross.
- **Connectivity**: The graph is connected.
- **Ordering**: The lexicon gives the left and right possible linkages of each word.
- **Exclusion**: No two links may connect the same pair of words.

For instance, the link grammar with the following nodes:



accepts the sentence "the cat eats the mouse":



A link is annotated with a primitive type that must match the left and right constraints of the two linked words. It corresponds to a functional dependency: S for subject, D for determinant, O for object.

1.2.1 Formal Definitions

- Let Σ be an *alphabet* and Pr a set of *primitive types*.
- A *link net* is a non-empty, planar and connected graph with a total order on vertices which must be on the border of the graph and where edges are annotated by primitive types and vertices by words. The set of link nets over Σ is noted $\mathcal{N}_{Pr}(\Sigma)$. Formally, let $(\mathcal{V}, \leq)$ be a totally ordered set ($\mathcal{N}$ for instance), a link net is a structure (V, w, E, t) where:
  - $V \subseteq \mathcal{V}$, the vertices, is a non-empty finite subset of $\mathcal{V}$ written $(v_1, \ldots, v_n)$ where $n = \#V$ and $v_1 < \cdots < v_n$;
  - $w : V \mapsto \Sigma$ maps each vertex to a word;
  - $E \subseteq V \times V$, the edges, is a symmetrical[1] and anti-reflexive[2] subset of $V \times V$;
  - $t : E \mapsto Pr$ maps each edge to a primitive type;
  - The edges do not cross[3]: if $(a, b) \in E$ and $(c, d) \in E$ such that $a < b$ and $c < d$ then it is not possible that $a < c < b < d$ or $c < a < d < b$.
  - The graph (V, E) is connected.[4]

[1] A relation E is symmetrical iff $(x, y) \in E \Leftrightarrow (y, x) \in E$
[2] A relation E is anti-reflexive iff $(x, x) \notin E$
[3] Thus the graph (V, E) is planar and the vertices are on its border.
[4] For every pair of vertices, there exists a path between them.

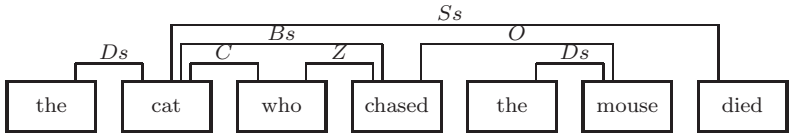


FIGURE 1 A link net

- An *untyped link net* is a link net where edges are not annotated by primitive types. If $N = (V, w, E, t)$ is a link net, we write $untyped(N) = (V, w, E)$ the corresponding untyped link net. The set of untyped link nets over Σ is noted $\mathcal{N}(\Sigma)$.
- The *yield* of a link net $N = ((v_1, \ldots, v_n), w, E, t)$ or of an untyped link net $N = ((v_1, \ldots, v_n), w, E)$ is $yield(N) = w(v_1) \cdots w(v_n) \in \Sigma^+$.
- The set of *link nodes* over *Pr* noted *Tp* is the set of pairs of finite lists of *Pr*. A link node X has a *left list of ports* noted $t_n^-, \ldots, t_1^-$ and a *right list of ports* noted $t_1^+, \ldots, t_m^+$. The *left arity* noted $a^l(X)$ is n and the *right arity* noted $a^r(X)$ is m. A link node is noted by its ports $t_n^-, \ldots, t_1^-, t_1^+, \ldots, t_m^+$. An *untyped link node* is just characterized by its left and right arity.

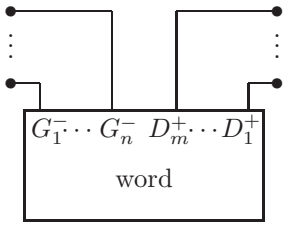


FIGURE 2 A link node

- For each vertex v of a link net $N = (V, w, E, t)$, the set of edges ending in v can be split into a left and a right lists $(x_n, v), \ldots, (x_1, v)$ and $(v, y_1), \ldots, (v, y_m)$ where $x_n < x_{n-1} < \cdots < x_1 < v < y_1 < \cdots < y_{m-1} < y_m$. Thus, v is associated to the link node $node(v) = t(x_n, v)^-, \ldots, t(x_1, v)^-, t(v, y_1)^+, \ldots, t(v, y_m)^+$.
- A *link grammar* is a structure $G = (\Sigma, I)$ where $I : \Sigma \mapsto \mathcal{P}^f(Tp)$ is a function that maps to each element of Σ a finite set of *link nodes*.
- For $a \in \Sigma$, if $A \in I(a)$, G *associates* A to a (written $G : a \mapsto A$).
- A link net $((v_1, \ldots, v_n), w, E, t)$ is generated by G iff $G : w(v_i) \mapsto t(v_i)$ for all i, $0 \leq i \leq n$.
- An untyped link net N is generated by G iff it exists a link net N' such that $N = untyped(N')$ and N' is generated by G.
- A sentence $c_1 \cdots c_n \in \Sigma^+$ is generated by G iff it exists a link net N such that $c_1 \cdots c_n = yield(N)$ and N is generated by G.

- The language of link nets of G, noted $\mathcal{L}_{\mathcal{N}_{Pr}(\Sigma)}(G)$ is the set of link nets generated by G.
- The language of untyped link nets of G, noted $\mathcal{L}_{\mathcal{N}(\Sigma)}(G)$ is the set of untyped link nets generated by G.
- The language of strings of G, noted $\mathcal{L}_{\Sigma^+}(G)$ is the set of sentences generated by G.
- Link grammars that associate at most k link nodes to each symbol of Σ are called *k-valued*.
- 1-valued grammars are also called *rigid*.

1.3 Learnability and Finite Elasticity

We have seen in the introduction that a class of languages described by a class of grammars $\mathcal{G}$ is learnable iff there exists a learning algorithm ϕ from finite sets of words to $\mathcal{G}$ that converges to G[5] for any $G \in \mathcal{G}$ and for any growing partial enumeration of $\mathcal{L}(G)$.

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 induces learnability. A very nice presentation of this notion can be found in Kanazawa (1998).

Definition

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

Theorem 1 [**Wright 1989**] *A class that is not learnable has infinite elasticity.*

Corollary 2 *A class that has finite elasticity is learnable.*

Finite elasticity is a very nice property because it can be extended from a class to another one that is obtained by a *finite-valued relation*.[6] We use here a version of the theorem that has been proved in Kanazawa (1998) and is useful for various kind of languages (strings, structures, nets) that can be described by lists of elements of some alphabets.

Theorem 3 [**Kanasawa 1998**] *Let $\mathcal{M}$ be a class of languages over Γ that has finite elasticity, and let $R \subseteq \Sigma^* \times \Gamma^*$ be a finite-valued relation.*

[5] In fact, it is not the output grammars that converge but their associated languages.

[6] 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 $(s, u) \in R$.

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.*

1.4 Finite Elasticity of Rigid Untyped Link Net Grammars

This section is concerned by grammars of untyped link nets rather than grammars of strings. The following theorem is essential because, as a corollary, the corresponding class of rigid link grammars has finite elasticity and thus is learnable from strings. This result can also be extended to the class of k-valued link grammars for every k.

Theorem 4 *Rigid link grammars define a class of languages of untyped link nets that has finite elasticity.*

Proof: We use a result of Shinohara (1990, 1991) that proves that *formal systems* that has *finite thickness* has finite elasticity. In Shinohara (1991) this 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) do not describe languages of strings only but also languages of terms. It can be applied to typed or untyped link nets which can be seen as well-bracketed strings (each link is associated to an opening and a closing (typed) bracket). For the class of rigid untyped link net grammars, the sketch of proof is as follows:

1. **Definition.** A link grammar $G_1 = (\Sigma_1, I_1)$ is *included* in a link grammar $G_2 = (\Sigma_2, I_2)$ (notation $G_1 \subseteq G_2$) iff $\Sigma_1 \subseteq \Sigma_2$ and $\forall x \in \Sigma_1$, $I_1(x) \subseteq I_2(x)$.
2. **Definition and lemma.** The mapping $\mathcal{L}_{\mathcal{N}(\Sigma)}$ from link grammars to untyped link net languages is monotonic: if $G_1 \subseteq G_2$ then $\mathcal{L}_{\mathcal{N}(\Sigma)}(G_1) \subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(G_2)$.
3. **Definition.** A grammar G is *reduced with respect to a set* $X \subseteq \mathcal{N}(\Sigma)$ iff $X \subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(G)$ and for each grammar $G' \subseteq G$, $X \not\subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(G')$. Intuitively, a grammar that is reduced with respect to X, does not have redundant expression to cover all the structures of X.
4. **Lemma.** For each finite set $X \subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(G)$, there is a finite set of rigid untyped link net languages that correspond to the grammars that are reduced from X. This is the main part of the proof. In fact, if $G = (\Sigma, I)$, a rigid untyped link net grammar, is reduced with respect to X then each word that does not appear in one of the untyped link net of X must be associated through I to the empty set. The other words must be associated to exactly one

type of Tp (the grammar is rigid). The left and the right arities are given by the occurrences of the word in the untyped link nets and they must be the same for all the occurrences because the language we try to learn corresponds to a *rigid* untyped link 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[7] 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 (1991) proves that a formal system that has bounded finite thickness has finite elasticity.
7. **Corollary.** Rigid untyped link net languages have finite elasticity.

A learning algorithm for rigid untyped link net grammars

In fact, the notion of reduced grammars suggests a simple learning algorithm for the class of rigid untyped link net grammars. The algorithm ϕ_1 takes a sequence of untyped link nets $N_1, \ldots, N_l$ and produces a link grammar that corresponds to the smallest rigid untyped link net language that is compatible to the input. Of course, this algorithm returns a failure if the sequence does not correspond to a rigid untyped link net language.[8]

1. We collect the occurrences of each word together with its left and right arities from the input sequence. The algorithm fails if a word is used with different left or right arities.
2. For each word w, we associates $n+m$ variables corresponding to the n left ports and the m right ports: $X^w_{-n}, \ldots, X^w_{-1}, X^w_1, \ldots, X^w_m$
3. Then, we extract equality constraints on the variables based on the links that appear in the input untyped link nets: a link corresponds to two variables on both ends that must be equal.
4. The equality system is resolved and a primitive type is associated to each equivalence class of variables noted $\overline{X^w_i}$.
5. For each word w that appears in the sequence and is associated to the $n+m$ variables $X^w_{-n}, \ldots, X^w_{-1}, X^w_1, \ldots, X^w_m$, the output link grammar associates the link node $\overline{X^w_{-n}}^-, \ldots, \overline{X^w_{-1}}^-, \overline{X^w_1}^+, \ldots, \overline{X^w_m}^+$ to w.

[7] 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.

[8] In other words, the algorithm returns a failure if this sequence is not included in at least one of the rigid untyped link net languages.

Theorem 5 *ϕ_1 learns rigid untyped link net grammars.*

Proof: ϕ_1 is monotonic: if $S_1 \subseteq S_2$ then $\phi_1(S_2)$ returns a failure or $\phi_1(S_1)$ and $\phi_1(S_2)$ succeed and $\mathcal{L}_{\mathcal{N}(\Sigma)}(\phi_1(S_1)) \subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(\phi_1(S_2))$. This is a consequence of the fact that the equality system corresponding to S_1 is a subset of the equality system corresponding to S_2.

Let $G = (\Sigma, I)$ be a rigid link grammar and $(N_i)_{i\in\mathcal{N}}$ an infinite sequence of untyped link nets that enumerates $\mathcal{L}_{\mathcal{N}(\Sigma)}(G)$. For $i \in \mathcal{N}$, $\phi_1(N_0, \ldots, N_i)$ does not return a failure because G is rigid so each word in the untyped link nets of $\mathcal{L}_{\mathcal{N}(\Sigma)}(G)$ is used with the same left and right arities. Because ϕ_1 is monotonic, $(G_i = \phi_1(N_0, \ldots, N_i))_{i\in\mathcal{N}}$ defines an infinite sequence of growing languages $\mathcal{L}_{\mathcal{N}(\Sigma)}(G_0) \subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(G_1) \subseteq \cdots$. Because the class has finite elasticity, this sequence must converge to a language L_∞ that must be (equal or) a superset of $\mathcal{L}_{\mathcal{N}(\Sigma)}(G)$ since the sequence enumerates $\mathcal{L}_{\mathcal{N}(\Sigma)}(G)$. In fact, for $i \in \mathcal{N}$, G verifies the equality system used by ϕ_1 with $N_0, \ldots, N_i$ as input, so $\mathcal{L}_{\mathcal{N}(\Sigma)}(\phi_1(N_0, \ldots, N_i)) \subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(G)$. Thus, we also have $L_\infty \subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(G)$. ■

1.5 k-valued Link grammars are learnable from strings

Because the class of rigid untyped link net languages has finite elasticity, we can find a finite-valued relation between rigid untyped link net languages and k-valued link languages. In fact, we define two relations and use twice Theorem 3. The first one from rigid untyped link net languages to rigid link languages and the second from rigid link languages to k-valued link languages.

Lemma 6 *Rigid link languages have finite elasticity.*

Proof: An untyped link net $((v_1, \ldots, v_n), w, E)$ over Σ is characterized by the left and right arities of each vertex. Thus, it can be completely described by the following string from the alphabet $\Sigma \cup \{[\,,]\}$ where $[$ and $]$ are not in Σ:

$$w(v_1)\underbrace{[\cdots[}_{a^r(v_1)}\underbrace{]\cdots]}_{a^l(v_2)}w(v_2)\underbrace{[\cdots[}_{a^r(v_2)}\cdots\underbrace{]\cdots]}_{a^l(v_{n-1})}w(v_{n-1})\underbrace{[\cdots[}_{a^r(v_{n-1})}\underbrace{]\cdots]}_{a^l(v_n)}w(v_n)$$

In fact, the relation between Σ^+ and $(\Sigma \cup \{[\,,]\})^+$ that adds square brackets such that the result corresponds to an untyped link net is finite-valued. Then, because the class of languages of $(\Sigma \cup \{[,]\})^+$ corresponding to rigid untyped link net grammars has finite elasticity, their inverse images by the previous relation, which define exactly the class of rigid link languages, have finite elasticity. ■

Lemma 7 *k-valued link languages have finite elasticity.*

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

1.6 Conclusion and perspectives

We have proved in the paper that the class of rigid untyped link net languages has finite elasticity. We have given a learning algorithm for this class of languages of untyped link nets. Finally, we have proved that k-valued link grammars have also finite elasticity and thus is learnable from strings.

This positive result may be compared to other learnability results in the same domain in particular in the field of k-valued categorial grammars. Kanazawa's positive result on classical categorial grammar is very similar to our result but other more complex but very close systems like Lambek calculus or pregroups have been proved to be not learnable from strings (Foret and Le Nir, 2002a,b). Due to the similarity between link grammars and Lambek calculus or pregroups presented as (proof) nets and their differences from the learnability point of view, we can try to deduce general rules for learnable classes.

For us, the reason why k-valued Lambek calculus grammars are not learnable comes from the fact that using a sequence of sentences, we can not bound the possible "interactions" (in term of proof nets, it means different set of axiom links) between two words which is not the case for k-valued link grammars because for one type there is only one possible interaction between two words. This remark may be interesting for defining useful learnable classes of logical categorial grammars that lay between classical categorial grammars and Lambek calculus.

References

Angluin, D. 1980. Inductive inference of formal languages from positive data. *Information and Control* 45:117–135.

Della Pietra, S., V. Della Pietra, J. Gillett, J. Lafferty, H. Printz, and L. Ures. 1994. Inference and estimation of a long-range trigram model. In *Grammatical Inference and Applications: Proceedings of the Second International Colloquium*, pages 78–92. Alicante, Spain.

Fong, E. and D. Wu. 1995. Learning restricted probabilistic link grammars. In *Proceedings of the IJCAI Workshop on New Approaches to Learning for Natural Language Processing*, pages 49–56. Montréal, Canada.

Foret, A. and Y. Le Nir. 2002a. Lambek rigid grammars are not learnable from strings. In *Proceedings of the 19th International Conference on Computational Linguistics*. Taipei, Taiwan.

Foret, Annie and Y. Le Nir. 2002b. On limit points for some variants of rigid Lambek grammars. In *Proceedings of the 6th International Colloquium on Grammatical Inference, LNAI 2484*. Springer.

Gold, E. M. 1967. Language identification in the limit. *Information and Control* 10:447–474.

Kanazawa, M. 1998. *Learnable Classes of Categorial Grammars*. Studies in Logic, Language and Information. Palo Alto: CSLI Publications.

Kübler, S. 1998. Learning a lexicalized grammar for German. In D. Powers, ed., *Proceedings of the 3rd International Conference on New Methods in Language Processing and 2nd International Conference on Computational Natural Language Learning*, pages 11–18. ACL.

Mel'cuk, I. 1988. *Dependency Syntax: Theory and Practive*. Albany: State University of New York Press.

Motoki, T., T. Shinohara, and K. Wright. 1991. The correct definition of finite elasticity: Corrigendum to identification of unions. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, page 375. San Mateo: Morgan Kaufmann.

Pedersen, T. and W. Chen. 1995. Lexical acquisition via constraint solving. In *Working Notes of the AAAI Spring Symposium on Representation and Acquisition of Lexical Knowledge*, pages 118–122.

Shinohara, T. 1990. Inductive inference from positive data is powerful. In *Proceedings of the 1990 Workshop on Computational Learning Theory*, pages 97–110. San Mateo: 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. 1991. Parsing English with a link grammar. Technical Report CMU-CS-91-196, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA.

Sleator, D. and D. Temperley. 1993. Parsing English with a link grammar. In *Proceedings of the Third International Workshop on Parsing Technologies*.

Wright, K. 1989. Identifications of unions of languages drawn from an identifiable class. In *Proceedings of the 1989 Workshop on Computational Learning Theory*, pages 328–333. San Mateo: Morgan Kaufmann.

2

Mildly Context-Sensitive Grammars for Estimating Maximum Entropy Parsing Models

David Chiang

2.1 Introduction

The maximum-entropy framework provides great flexibility in specifying what features a model may take into account, making it effective for a wide range of natural language processing tasks. But because parameter estimation in this framework involves computations over the whole space of possible labelings, it is unwieldy for the parsing problem, where this space is very large.

Researchers have tried several strategies for efficiently training parsing models in the maximum-entropy framework. Ratnaparkhi's parser (1997) models the probabilities of actions of a pushdown automaton instead of the probabilities of entire parses, but for this reason is susceptible to the label-bias problem (Lafferty et al. 2001). Abney (1997) proposes random sampling of the parse space. Johnson et al. (1999) propose using conditional estimation instead of joint estimation. This reduces the space to the possible parses of a single sentence, which is much smaller but can still be unmanageably large for many grammars. Collins' discriminative models (2000) are cast as parse-reranking models instead of parsing models, and are therefore dependent on his earlier generative model (Collins 1999).

Geman and Johnson (2002) and Miyao and Tsujii (2002) propose compact representations of the parse space that make processing

Proceedings of FGVienna: The 8th Conference on Formal Grammar.
Gerald Penn (ed.).
Copyright © 2008, CSLI Publications.

tractable, at the cost of flexibility. Neither, however, details how these compact representations are obtained. Geman and Johnson refer to an algorithm from the LFG literature (Maxwell and Kaplan 1995) to fill this gap in the former method; the purpose of the present paper is to fill this gap in the latter method. We propose to do this by identifying feature forests with the derivation forests of linear context-free rewriting systems (LCFRSs, defined below).

Both of these methods bring tractability at the cost of some generality: features can no longer be completely arbitrary but only *locally* arbitrary. In Section 2.4 we discuss the nature of the locality restriction imposed by feature forests based on LCFRSs, arguing that these formalisms provide ample flexibility in specifying parsing models. We also include a comparison with Geman and Johnson's approach, finding that feature forests are a special case of the packed representations they use, but that feature forests are more efficient for this special case.

2.2 Background

2.2.1 Maximum entropy models

A maximum-entropy model defines a function $\mathbf{f} : \mathcal{T} \to \mathbb{R}^d$ from parses to feature vectors. Let $f_1, \ldots, f_d$ denote the component functions of $\mathbf{f}$. Then the probability of a parse T given a sentence S has the form

$$P(T \mid S) = \frac{1}{Z} \prod_i \theta_i^{f_i(T)} \tag{2.1}$$

where the θ_i are real-valued feature weights and Z is a normalization constant.

There are various methods for obtaining a conditional maximum-likelihood estimate of the θ_i for models of this form, including iterative-scaling methods and gradient-ascent methods (Berger et al. 1996, Malouf 2002), but all require the computation of

$$E(f_i \mid S) = \sum_T f_i(T) P(T \mid S) \tag{2.2}$$

where T ranges over parses of S. Since, for a parsing model, S can have exponentially many parses, it is not practical to perform this summation by brute force. For the same reason, computing the most probable parse

$$\arg\max_T P(T \mid S)$$

is also not practical to compute by brute force.

2.2.2 Miyao and Tsujii's feature forests

Miyao and Tsujii (2002) present a solution to this problem using the notion of a *feature forest*, a compact representation of the feature vectors of all the possible parses of S. Their formulation views forests as and/or graphs, but we provide here a reformulation in terms of context-free grammars (CFGs).

Definition 1 A *feature forest* $F = \langle G, \mathbf{f} \rangle$ is a CFG G together with a mapping $\mathbf{f}$ from productions of G to feature vectors. We can view F as a parameterized weighted CFG by defining the weight of each production $A \to \beta$ to be

$$\prod_i \theta_i^{f_i(A \to \beta)}$$

where, as in (2.1), the θ_i are the feature weights and the f_i are the component functions of $\mathbf{f}$.

Definition 2 The feature vector of a derivation of a feature forest is the sum of the feature vectors of the occurrences of productions in the derivation.

Definition 3 We say that a feature forest F *represents* a set $\mathcal{T}$ of parses if there is a one-to-one correspondence between derivations of F and parses in $\mathcal{T}$ such that each derivation of F has the same feature vector as the corresponding parse.

Note that when F represents the set of all parses of S, the derivations of F need not yield S; only the feature vectors need to match. Therefore the choice of terminal alphabet for F is immaterial.

Miyao and Tsujii's algorithm is essentially the same as the Inside-Outside algorithm (Baker 1979, Lari and Young 1990), which computes the expected number of times that each production will be used. If F is a feature forest representing all the possible parses of S, then

$$E(f_i \mid S) = \sum_{A \to \beta \in F} f_i(A \to \beta) E(A \to \beta) \tag{2.3}$$

where the expectation $E(A \to \beta)$ is computed by running the Inside-Outside algorithm using the weights of F (it makes no difference algorithmically that F is not a proper PCFG), and then dividing by the normalization constant Z from (2.1).

Similarly, though not noted by Miyao and Tsujii, it is simple to use dynamic programming to efficiently compute the feature vector of the most probable parse (Goodman 1999). Also, their algorithm only works on feature forests which generate a finite number of derivations. It can be extended to the general case (Goodman 1999), but we assume here

that all feature forests generate only a finite number of derivations, as this is not a severe limitation.

Miyao and Tsujii's method cannot be used on its own, because it does not provide any way of constructing a feature forest given a sentence. This is a problem for both training and decoding, since the methods described above for both tasks take a feature forest as their starting-point. The solution we propose below is to identify Miyao and Tsujii's feature forests with the derivation forests of linear context-free rewriting systems. Since they are polynomial-time parsable, and the resulting derivation forests have the same structure as feature forests, they tractably bridge the gap in Miyao and Tsujii's method between sentences and feature forests.

In Section 2.3 we describe the details of this solution, and in Section 2.4 we discuss its theoretical and practical possibilities.

2.3 Linear context-free rewriting systems

2.3.1 Definition

Linear context-free rewriting systems or LCFRSs (Vijay-Shanker et al. 1987, Weir 1988) are grammar formalisms with context-free derivation sets and whose elementary operations do not copy or erase arbitrary substrings. Examples of LCFRSs are CFG, tree-adjoining grammar (TAG), and combinatory categorial grammar (CCG).[1]

Definition 4 A *generalized CFG* G is a tuple $\langle V, S, F, P\rangle$, where

1. V is a finite set of nonterminal symbols,
2. $S \in V$ is a start symbol,
3. $\mathcal{F}$ is a finite set of function symbols, and
4. P is a finite set of productions of the form

$$A_0 \to f(A_1, \ldots, A_n) \tag{2.4}$$

where $n \geq 0$, $f \in \mathcal{F}$, and $A_i \in V$.

A generalized CFG rewrites nonterminal symbols just like a CFG, but the end result is a *term*, for example, $f(g(h()), h())$, rather than a string. This term must be evaluated with respect to some interpretation of the function symbols to produce a derived structure. In an LCFRS the interpretation must obey certain constraints, defined below.

Definition 5 We say that an interpretation function $[\![\cdot]\!]$ on terms is *compositional* if for each function symbol f, there is a unique *composi-*

[1] Some variants of CCG are not LCFRSs: for example, if the coordination schema $X \to X$ **and** X has an infinite number of instantiations.

tion operation $\bar{f}$ such that

$$[\![f(t_1, \ldots, t_n)]\!] = \bar{f}([\![t_1]\!], \ldots, [\![t_n]\!]).$$

Definition 6 A grammar of a *linear context-free rewriting system* is a generalized CFG together with a compositional interpretation function, called the *string yield function*, which maps each function symbol f to a composition operation $\bar{f}$ on string tuples definable as:

$$\bar{f}(\langle x_{11}, \ldots, x_{1m_1}\rangle, \ldots, \langle x_{n1}, \ldots, x_{nm_n}\rangle) = \langle w_1, \ldots, w_m\rangle \tag{2.5}$$

where the w_i are concatenations of the variables x_{ij} and symbols from a terminal alphabet Σ, and every variable which appears on the left-hand side appears exactly once on the right-hand side.

We also define a function ϕ from nonterminal symbols to integers as follows: for each nonterminal A, it must be the case that m in equation (2.5) is the same for all productions with left-hand side A; let $\phi(A)$ be this value.

For example, the composition operations of a CFG are defined over single strings (tuples of length 1) and simply concatenate their arguments. In a TAG, it is convenient to define a *tree* yield function first: each tree composition operation corresponds to an elementary tree γ, taking a number of derived auxiliary trees as arguments and evaluating to γ with the arguments adjoined into it. Define the *yield* of a derived auxiliary tree be $\langle w_l, w_r\rangle$, where w_l and w_r are the yield of the tree to the left and right of the foot, respectively. Then each tree composition operation $\bar{f}$ induces a string composition operation, taking yields of derived auxiliary trees as arguments and evaluating to the yield of the value of $\bar{f}$ on those trees.

Note, in fact, that any grammar which is to produce parse trees, for example, as found in the Penn Treebank, must have a tree yield function, even though the above definition of LCFRS only defines the string yield function. For our purposes, the tree yield function can be defined in any way which is consistent with the string yield function.

2.3.2 Parsing

Several parsing algorithms for equivalents to LCFRS have been described in the literature (Seki et al. 1991, Bertsch and Nederhof 2001). We present here, for completeness, a parsing algorithm as a procedure for intersecting a grammar with a finite-state automaton (Lang 1994, Bar-Hillel et al. 1964). The parsing problem is to compute, given an input string w, the derivation forest for w, that is, a compact representation of all derivations of w under a grammar G. Just as we defined feature forests as CFGs above, we can also define the parse forest

as a CFG: a CFG generating all and only the possible derivations of w. Suppose we have a construction which, given a grammar G and a finite-state automaton M, computes a new grammar G' that generates just those derivations of G (up to nonterminal relabeling) which yield strings accepted by M—call this the intersection of G with M. If we then let M be an automaton accepting the singleton language $\{w\}$, the intersection of G with M is a parse forest of w. Thus an algorithm for intersecting G with a finite-state automaton constitutes a parser for G.

We present below an algorithm for computing the intersection of any LCFRS grammar $G = \langle V, S, \mathcal{F}, P\rangle$ with a finite-state automaton $M = \langle Q, \Sigma, \delta, q_0, Q_f\rangle$. Define a new start symbol S' and a new nonterminal alphabet

$$V' = \{S'\} \cup \bigcup_{A \in V} \{A\} \times Q^{2\phi(A)}$$

Intuitively, each nonterminal $A \in V$ is accompanied by $2\phi(A)$ states, indicating the start and stop states of M when recognizing the substrings yielded by A. Next define a new set of productions P': for each production in P, if the production has the form (2.4) and $\bar{f}$ is defined as in (2.5), we replace A_i with

$$\langle A_i, p_{i1}, q_{i1}, \ldots p_{i,\phi(A_i)}, q_{i,\phi(A_i)}\rangle,$$

for all possible values of $p_{ij}, q_{ij} \in Q$, subject to the following constraints (in all the following $v \in \Sigma^*$):

- If vx_{ij} is a prefix of w_k, then $\delta(p_{0k}, v) = p_{ij}$;
- if $x_{ij}v$ is a suffix of w_k, then $\delta(q_{ij}, v) = q_{0k}$;
- if $x_{ij}vx_{kl}$ is a substring of w_m, then $\delta(q_{ij}, v) = p_{kl}$.

Each occurrence of A_i in a derivation yields $\phi(A_i)$ substrings; p_{ij} and q_{ij} are the start and stop states of the jth substring. The purpose of the constraints is to ensure that the substrings are only combined in ways that are licensed by M.

Finally, for every $q \in Q_f$, add to P' the production

$$S' \to \langle S, q_0, q\rangle$$

Then the final result is the generalized CFG

$$G' = \langle V', S', \mathcal{F}, P'\rangle$$

In the case where M is a straight-line automaton accepting a single string, P' has $\mathcal{O}(|G|\, n^{(r+1)f})$ productions, where n is the number of states in Q, f is the maximum value of $\phi(A)$, $A \in V$, and r is the maximum number of nonterminals on the right-hand side of a production. Therefore the running time of this construction is also $\mathcal{O}(|G|\, n^{(r+1)f})$.

This algorithm always runs in polynomial time, but is not necessarily optimal. The grammar may need to be transformed first to achieve the optimal running time. For example, for CFG, we have $f = 1, r \geq 1$, but if we first convert the grammar into Chomsky normal form, we always have $r \leq 2$, giving a running time of $\mathcal{O}(n^3)$. Similarly, for TAG, we have $f = 2, r \geq 1$, but any TAG can be converted into an equivalent TAG with $r \leq 2$ (Lang 1994), giving $\mathcal{O}(n^6)$.

2.3.3 Feature forests

All that remains is to add features to the LCFRSs:

Definition 7 A *generalized CFG (or LCFRS) with features* is a generalized CFG (resp., LCFRS) together with an interpretation function that maps each function symbol f to a composition operation $\bar{f}$ on feature vectors definable as:

$$\bar{f}(\mathbf{x}_1, \ldots \mathbf{x}_n) = \mathbf{x}_f + \sum \mathbf{x}_i \tag{2.6}$$

where $\mathbf{x}_f$ is a feature vector depending on f.

The parsing algorithm for LCFRSs applies equally well to LCFRSs with features, and the resulting generalized CFG with features can be viewed as a feature forest: just treat each production of the form (2.4) as a production $A_0 \rightarrow A_1 \cdots A_n$ with feature vector $\mathbf{x}_f$. This gives, as desired, a procedure for computing feature forests from strings. Together with Miyao and Tsujii's algorithm, this provides a complete method for training a maximum-entropy parsing model.

2.4 Discussion

Thus an LCFRS becomes a formalism for describing maximum-entropy models: the features are defined not on entire parses, but on the elementary structures of the grammar. Therefore features are no longer completely arbitrary, but only *locally* arbitrary, i.e., arbitrary within the local domains defined by the elementary structures of the grammar. The question is, does this provide enough flexibility to be useful for natural language parsing?

2.4.1 Large domains

First of all, there is no fixed limit on the size of local domains. We can extend an existing parsing model by increasing the size of its local domains, widening the range of possible features. Crucially, this move does not require us to eliminate the original features, as would be the case in a generative model; it will make more features available, while retain the original ones if desired.

For example, when we modify a generative TAG model (Chiang 2000) to use elementary TAG trees with multiple lexical anchors instead of only one, accuracy decreases (Chiang 2003), probably because the smoothing method used does not cope well with the larger elementary trees. But a model that allows locally arbitrary features could circumvent this difficulty. For example, if the elementary trees for PPs were fused with their objects, there could be a feature for the dependency between the preposition and the modified headword and a feature for the dependency between the preposition and the object headword, just as in the original model, but also a feature involving the prepositional object headword and the modified headword, or all three words.

One practical issue that would arise from enlarging the elementary structures would be *grammar* sparseness: if the parser's grammar is obtained by fragmenting trees from a corpus (e.g., Chiang 2000, Hockenmaier and Steedman 2002), then larger structures will be less likely to recur in new data. This problem already occurs with lexicalized grammars, whose coverage can be improved by recombining lexical anchors and structures which have the same part-of-speech tag. An analogous solution for even larger elementary structures would be to define additional recombination points (for example, every S or NP node). Of course, the increase in the effective size of the grammar will also increase parsing time.

2.4.2 Overlapping domains

Local domains may also overlap by a finite amount. For example, in a CFG two adjacent local domains have a nonterminal symbol in common. By means of grammar transformations, one can use this overlap as a conduit through which information can be propagated between local domains, effectively extending the domains of features. This is the technique lexicalized PCFG parsers use, for example: they propagate headwords up through the tree to capture bilexical dependencies. We note two consequences of this observation.

First, it is possible to "squeeze" strong generative capacity out of a grammar formalism without increasing its weak generative capacity (Joshi 2003), and this extra power can be demonstrated formally in certain respects (Chiang 2002); for example, regular form TAG (Rogers 1994) is weakly equivalent to CFG, but its synchronous version generates more string relations than synchronous CFG. It might be thought that something similar could be shown for feature forests, that a feature forest based, for example, on regular form TAG allows more definable features than one based on CFG. But this is not true in general. Often a grammar G of a "squeezed" formalism can be converted into a gram-

mar G' of the "unsqueezed" formalism such that the derivations by G can be recovered from the derivations by G'.

Definition 8 We say that a grammar G' of an LCFRS *simulates*[2] a grammar G if there is a bijection $[\![\cdot]\!]_s$ from the terms of G' to the terms of G such that $[\![[\![t]\!]_s]\!]_G = [\![t]\!]_{G'}$ for all terms t of G'.

If $[\![\cdot]\!]_s$ is compositional and "linear" by analogy with Definition 6, that is, each of its composition operations introduces a fixed set of function symbols and only rearranges existing symbols without copying or erasing, then it is easy to show that G' can define all the features that G can. This is because as G' simulates G, it can emit the same function symbols (and therefore the same feature vectors) that G' would in its derivation, though possibly in a very different order; but here order does not matter because addition of feature vectors is commutative. Therefore G' can define any feature that G can.

This is the case with regular form TAG (as defined by Chiang (2001)) relative to CFG, and tree-local multicomponent TAG (Weir 1988) relative to TAG. (Tree insertion grammar (TIG, Schabes and Waters 1995) can be simulated by CFG, but the simulation only preserves strings, not trees.) We expect this relationship to obtain whenever a grammar formalism can be parsed using an algorithm similar to that of a simpler formalism, because such parsers typically work by computing $[\![\cdot]\!]_s$.[3] So although this type of "squeezing" may allow more felicitous descriptions of parsing models, it does not actually allow any new models to be described.

On the other hand, this also means that one can model a wide range of features without even going beyond CFG. Any decision that can be made by a finite-state tree automaton can be made into a feature that is local with respect to CFG. Nearly all the features of Collins' discriminative reranker, for example, fall into this class.[4] What's more, since finite-state tree automata are closed under intersection, they can be freely combined. The blow-up in the size of the grammar, however, may be considerable.

[2]This definition is somewhat broader than the original definition (Chiang 2001).

[3]The Lambek calculus would be an example of a weakly context-free formalism with no CFG-like parsing algorithm (Pentus 1993, 2003)

[4]The distance features cannot be encoded by a tree automaton. However, one of the features, which fires when the distance between a headword and its modifier is not greater than some threshold, could be so encoded, because the threshold is limited to 9.

2.4.3 Interleaving domains

Local domains also need not be contiguous regions of the parse tree, because parts of a local domain may end up in very distant parts of the parse tree. (For example, nodes of a TAG or TIG elementary tree get stretched apart when an auxiliary tree adjoins between them.) The parameter f above (Section 2.3.2) controls how many discontinuous substrings the yield of a subderivation may have. Since parsing time depends on this parameter, there is a tradeoff between interleaving of local domains and parsing efficiency.

But the existence of wide-coverage TAGs for English and French suggests that many linguistically-motivated features do not require $f > 2$. Moving from CFG to TAG (or just to TIG) would make it possible to define features capturing nonlocal dependencies, like the dependency between the subject and embedded verb in a raising or control construction ("The *cat* wants to *sit* on the mat"), or between the verb in a relative clause and the noun modified by the relative clause ("the *cat* which *sat* on the mat"). Moving from a less powerful formalism to a more powerful one would make more features available, without affecting the original ones.

One might also consider extending further to multicomponent TAG. The standard TAG extraction method uses head rules to propagate headwords through each training example, then defines local domains such that two nodes are in the same local domain if and only if they have the same headword. This method extends straightforwardly to noncontiguous local domains. For example, one could preserve the empty elements in the Penn Treebank and assign them to the same local domains as the nodes with which they are coindexed. The extracted multicomponent TAG would localize filler-gap dependencies and could be used to reconstruct them.

2.4.4 Comparison with Maxwell-Kaplan packed representations

Geman and Johnson (2002) present a similar method for parsing and estimation of stochastic unification-based grammars. Like the present method, it uses packed representations of sets of parses. The representation they use comes from earlier work by Maxwell and Kaplan (1995), which we define briefly here. In this view, a parse $T \in \mathcal{T}$ is thought of as a set of *parse-features* (called simply "features" in their terminology).

Definition 9 A *Maxwell-Kaplan packed representation* (or MKPR) is a tuple $\langle \mathcal{P}, \mathcal{X}, N, \alpha \rangle$, where

- $\mathcal{P}$ is a finite set of possible parse-features,

- $\mathbf{X}$ is a finite vector of variables, each variable X_i ranging over a finite set $\mathcal{X}_i$,
- N is a finite set of conditions on $\mathbf{X}$, and
- α is a function mapping each parse-feature $p \in \mathcal{P}$ to a condition (not necessarily in N) on $\mathbf{X}$.

Definition 10 We say that an assignment of values to $\mathbf{X}$ *identifies* a parse T if it satisfies the conditions (given by α) of the parse-features in T, and no others.

Definition 11 We say that an MKPR R *represents* a set $\mathcal{T}$ of parses if there is a one-to-one correspondence between assignments of $\mathbf{X}$ satisfying N and parses in $\mathcal{T}$ such that the parse identified by an assignment is equal to the corresponding parse.

As with feature forests, Geman and Johnson's method also imposes a notion of locality on features: the feature vector ("property vector," in their terminology) of a parse is the sum of the feature vectors of its parse-features.

Definition 12 Let $\mathbf{f}$ be a mapping from parse-features to feature vectors. Then the feature vector of a parse T is $\sum_{p \in T} \mathbf{f}(p)$.

Any feature forest F can be converted into an MKPR. Let V be the nonterminal alphabet of F, and P the productions of F. First we must assume that no nonterminal symbol can be used twice in a single derivation. If this is not the case, then the offending nonterminals must first be renamed apart, duplicating productions as necessary (see below for analysis).

The parse-features are just the productions in P, and each nonterminal symbol $A \in V$ corresponds to a variable X_A, ranging over the set $\{\beta \mid A \to \beta \in P\} \cup \{*\}$. Each X_A gives the right-hand side with which A gets rewritten ($*$ means that A does not get used at all). Next we define the conditions on $\mathbf{X}$. Let α assign to the parse-feature $A \to \beta$ the condition that $X_A = \beta$. The conditions in N ensure that $X_S \neq *$, and that for each production $A \to \beta$ and each nonterminal B occurring in β, $X_B \neq *$ if and only if $X_A = \beta$.

How does the size of the resulting MKPR compare with the original feature forest? Ordinarily it should be the same, but we noted above that if a nonterminal symbol of the feature forest can occur twice in a single derivation, the two occurrences must be renamed apart, duplicating productions as necessary. This can only happen in two cases. Let $A' = \langle A, p_1, q_1, \ldots, p_{\phi(A)}, q_{\phi(A)} \rangle$ stand for a nonterminal of the feature forest (as produced by the construction of Section 2.3.2). Then the two cases are when $A' \stackrel{*}{\Rightarrow} \cdots A' \cdots$, which is already ruled out by the

assumption that the feature forest has a finite number of derivations, or when $S' \stackrel{*}{\Rightarrow} \cdots A' \cdots A' \cdots$, which is possible in a parse forest only if $p_i = q_i$ for all i, that is, the two occurrences of A have the same empty span. So in most cases the MKPR will be the same size as the feature forest, though in the worst case, the blowup could be arbitrarily large.

How does the time complexity of processing the resulting MKPR compare with processing the original feature forest? Geman and Johnson use graphical techniques to perform computations on MKPRs, the complexity of which is a function of the pattern of dependencies between the X_A variables in the conditions. For MKPRs produced by the above construction, for each production $A \to \beta$ there is a dependency between X_A and X_B, where B occurs in β. If every nonterminal appears only once among the right-hand sides (i.e., there is no subderivation sharing), then the dependency graph will be a tree, the most efficient case. But subderivation sharing is to be expected; indeed, dynamic programming would not provide any benefit if there were none. So Geman and Johnson's algorithm will not, in general, achieve its optimal case. By contrast, Miyao and Tsujii's algorithm, because it takes advantage of the particular structure of feature forests, always matches the optimal case of Geman and Johnson's algorithm.

2.4.5 Extension to range-concatenation grammars

Most of the ideas presented here extend straightforwardly to range-concatenation grammars, or RCGs (Boullier 2000),[5] which are more powerful still than LCFRSs, generating exactly the languages which are recognizable in polynomial time (Bertsch and Nederhof 2001). Because the derivations forests of RCGs can be thought of as context-free grammars, just as with LCFRSs, the method described above of attaching feature vectors to productions and estimating parameters applies equally well to RCGs as to LCFRSs.

The class of range-concatenation languages has the convenient property of being closed under intersection. A negative consequence of this is that it does not seem possible in general to define generative statistical models for RCGs. The method described here circumvents this problem, providing a novel way of defining statistical RCGs. Moreover, the ability of RCGs to superimpose different analyses via intersection would seem to be ideal for defining models with overlapping features. However, the fact that this intersection ability does not carry over straightforwardly to trees may reduce its usefulness for defining maximum-entropy parsing models. Further investigation into such models is necessary.

[5] Thanks to an anonymous reviewer for this suggestion.

2.5 Conclusion

We have described how to integrate Miyao and Tsujii's algorithm into a full system for training on treebank data and parsing new sentences, and discussed the advantages and limitations of this approach relative to history-based models and relative to Geman and Johnson's approach. We have shown how the descriptive power of a formalism correlates with its flexibility for specifying parsing models; it remains to be seen whether more powerful formalisms can give rise to more accurate parsing models in practice.

Acknowledgements

This research was supported in part by NSF grants SBR-89-20230 and ITR EIA-02-05456. I would like to thank Fernando Pereira, William Schuler, Dan Bikel, Aravind Joshi, and the anonymous reviewers for their valuable help. *S. D. G.*

References

Abney, S. P. 1997. Stochastic attribute-value grammars. *Computational Linguistics* 23(4):597–618.

Baker, J. K. 1979. Trainable grammars for speech recognition. In *Proceedings of the Spring Conference of the Acoustical Society of America*, pages 547–550.

Bar-Hillel, Y., M. Perles, and E. Shamir. 1964. On formal properties of simple phrase structure grammars. In Y. Bar-Hillel, ed., *Language and Information: Selected Essays on their Theory and Application*, pages 116–150. Addison-Wesley.

Berger, A. L., S. A. Della Pietra, and V. J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics* 22(1):39–71.

Bertsch, E. and M.-J. Nederhof. 2001. On the complexity of some extensions of RCG parsing. In *Proceedings of the 7th International Workshop on Parsing Technologies*, pages 66–77. Beijing.

Boullier, P. 2000. Range concatenation grammars. In *Proceedings of the 6th International Workshop on Parsing Technologies*, pages 53–64. Trento.

Chiang, D. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 456–463. Hong Kong.

Chiang, D. 2001. Constraints on strong generative power. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics*, pages 124–131. Toulouse.

Chiang, D. 2002. Putting some weakly context-free formalisms in order. In *Proceedings of the 6th International Workshop on TAG and Related Formalisms*, pages 11–18.

Chiang, D. 2003. Statistical parsing with an automatically extracted tree adjoining grammar. In R. Bod, R. Scha, and K. Sima'an, eds., *Data-Oriented Parsing*, pages 299–316. Palo Alto: CSLI Publications.

Collins, M. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Collins, M. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the 17th International Conference on Machine Learning*, pages 175–182. San Mateo: Morgan Kaufmann.

Geman, S. and M. Johnson. 2002. Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 279–286. Philadelphia, PA.

Goodman, J. 1999. Semiring parsing. *Computational Linguistics* 25:573–605.

Hockenmaier, J. and M. Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 1974–1981.

Johnson, M., S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 535–541. College Park, MD.

Joshi, A. K. 2003. A note on the strong and weak generative powers of formal systems. *Theoretical Computer Science* 293:243–259. Originally presented at TAG+5 in 2000.

Lafferty, J., A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*. San Mateo: Morgan Kaufmann.

Lang, B. 1994. Recognition can be harder than parsing. *Computational Intelligence* 10(4):484–494. Special Issue on Tree Adjoining Grammars.

Lari, K. and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language* 4:35–56.

Malouf, R. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning*, pages 49–55.

Maxwell, J. T., III and R. M. Kaplan. 1995. A method for disjunctive constraint satisfaction. In M. Dalrymple, R. M. Kaplan, J. T. Maxwell, III, and A. Zaenen, eds., *Formal Issues in Lexical-Functional Grammar*. Palo Alto: CSLI Publications.

Miyao, Y. and J. Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 292–297. San Diego.

Pentus, M. 1993. Lambek grammars are context-free. In *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science*, pages 429–433. Los Alamitos.

Pentus, M. 2003. Lambek calculus is NP-complete. Technical Report TR-2003005, CUNY Ph.D. Program in Computer Science.

Ratnaparkhi, A. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP-2)*. Providence, RI.

Rogers, J. 1994. Capturing CFLs with tree adjoining grammars. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 155–162. Las Cruces, NM.

Schabes, Y. and R. C. Waters. 1995. Tree insertion grammar: a cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics* 21:479–513.

Seki, H., T. Matsumura, M. Fujii, and T. Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88:191–229.

Vijay-Shanker, K., D. Weir, and A. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 104–111. Stanford.

Weir, D. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania.

3

Automating the Generation of a Wide-coverage LFG for French using a MetaGrammar

Lionel Clément and Alexandra Kinyon

[1]In this paper, we explain how the notion of MetaGrammar, which has successfully been used for generating wide-coverage tree adjoining grammars (TAGs) for various languages such as French (Abeillé et al. (1999)) and German (Gerdes (2002)), may be used to generate a wide-coverage Lexical Functional Grammar (LFG) for French. We first introduce the notion of MetaGrammar and present the tools we use to automatically generate, from a compact MetaGrammar hierarchy, a LFG for French: the grammar writer specifies in compact manner syntactic properties that are usually encoded separately in the LFG machinery in the lexicon, in lexical rules and in rewriting rules. From this hierarchy an LFG is automatically generated. We also propose an augmentation of the existing MetaGrammar tool.

3.1 Introduction

Expensive dedicated tools and resources (e.g., grammars, parsers, lexicons) have been developed for a variety of grammar formalisms - which all have the same goal - model the syntactic properties of natural language, but resort to a different machinery to achieve that goal. However, there are some core syntactic phenomena on which a cross-framework

[1]We thank Eric de la Clergerie, Julia Hockenmaier, Aravind Joshi, Owen Rambow, Pierre Boullier and three anonymous reviewers for helpful comments on earlier versions of this work.

Proceedings of FGVienna: The 8th Conference on Formal Grammar.
Gerald Penn (ed.).
Copyright © 2008, CSLI Publications.

(and to some extent a cross-language) consensus exists, such as the notions of subcategorization, valency alternations, and the notion of syntactic function. From a theoretical perspective, a MetaGrammatical level of representation allows one to encode such consensual pieces of syntactic knowledge and to compare different frameworks. From a practical perspective, encoding syntactic phenomena at a MetaGrammatical level, from which grammars in different frameworks can be generated, has several advantages such as portability among grammatical frameworks, increased coherence and consistency in the generated grammars and less need for human intervention in the grammar development process. Here, we focus on the use of MetaGrammars to automate the generation of a wide-coverage LFG for French.

In the first part of this paper, we explain the notion of *MetaGrammar* and present the MetaGrammar tool we have used. In the second part of this paper, we justify the use of a MetaGrammar for generating LFGs and explore several options for doing so. In the third part of this paper we discuss the coverage of the grammar we generate as well as the advantages of a MetaGrammar approach. We assume the reader is familiar with the LFG framework and refer to (Bresnan and Kaplan (1982)) for an introduction.

3.2 What is a MetaGrammar?

The notion of MetaGrammar (MG) was originally presented in (Candito (1996)) to automatically generate a Tree Adjoining Grammar (TAG)[2] for French and Italian.[3] The idea is to add a higher-level of linguistic description to encode, partial descriptions of trees. The general organization for syntactic information is defined in three multiple inheritance networks. Properties are shared among different trees in the grammar thanks to the multiple inheritance mechanism.[4]

The first hierarchy encodes canonical lexical subcategorizations (i.e., transitive, ditransitive), the second encodes valency alternations and redistribution of syntactic functions (e.g., it allows to add an argument for causatives, to erase one for passive with no agents, etc.), and the third encodes the surface realization of arguments (e.g., declares if a direct-object is pronominalized, wh-extracted, etc.).

[2]We do not present the TAG formalism here. Suffice it to say that it is a tree rewriting system, hence a TAG rule is a tree which may be of depth greater than one, unlike LFG rewriting rules.

[3]A Similar MetaGrammar type of organization for TAGs was independently presented in (Xia (2001)) for English.

[4]The main difference with HPSG hierarchical organization (e.g. Flickinger (1987)) is that HPSG type hierarchies are an inherent part of the HPSG framework, whereas the MetaGrammar specifications are framework-independent.

A well-formed TAG elementary tree is generated by inheriting from exactly one initial subcategorization, one redistribution (which can be complex e.g., Passive + Causative), and for each argument, from one realization. From a linguistic perspective, predicate-Argument Co-occurrence Principles are enforced by the fact that the tool automatically crosses the constraints between the three dimensional networks.[5]

For instance, the elementary tree for "Par qui sera accompagnée Marie" (*By whom will Mary be accompanied*) is generated by inheriting from **transitive** in dimension 1, from **passive** in dimension 2 and **subject-nominal-inverted** for its subject and **Wh-questioned-object** for its object in dimension 3.

3.2.1 HyperTags

A grammar rule is associated to each grammar rule which is automatically generated from the MetaGrammar hierarchy. The main idea behind HyperTags, which is a notion first introduced in (Kinyon (2000)), was to keep track, when trees (i.e., TAG grammar rules) are generated from a MetaGrammar hierarchy, of which terminal classes were used for generating the tree. This allows one to obtain a framework-independent feature structure containing the salient syntactic characteristics of each grammar rule.[6] For instance, the verb *give* in *A book was given to May* could be assigned the HyperTag:

$$\begin{bmatrix} \text{Subcat} & \text{Ditransitive} \\ \text{Valency alternations} & \text{Passive with no agent} \\ \text{Argument Realization} & \begin{bmatrix} \text{Subject:} & \text{Canonical NP} \\ \text{Object:} & \text{Not realized} \\ \text{By-Phrase:} & \text{Canonical PP} \end{bmatrix} \end{bmatrix}$$

Although we retain the linguistic insights presented in (Candito (1996)), that is the three dimensions to model syntax (subcategorization, valency alternation, realization of syntactic arguments) we use a different MetaGrammar tool which, although it was also originally designed to generate TAGs, is less framework-dependent and supports the notion of HyperTag.

[5] So, in Candito's implementation, classes cross by default.

[6] The notion of HyperTag was inspired by that of supertags presented in (Srinivas (1997)), which consists in assigning a TAG elementary tree to lexical items, hence enriching traditional POS tagging. However, HyperTags are framework-independent.

3.2.2 The LORIA MetaGrammar compiler

To generate LFGs, we use a modified version of the LORIA MG compiler presented in (Gaiffe et al. (2002)).[7] In the LORIA tool, each class in the MG hierarchy encodes:

- Its Super-class(es)
- A HyperTag which captures the salient linguistic characteristics of that class
- What the class needs and provides
- A set of quasi-nodes (i.e., variables)
- Topological relations between these nodes (*parent, dominates, precedes, equals*).
- A function for each quasi-nodes to decorate the tree (e.g., traditional agreement equations and/or LFG functional equations).

Each class in the hierarchy encodes properties of its own (such as parts of decorated trees, what the class needs and provides, etc.), and also inherits properties from its super-classes. The MG tool automatically crosses terminal classes in the hierarchy[8], looking to create *balanced* classes, that is classes which do not need nor provide any resource. Then for each balanced class, their HyperTags and the structural constraints between their quasi-nodes are unified. If the unification succeeds, one or more <HyperTag, tree> pairs are generated. Figure 1 illustrate how a simple TAG rule is generated with this tool.

Contrary to Candito's MetaGrammar, the LORIA MetaGrammar compiler explicitly describes how the classes cross. There are no general constraints such as **completeness** and **coherence** to construct well-formed objects. So the LORIA tool is more flexible in the sense that there is no explicit notion of *dimension*: it is possible to construct a MetaGrammar in a way which differs from the three-dimensional framework based on functional complementation. Instead, for making classes cross, one must resort to the resource allocation model (i.e. one must explicitly state what each class needs and provides) in order to obtain an adequate crossing between terminal classes in the hierarchy (e.g. to ensure an appropriate complementation, a correct ordering between constituents and other syntactic properties).[9]

[7] The LORIA compiler is freely available on http://www.loria.fr/equipes/led/outils/mgc/mgc.html

[8] Terminal classes are classes which do not have any subclass

[9] Another way to rephrase this is to say that, in Candito's MetaGrammar tool, classes cross by default, whereas with the LORIA tool, the default is for a class not to cross unless one of its resource needs to be satisfied.

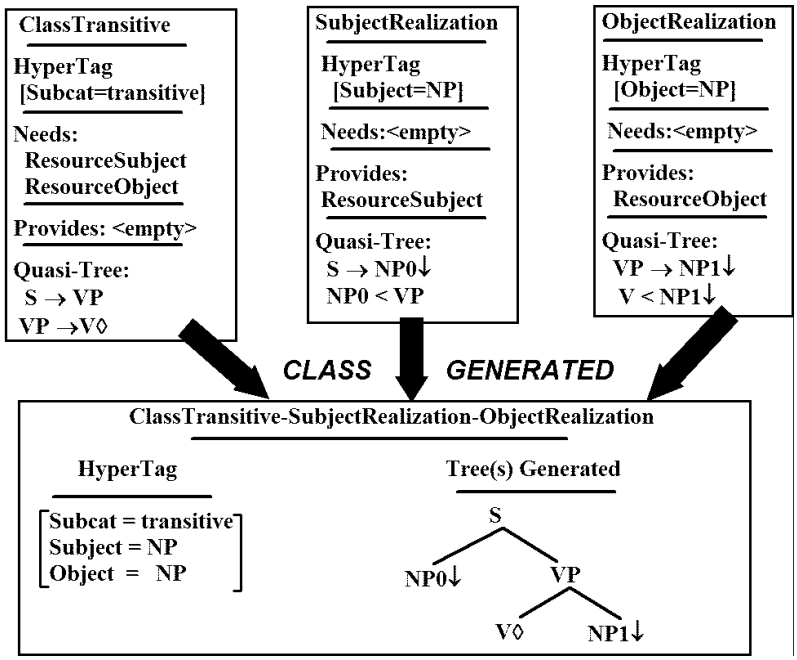


FIGURE 1 Generation of a simple TAG rule → stands for father, < for precedes,◇ for a lexical anchor and ↓ for substitution node in the TAG terminology

3.3 Generating LFGs with a MetaGrammar

3.3.1 Why use a MetaGrammar for LFGs ?

Because TAGs are a tree rewriting system, there are intrinsic redundancies in the rules of a TAG. E.g., all the rules for verbs with a canonical NP subject and a canonical realization of the verb will have a redundant piece of structure
(S NP0↓ (VP (V◇))). This piece of structure will be present not only for each new subcategorization frame (intransitive, transitive, ditransitive,...), but also for all related non-canonical syntactic constructions such as in each grammar rule encoding a Wh-extracted object. This redundancy justifies the use of a MetaGrammar for TAGs. Since LFG rules rely on a context free backbone, it is generally admitted that there is less redundancy in LFG rules than in TAG rules. However, there are still redundancies, at the level of rewriting rules, at the level of functional equations, and at the level of lexical entries. For instance, a rule such as the one shown on figure 2 could be written for the VP expansion

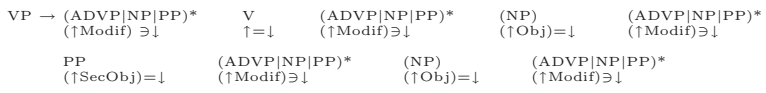


FIGURE 2 Redundant VP expansion for French ditransitives

of a ditransitive verb for French, with the free insertion of a modifier anywhere in the right hand-side of the rule. We see that the modifier is inserted 5 times, with its corresponding functional equation, and the NP for the direct object is repeated twice.

So, in addition of inherent advantages of a MG approach (independently of the grammar framework one chooses to generate), LFG has redundancies which justify the use of a MG for this particular framework. LFG redundancies and a comparison of the MG to the use of *LFG specific* operators such as *shuffle* are discussed in more detail in (Clément and Kinyon (2003a)) and (Clément and Kinyon (2003b)): the main distinction between our MG approach and the use of operators is that the proliferation of operators (e.g. shuffle, ID/LP rules, macros etc.) makes it possible to express the same rule in many different ways, which may be helpful for grammar writing purpose, but not so much for maintenance purpose.[10] By contrast, the MetaGrammar outputs very *homogeneous* grammars: even though different choices can be made at the MetaGrammatical level (e.g. should we resort to VP nodes?, should these VPs be flat or right branching? etc.), once a choice is made, it applies homogeneously throughout the whole generated grammar.

3.3.2 Generating LFGs

We have seen in section 3.2.2 that the MetaGrammar tool we use outputs <HyperTag, tree> pairs. When generating TAGs, *tree* is a grammar rule (i.e., a TAG elementary tree). We extend the approach to LFG by generating <HyperTag, tree> pairs, where *tree* is a constituent structure decorated with functional equations and corresponds to one or more LFG rewriting rules, as shown in figure 5. Note that this decorated tree is obtained in exactly the same manner as its TAG counterpart from fig 1. In addition to generating rewriting rules, our MetaGrammar hierarchy also yields lexical templates.[11] This is illustrated in figure 3 and figure 4: a decorated tree is generated automatically from our MG hierarchy (3). This decorated tree yields one LFG rewriting rule and one lexical template (4) for a French verb such as

[10] An analogy can be made to computer programs written in Perl.

[11] A lexical template encodes a linguistic generalization (such as a subcategorization frame) shared by a set of lexical entries

"éloigner" (*take away from*), which subcategorizes for an NP object and for a PP object introduced by "de". (Ex: "Peter éloigne son enfant de la fenêtre"-*Peter takes his child away from the window*).

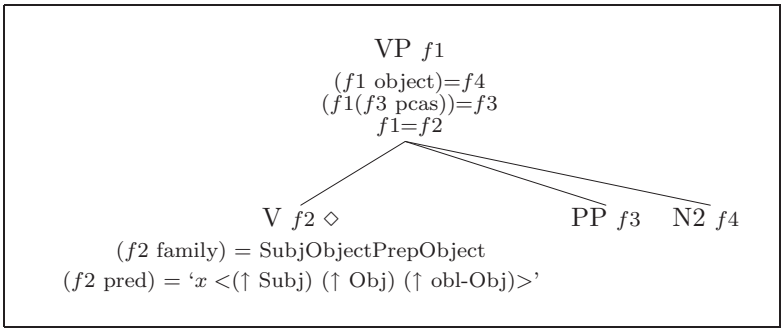


FIGURE 3 Decorated constituent tree for a simple clause

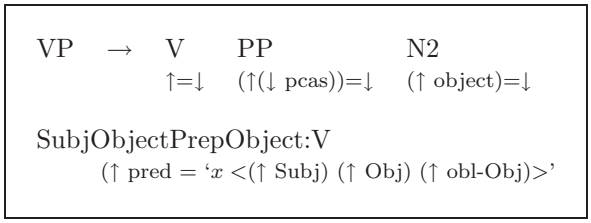


FIGURE 4 LFG Rule and a lexical template

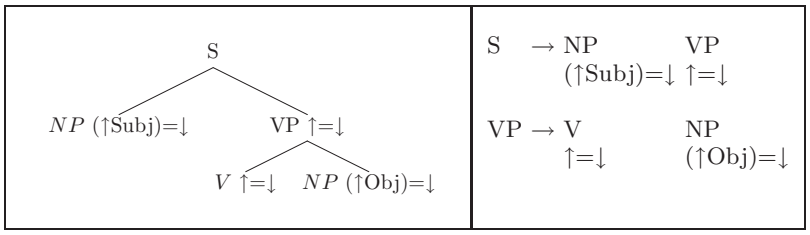


FIGURE 5 Generating a decorated tree which yields two simple LFG rules

As discussed in (Joshi and Vijay-Shanker (1989)), an essential difference between TAGs and LFGs resides in the domain of locality of the grammar rules. The MG does not impose a given domain of locality.

Hence, in order to generate LFG rules with a MG, we have two options. The first option consists in generating *standard* LFG rules, that is trees of depth 1 decorated with functional equations (as shown in fig 3): the root of the tree of depth one corresponds to the left hand-side of an LFG rewriting rule,[12] the children of this root correspond to the right hand-side of the rule. The second option, which is the one we have opted for (illustrated in fig 5 and 6), consists in generating constituent trees which may be of depth superior to one, decorated with feature equations. Namely, we adopt a domain of locality that is at the *clause* level, which has the following advantages:

- It allows for a more natural parallelism between the LFG grammars generated and other approaches.[13]
- It allows for a more natural encoding of syntax at the MetaGrammar level.
- It allows us to generate LFGs without resorting to LFG Lexical Rules.

The trees automatically generated from the MetaGrammar hierarchy, are decorated with LFG functional annotations, as was first proposed by (Kameyama (1986)), and in a way which is similar to that presented in (Frank (2000)). These decorated trees are then decomposed into standard LFG rewriting rules (similarly to the work presented in Hepple and van Genabith (2000)), and into lexical templates. For modifiers, we have implemented right branching adjuncts (e.g. non-flat VPs for free modifier insertion, i.e. unlike the VP rule shown in 2.).[14]

In a first step, for each decorated tree we generate, each subtree of depth one produces one LFG rewriting rule. In a second step, equivalent rules are merged. Equivalent rules are defined by induction: Two rules are equivalent if and only if they differ only on equivalent non-terminal symbols with identical functional equations. The grammar we obtain is then interfaced with a parser.[15]

The *resource model* of the MetaGrammar, based on *needs* and *provides*, allows for a natural encoding and enforcement of LFG coherence, completeness and uniqueness principles: A transitive verb needs exactly

[12] Modulo the slight difference that the root may be decorated, contrary to standard left hand-sides of LFG rules

[13] Although we do not develop this point here, we have used our MetaGrammar hierarchy to generate grammars in frameworks other than LFG, esp. TAGs and Range Concatenation Grammars (Boullier (1998))

[14] As an alternative, (Kinyon (2003)) resorts to flat VPs with a MG hierarchy for free modifier insertion between each constituent in the VP, except the clitics.

[15] We use the freely available XLFG parser described in (Clément and Kinyon (2001)) and have experimented with the LFG parser described in (Kaplan and Maxwell (1996)).

one resource *Subject* and one resource *Object*. Violations result in invalid classes which do not yield any rules. So from that perspective, it makes little sense (apart from practical reasons, such as interfacing the grammar with an existing parser) to force the rules generated to be trees of depth one. Moreover, the same type of resource-sensitivity is used within the LFG community. e.g. in (Dalrymple et al. (1995)) to compute semantic forms using linear logic.

Concerning lexical rules, traditional LFGs encode phrase structure realizations of syntactic functions such as the wh-extraction or pronominalization of an object in phrase structure rules. In the MetaGrammar, these are encoded in the *Argument Realization* dimension (dimension 3 in Candito's terminology). For valency alternations, i.e., when initial syntactic functions are modified, LFG resorts to the additional machinery of lexical rules.[16] However, these valency alternations are encoded directly in the MetaGrammar in the *valency alternation* dimension (dimension 2 in Candito's terminology). Hence, when a rule is generated for a canonical transitive verb, rules are generated not only for all possible argument realization for the subject and direct object (wh-questioned, relativized, cliticized for French etc.), but also for all the valency alternations allowed for the subcategory frame concerned (here, passive with/without agent, causative etc) in a monotonic manner. Therefore, there is no need to generate usual LFG lexical rules, and the absence of lexical rules has not effect on interfacing the grammars we generate with existing LFG parsers.

3.3.3 Yet another MetaGrammar compiler

In order to implement some additions to the MetaGrammar formalism, we have developed our own tool. In our compiler, it is possible to describe topological relations between variables (encoded as logical constraints), even if these variables are not instantiated. This feature helps to describe in an abstract manner some syntactic phenomena such as clitic ordering in French, without worrying about whether a given phenomenon will appear in a grammar rule. More precisely, for clitics, one encodes in a single class ordering constraints for the realization of nominative, accusative, dative, genitive clitics[17], and from these abstract specifications, one obtains several rules with one, two, three or more clitics realized. By contrast, the LORIA MG tool only allows one to formulate constraints on elements which are going to be realized in a rule, so if one writes a class for ordering 3 clitics (e.g. nominative,

[16] Although some versions of LFG dispense with lexical rules altogether.

[17] The ordering rules are quite complex and depend on the person and case of the clitic, as well as on the class of verb to which the clitic will attach

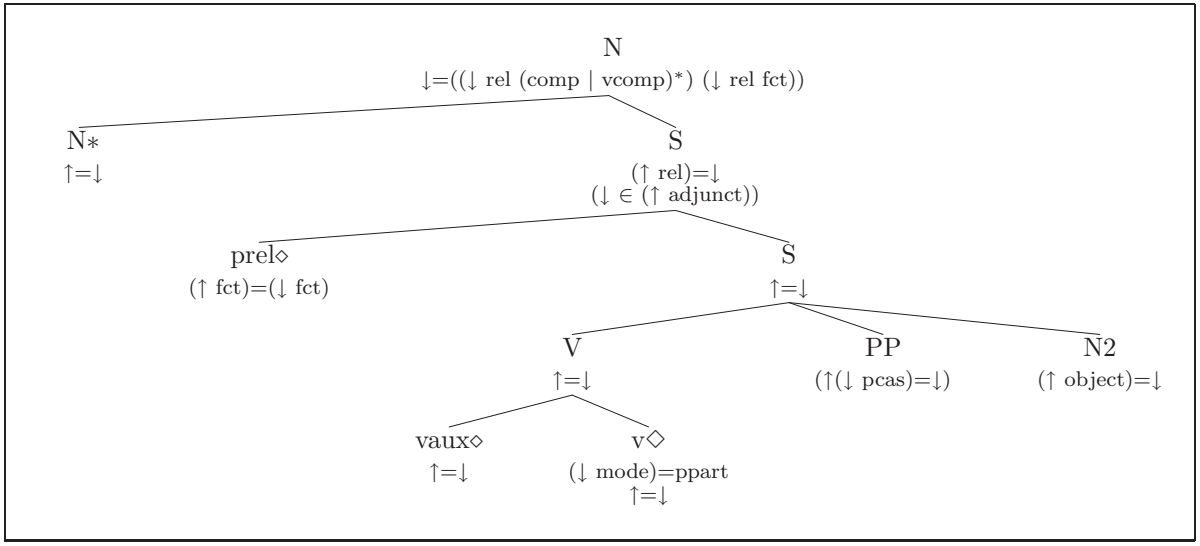


FIGURE 6 Generated tree (partial representation) for a simple relative clause with a ditransitive verb, decorated with LFG functional notation.

accusative, dative), one will generate rules realizing 3 cliticized arguments such as the subject, the object, and the second-object (ex: "Il le lui donne" - lit: *he:nom it:acc him:dat gives*), but will not be able to generate a rule where only two arguments are cliticized (ex: "Il lui donne une pomme" - lit: *He him:dat gives an apple*). This means that with the LORIA tool, one has to encode a class for each possible combination of clitic(s). We have also enriched the MetaGrammar compiler with the notion of *optional resource*. The idea is that, in some cases, one wants a class to be able to cross, or not cross, with some other class. For instance, a verb realization may require a negation, but it also may not. With the LORIA tool, both cases have to be listed e.g. by having a class *standardVerb* and a class *negatedVerb*, Our notion of *optional resource* prevents redundancy at the MetaGrammar level and therefore allows for an even more compact encoding of syntactic knowledge.

3.4 Advantages of a MetaGrammatical level

A first advantage of using a MetaGrammar, as is argued in (Kinyon and Prolo (2002))[18] is that the syntactic phenomena covered are quite systematic: if rules are generated for "transitive-passive-whExtractedByPhrase" (e.g., *By whom was the mouse eaten*), and if the hierarchy includes ditransitive verbs, then the automatic crossing of phenomena ensures that sentences will be generated for "ditransitive-passive-whExtractedByPhrase" (i.e., *By whom was Peter given a present*). All rules for word order variations are automatically generated by underspecifying relations between quasi-nodes in the MetaGrammar hierarchy. For example, the precedence relation between direct object (NP) and second object (PP) is left unspecified to allow both:

"Il donne une pomme à Marie" (*lit: He gives an apple to Mary*)
"Il donne à Marie une pomme" (*lit: He gives to Mary an apple*)

A second advantage of the MetaGrammar is to minimize the need for human intervention in the grammar development process. The grammar writer encodes the linguistic knowledge in a compact manner i.e., the MG hierarchy, and then verifies the validity of the rules generated. If some grammar rules are missing or incorrect, then changes are made directly in the MG hierarchy and never in the generated rules. This ensures a homogeneity not necessarily present with traditional handcrafted grammars. A third advantage is that it is straightforward to obtain from a single hierarchy parallel multi-lingual grammars such as the parallel LFG grammars presented in (Butt et al. (1999)) and (Butt

[18]They also contrast MG and MetaRule approaches: the main difference is that MG are, unlike MetaRules, monotonic.

et al. (2002)), but with an explicit sharing of classes in the MetaGrammar hierarchy: (Kinyon and Rambow (2003b)) have used the tool to generate from a single hierarchy cross-framework and cross-language annotated test-suites, including English and German sentences annotated for F-structure, as well as for constituent and dependency structure.[19]

3.5 Coverage of the grammar

So far, we have implemented a non trivial hierarchy which consists in 189 classes. A fragment of the hierarchy is shown in Figure 7. From this hierarchy, we generate 550 trees decorated with functional annotation, which are then decomposed and compacted into 140 standard LFG rewriting rules. We cover the following syntactic phenomena: 50 verb subcategorization frames (including auxiliaries, modals, sentential and infinitival complements), clitics (and their placement), passives with and without agent, long distance dependencies (relatives, wh-questions, clefts) and a few idiomatic expressions. We handle most of the simple Eurotra sentences taken from (Danlos and Laurens (1991)): We have chosen those sentences because we had an appropriate lexical coverage. The grammar is under constant development, so all the numbers provided in this section are approximations, and prone to change over time. Here are some examples of constructions handled by our grammar:

Relatives

- L'ingénieur qui brigue le poste ... *(lit: The engineer who wants the position...)*
- L'ingénieur à qui l'on propose le poste ...*(lit: The engineer to whom one offers the position ...)*
- L'ingénieur à qui l'on pense proposer le poste *(lit: The engineer to whom one thinks about offering the position to ...)*
- L'ingénieur à qui Marcel pense que la DRH proposera le poste *(lit:The engineer to whom Marcel thinks the manager will offer the position ...)*

Topicalization

- C'est l'ingénieur qui brigue le poste *(lit:It is the engineer who wants the position)*
- C'est l'ingénieur à qui l'on propose le poste *(lit:It is the engineer to whom one offers the position)*
- C'est l'ingénieur à qui l'on pense proposer le poste *(lit:It is the engineer to whom one considers offering the position to)*

[19]The cross-language application of the MG for LFG is further discussed in (Kinyon and Rambow (2003a)).

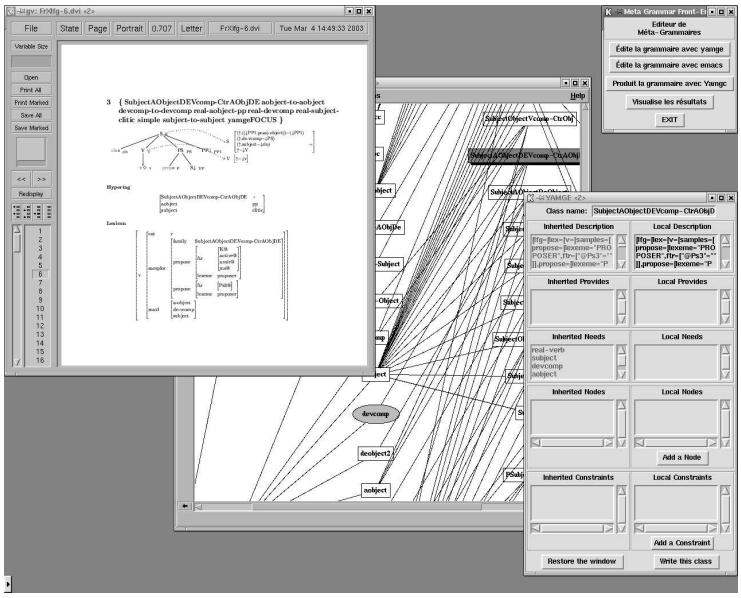


FIGURE 7 Screen capture of a fragment of our MetaGrammar hierarchy

- C'est l'ingénieur à qui Marcel pense que la DRH proposera le poste *(lit:It is the engineer to whom Marcel believes that the manager will offer the position)*

Wh-Questions

- Qui brigue le poste ? *(Who wants the position?)*
- A qui propose-t-on le poste ? *(To whom does one offer the position?)*
- A qui pense-t-on proposer le poste ? *(To whom does one think about offering the position?)*
- A qui Marcel pense-t-il que la DRH proposera le poste ? *(To whom does Marcel think that the manager will offer the position?)*

Clitics, including clitic climbing from argument position, but also from modifier position [20]

- L'ingénieur me le donne *The engineer gives it to me, (lit: The engineer me:dat it:acc gives)*
- L'ingénieur le lui donne *The engineer gives it to him, (lit: The engineer it:acc him:dat gives)*

[20]Clitic climbing, and in general extraction, from modifer positions, is easily handled by LFG, but not by TAGs (ex: *Which castle did you take a picture of*)

- L'ingénieur l'y rencontre *The engineer meets him there, (lit: The engineer him:acc there meets)*
- L'ingénieur l'y a toujours rencontré *The engineer always met him there, (lit: The engineer him:acc there has always met)*
- L'ingénieur le fait travailler *The engineer makes him work, (lit: The engineer him:acc makes to work)*
- L'ingénieur en propose le contenu *The engineer proposes the contents of it, (lit: The engineer of it proposes the content)*

3.6 Conclusion

We have presented a MetaGrammar tool which allows us to automatically generate a French LFG grammar. We keep enriching our hierarchy in order to increase the coverage of our grammars, and plan to add new languages and new formalisms. We are also investigating the automatic acquisition of a MetaGrammar can from a treebank.

References

Abeillé, A., M. Candito, and A. Kinyon. 1999. FTAG: current status and parsing scheme. In *Proceedings of the VEXTAL Conference (Venezia per il Trattamento Automatico delle Lingue*. Venice.

Boullier, P. 1998. Proposal for a natural language processing syntactic backbone. Technical Report, INRIA, France.

Bresnan, J. and R. Kaplan. 1982. Introduction: grammars as mental representations of language. In *The Mental Representation of Grammatical Relations*, pages xvii–lii. Cambridge: MIT Press.

Butt, M., S. Dipper, A. Frank, and T. Holloway-King. 1999. Writing large-scale parallel grammars for English, French, and German. In *Proceedings of the LFG '99 Conference*. CSLI Online Publications.

Butt, M., H. Dyvik, T. H. King, H. Masuichi, and C. Rohrer. 2002. The parallel grammar project. In *Proceedings of 19th International Conference on Computational Linguistics, Workshop on Grammar Engineering and Evaluation*, pages 1–7. Taipei.

Candito, M. H. 1996. A principle-based hierarchical representation of LTAGs. In *Proceedings of the 16th International Conference on Computational Linguistics*, vol. 1, pages 194–199. Copenhagen.

Clément, L. and A. Kinyon. 2001. XLFG: an LFG parsing scheme for French. In *Proceedings of the LFG '01 Conference*. Hong-Kong. CSLI Online Publications.

Clément, L. and A. Kinyon. 2003a. Generating LFGs with a MetaGrammar. In *Proceedings of the LFG '03 Conference*. Saratoga Springs. CSLI Online Publications.

Clément, L. and A. Kinyon. 2003b. Generating parallel multilingual LFG-TAG grammars with a MetaGrammar. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Sapporo.

Dalrymple, M., J. Lamping, F. Pereira, and V. Saraswat. 1995. Linear logic for meaning assembly. In *Proceedings of the Joint COMPULOGNET/ELSNET/EAGLES Workshop on Computational Logic for Natural Language Processing*. Edinburgh.

Danlos, L. and O. Laurens. 1991. Présentation du projet Eurotra et des grammaires d'Eurotra-France. Technical Report, Talana, CNRS Paris 7.

Flickinger, D. 1987. *Lexical rules in the hierarchical lexicon*. Ph.D. thesis, Stanford.

Frank, A. 2000. Automatic F-Structure annotation of treebank trees. In *Proceedings of the LFG '00 Conference*. Berkeley. CSLI Online Publications.

Gaiffe, B., B. Crabbe, and A. Roussanaly. 2002. A new metagrammar compiler. In *Proceedings of the 6th International Workshop on TAG and Related Formalisms*. Venice.

Gerdes, K. 2002. DTAG: attempt to generate a useful TAG for German using a metagrammar. In *Proceedings of the 6th International Workshop on TAG and Related Formalisms*. Venice.

Hepple, M. and J. van Genabith. 2000. Experiments in structure preserving grammar compaction. In *Proceedings of the 1st meeting on Speech Technology Transfer*. Sevilla.

Joshi, A. K. and K. Vijay-Shanker. 1989. Treatment of long distance dependencies in LFG and TAG: Functional uncertainty in LFG is a corollary in TAG. In *Proceedings of the 27th Meeting of the Association for Computational Linguistics*, pages 220–227. Vancouver.

Kameyama, M. 1986. Characterising LFG in terms of TAG. Technical Report, University of Pennsylvania.

Kaplan, R. and J. Maxwell. 1996. LFG grammar writer's workbench. Technical Report, Xerox Corporation. Version 3.1.

Kinyon, A. 2000. Hypertags. In *Proceedings of the 18th International Conference on Computational Linguistics*, vol. 1, pages 446–452. Saarbruecken.

Kinyon, A. 2003. MetaGrammars for efficient development, extraction and generation of parallel grammars. Ph.D. thesis proposal, University of Pennsylvania.

Kinyon, A. and C. Prolo. 2002. A classification of grammar development strategies. In *Proceedings of 19th International Conference on Computational Linguistics, Workshop on Grammar Engineering and Evaluation*. Taipei.

Kinyon, A. and O. Rambow. 2003a. Using a MetaGrammar for parallel multilingual grammar development and documentation. In *Proceedings of the 15th European Summer School in Logic Language and Information, Workshop on multilingual grammar development*. Vienna.

Kinyon, A. and O. Rambow. 2003b. Using the metagrammar to generate cross-language and cross-framework annotated test-suites. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics, 4th International Workshop on Linguistically Interpreted Corpora*. Budapest.

Srinivas, B. 1997. *Complexity of lexical descriptions and its relevance for partial parsing*. Ph.D. thesis, University of Pennsylvania.

Xia, F. 2001. *Automatic grammar generation from two perspectives*. Ph.D. thesis, University of Pennsylvania.

4

An Asymmetric Theory of Peripheral Sharing in HPSG: Conjunction Reduction and Coordination of Unlikes

Berthold Crysmann

In this paper, I will address the phenomenon of Conjunction Reduction and suggest a linearisation-based account that differentiates between heads and dependents as to the strength of the identity requirements they observe in constructions displaying peripheral sharing. This distinction will build the foundation of an asymmetric surface-oriented theory of reconstruction that will be able to avoid unwanted unification clashes on the valence lists of heads, obviating the need for subsumption checks and/or generalisation operations in HPSG. I will further show that this theory of Conjunction Reduction can also be applied to resolve similar issues associated with Coordination of Unlikes, resulting in a much more stream-lined, purely unification-based perspective on coordination in HPSG.

4.1 Introduction

The area of coordination has always been a challenge for unification-based phrase structure grammar as soon as the scope of data to be addressed was extended beyond simple constituent coordination. While on the one hand, unification was claimed to be too strong a concept to implement the kind of likeness conditions that appear operative in

Proceedings of FGVienna: The 8th Conference on Formal Grammar.
Gerald Penn (ed.).
Copyright © 2008, CSLI Publications.

cases of feature indeterminacy and Coordination of Unlikes (Ingria, 1990, Bayer and Johnson, 1995, Bayer, 1996), orientation towards classical constituent structure does not seem to provide the right tools to address instances of Non-Constituent Coordination (NCC), such as Conjunction Reduction (CR; see (1)) or Right-Node Raising (RNR; see (2)).

(1) Bill gave the girls spades and the boys recorders. (Maxwell and Manning, 1996)

(2) Bill likes, and Joe is thought to like cigars from Cuba. (Maxwell and Manning, 1996)

Despite differences in implementation, the solutions that researchers within the frameworks of LFG or HPSG have come-up with display a high degree of similarity. Whereas feature indeterminacy is addressed by means of a refinement to the representation of morphosyntactic features, either as sets (Dalrymple and Kaplan, 2000) or type hierarchies (e.g. Daniels, 2001),[1] NCC is resolved by appeal to surface order, using suspension/resumption of c-structure rules (Maxwell and Manning, 1996), or by applying extended word order domains (Reape, 1994, Kathol, 1995) to the domain of coordination (Blevins and Sag, 1996, Yatabe, 2000, 2003, Crysmann, 2000). In essence, the HPSG approaches to NCC appear to belong to what Maxwell and Manning (1996) called "a new breed of theories which allow non-constituent coordinations by effectively moving coordination up one level while maintaining classical notions of constituency."

In this paper I will discuss the phenomenon of Conjunction Reduction and the kind of problems it poses for a linearisation-based treatment in HPSG. We will find that heads behave quite differently from dependents in this construction and show that a naive theory of peripheral sharing as unification of domain objects will run into systematic problems which cannot be easily resolved along the lines of feature indeterminacy. I will propose an asymmetric approach to head and dependent sharing which essentially constitutes a surface-oriented theory of reconstruction and show that this perspective will also obviate the need of generalisation or subsumption tests with Coordination of Unlikes.

[1] See Levy and Pollard (2001) for a detailed comparison, suggesting formal equivalence of set-based and type-based approaches.

4.2 Word order domains and coordination: Some basic issues

It has been argued over the last decade that the description of linearisation in so-called free word order languages, such as German should best be stated at a level of structure that is considerably larger than immediate constituents: consequently Reape (1994) and Kathol (1995) have suggested to describe word order regularities at the level of a list-valued feature DOM that collects all major constituents along the head projection path. In essence, order domains offer a flat, linear representation of the main constituents, including complements, modifiers, as well as a representation of the lexical head. The detachment of word order domains from immediate constituency offers a convenient tool to implement a *shared-structure* approach (Milward, 1994) to peripheral sharing within HPSG.

It should come as no surprise that most attempts at a theory of NCC in HPSG build on the concept of order domains (Blevins and Sag, 1996, Yatabe, 2000, Crysmann, 2000), and unanimously exploit the same core idea, namely, that in the course of complex domain formation in coordinate structures peripheral domain objects can be collapsed into one, under unification. A similar idea was developed in Kathol (1995) to account for SGF-coordination in German. Yatabe (2000) explicitly relates the phenomenon of peripheral sharing to that of extraposition.

A linearisation-based theory of peripheral sharing typically makes two basic assumptions: first, that unifiable domain objects can be collapsed under token-identity, and, second, that non-shared domain objects cannot be interleaved. Owing to the monotonicity of the shuffle operation ($\bigcirc$) used for domain construction, ordering within each conjunct will be preserved in the order domain of the coordinate structure. Thus, it will be possible to identify peripheral elements solely in terms of their linear position on the DOM-list of each conjunct daughter. In order to prevent interleaving of non-shared domain objects on the DOM-list of the entire coordinate structure (see the examples in (3)), Yatabe (2000) and Crysmann (2000) suggest partial compaction of the non-shared rest, an operation familiar from Kathol's (1995) approach to relative clause extraposition: basically, the non-shared domain objects of each conjunct are compacted into a single domain object on the DOM-list of the mother.

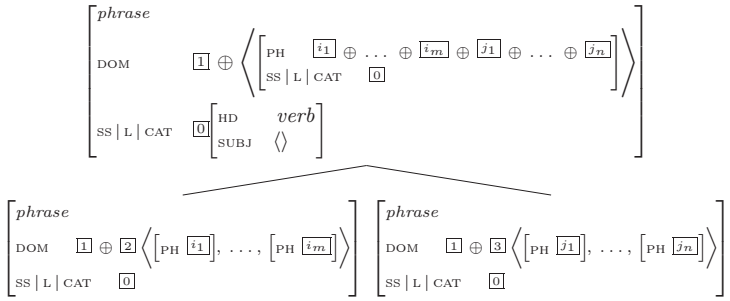
(3) a. * John gave Mary a book and a record on Wednesday on Tuesday.

b. * John gave Mary a book a record and on Wednesday on Tuesday.

c. * John gave Mary a book and a record on Wednesday and on Tuesday.

Under these accounts, the phonological contribution of this domain object is derived by means of concatenation of the compacted domain objects, with categorial information being constructionally assigned by the surrounding coordinate structure.

(4)



As a side-effect, partial compaction will also solve the issue of inconsistent ordering within and across conjuncts: given standard semantics of LP constraints, a statement like the one in (5) would be violated, if we were to represent non-shared domain objects as such in the order domain of the entire coordeinate structure.

(5) NP[acc] $\prec$ NP[to]
($\equiv$ *sign* $\rightarrow \neg$ [DOM $\langle$...,NP[to], ..., NP[acc], ...$\rangle$])

(6) John gave a book *to Mary* and *a record* to Peter.

Partial compaction, however, renders the non-shared rest opaque, effectively guaranteeing consistent ordering.

To summarise, linearisation-based accounts of NCC try to capture the intuition that a structure featuring peripheral sharing is well-formed whenever the individual conjuncts are well-formed, provided a contiguous portion of the surface string is shared. Thus, under these theories, coordination targets fully saturated signs, i.e. feature structures where all internal subcategorisation requirements have already been fulfilled, and then merely collapses a peripheral sequence of domain objects in the linear representation of the coordinated structure.

4.3 Peripheral sharing and valence information: a problem

An aspect that has been glossed over by most domain-based approaches to NCC to date (e.g. Blevins and Sag, 1996, Crysmann, 2000, Yatabe,

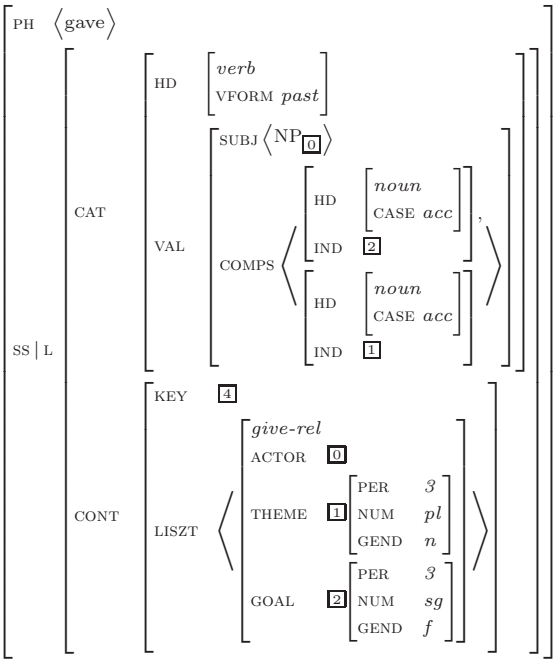
2000) relates to the sharing of the head domain object in CR (or LNR/RNR) constructions: while the linearisation-based approach to CR in terms of shared peripheral domain objects gives sound results for dependents, the coordination schema given above actually breaks, if the non-shared dependents differ with respect to their semantics. The reason is that the head domain object specifies, on its valence lists, relations to both shared and non-shared dependents. With shared dependents, no problem may arise, as they are, by necessity, coreferent across the two conjuncts. For non-shared dependents, however, the head domain objects contributed by each conjunct will have differing CONT values on their valence lists.

(7) John gave Mary a book on Wednesday and several records on Saturday.

(8)

$$\left[\begin{array}{ll}
\textsc{ph} & \langle \text{gave} \rangle \\
\textsc{ss}\,|\,\textsc{l} & \left[\begin{array}{ll}
\textsc{cat} & \left[\begin{array}{ll}
\textsc{hd} & \left[\begin{array}{ll} \textit{verb} & \\ \textsc{vform} & \textit{past} \end{array}\right] \\
\textsc{val} & \left[\begin{array}{ll}
\textsc{subj} & \langle \text{NP}\boxed{0} \rangle \\
\textsc{comps} & \left\langle \left[\begin{array}{ll} \textsc{hd} & \left[\begin{array}{ll} \textit{noun} & \\ \textsc{case} & \textit{acc} \end{array}\right] \\ \textsc{ind} & \boxed{2} \end{array}\right], \left[\begin{array}{ll} \textsc{hd} & \left[\begin{array}{ll} \textit{noun} & \\ \textsc{case} & \textit{acc} \end{array}\right] \\ \textsc{ind} & \boxed{1} \end{array}\right] \right\rangle
\end{array}\right]
\end{array}\right] \\
\textsc{cont} & \left[\begin{array}{ll}
\textsc{key} & \boxed{4} \\
\textsc{liszt} & \left\langle \boxed{4} \left[\begin{array}{ll}
\textit{give-rel} & \\
\textsc{actor} & \boxed{0} \\
\textsc{theme} & \boxed{1} \left[\begin{array}{ll} \textsc{per} & \textit{3} \\ \textsc{num} & \textit{sg} \\ \textsc{gend} & \textit{n} \end{array}\right] \\
\textsc{goal} & \boxed{2} \left[\begin{array}{ll} \textsc{per} & \textit{3} \\ \textsc{num} & \textit{sg} \\ \textsc{gend} & \textit{f} \end{array}\right]
\end{array}\right] \right\rangle
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

(9)



To make the problem a bit more apparent, let us consider a concrete example: in the above sentence, the head domain object contributed by the first conjunct will have a COMPS value as indicated in (8), i.e. requiring the second complement's CONT—INDEX—NUM to be of type *sg*, whereas the head of the second conjunct will require its corresponding second complement to have a specification for *pl*, cf. (9).[2]

As a result, if the head is shared on the DOM list of the entire coordinated structure, simple token-identity of the domain objects will not do, due to a unification failure on the subcategorisation requirements for the second complement. Similarly, unification of the verbs' CONT values will equally clash.

4.3.1 Semantic differences between heads and dependents

Thus, the first issue to be addressed is the question of what semantic interpretation we would want to assign to coordinated structures in which the head is shared. A closer look at (7) reveals that sharing of the entire domain object (and, hence its SYNSEM value) does not give sound results for the event semantics either. As the shared head *gave*

[2]Certainly, a difference in index features is not the only discrepancy that will arise here. Rather, the incompatibilities will affect the entire CONT-value, including semantic relations.

is modified by a different temporal expression in either conjunct, it is clear that (7) actually denotes two independent giving events. Token-identity of the domain objects obviously gives the wrong result here, as it entails structure sharing of the event variable as well.[3]

Thus, in semantic terms, sharing of heads differs quite drastically from sharing of dependents, where token-identity of CONT values is necessary to derive the correct interpretation:[4]

(10) a. Few men drink and smoke.
b. $\not\rightarrow$ Few men drink and few men smoke.

(11) a. Few men gave Mary a book on Friday and a record on Saturday.
b. $\not\rightarrow$ Few men gave Mary a book on Friday and few men gave Mary a record on Saturday.

(12) a. I gave few men a book on Friday and a record on Saturday.
b. $\not\rightarrow$ I gave few men a book on Friday and I gave few men a record on Saturday.

Without token-identity of CONT values, we would give the above sentences in a. an interpretation equivalent to b., whereas structure-sharing will ensure that the sets denoted by *few men* are identical across the two conjuncts in the a. sentences, while they may be disjoint in the corresponding sentences in b. From this, we can conclude that, semantically, shared dependents and shared heads behave entirely different from another: as a consequence, we would want to equate the CONT values of the former, whereas we certainly do not want to do so for the latter. Thus, it seems to make perfect sense to relax the requirement concerning structure-sharing of the entire domain object in the case of heads, yet retain this requirement for dependents.

Now that we have established that sharing of the heads' CONT values in coordinated structures is undesirable for independent reasons, we can

[3]The issue of token-identity of domain objects is pretty much independent of where exactly semantic composition will take place (which I take to be constituent structure): if SYNSEM values get unified on DOM, there is no way of undoing this effect anywhere else in the sign.

[4]I will restrict myself here to a discussion of CR. What is clear, though, is that sharing in RNR cannot involve token-identity of entire head domain objects:

We can but you can't come to his party on Friday.

Independent of what the exact empirical situation is for dependent sharing here, the asymmetric approach to be developed for CR should be flexible enough to accommodate either situation. Thus, if, with RNR, shared dependents are not necessarily coreferent across the conjuncts, a conclusion suggested by binding facts, it will be sufficient to impose the analogue of (18), as well as (19). However, I will reserve this phenomenon as a topic for future research.

move on to the second issue, namely sharing of valence information.

4.3.2 Parallelism requirements

Although it is clear that the CONT specifications on the valence lists of the shared heads do not need to match up across the two conjuncts, we have to ask, of course, whether this is also true for any of the other information these heads subcategorise for, e.g. syntactic category and case. A piece of data that may bear on this point is given in (13) below: while speakers of English seem to be strictly divided as to the acceptability of this sentence, there is a considerable subset of speakers who utterly reject it.

(13) (*) John gave Mary a book and a record to Peter.

However, as pointed out to me by Doug Arnold (p.c.), the following example featuring heavy NP-shift in the second conjunct is equally ruled out for him, despite the parallelism in subcategorisation frames.

(14) * John gave a book to Mary and to Peter a very old and famous early Beatles' record featuring vocals by Ringo.

Similarly, Bob Levine (p.c.) remarks that (13) improves considerably, if surface order of theme and goal arguments match up across the two conjuncts:

(15) John gave Mary a book, and to Peter, a record.

Thus, whatever causes (13) to be unacceptable for some speakers, it is certainly a factor unrelated to the mismatch in subcategorisation requirements. As argued by Whitman (2002), acceptability of the non-parallel cases improves, once a suitable context is provided.

(16) Q: So Neal, did you give the dog that toy like you were planning to do?

(17) A: No, actually I changed my mind. I gave the dog a bone, and the toy to the cat.

Taken together, all these data appear to indicate that the parallelism requirement should better be understood in terms of information structure, rather than in terms of strictly parallel subcategorisation frames. Moreover, even if one wished to impose a restriction as to the parallelism in category and case of non-shared valencies, the VAL feature of the shared head is surely not the only locus where such a constraint could be expressed: alternatively, the parallelism may equally well be captured as a constraint on the DOM lists of the conjoint daughters.

Such a solution will not only be more flexible but also feature the additional benefit of offering an explanation for the data in (14) concerning heavy NP shift.

4.3.3 Some suboptimal solutions

Having established now that parallelism of valencies is not an intrinsic property of head sharing, we have to address the issue how VAL features should be composed in CR. A quick and presumably quite dirty fix to the incompatibility of subcategorisation requirements would be to assume that the valence lists of the shared head domain object are the generalisation, rather than the unification of the valence lists of the corresponding domain objects on each conjunct. To say the least, such a solution would stand in sharp contradiction to the assumption that the feature structures modelled by an HPSG theory are required to be totally well-typed and sort-resolved. Moreover, outside the domain of coordination, use of this operation is marginal at best, if not entirely unattested. The only exception to be found is Daniels's (2001) account of Coordination of Unlikes. However, as we will see shortly, our asymmetric theory of head and dependent sharing will provide a solution here as well.

Another, probably equally bad, solution would be to concatenate, rather than unify the valence lists. Certainly, such a move is pretty ad hoc and devoid of any deeper linguistic motivation. Worse, as conflict is not limited to the valence lists, but also affects ARG-ST, concatenation would have to be replicated here as well. This, however, will interfere quite badly with Binding Theory, or, more precisely, the obliqueness command relationship represented in this feature.

A third alternative has been proposed in Yatabe (2003): here, the valence lists are composed component-wise creating a conjoint subcategorisation requirement. Although this is a technically viable solution, it introduces a new species of entities into the domain of subcategorisation: Outside the area of peripheral sharing, however, specific subcategorisation for distributive conjoint dependents is both unattested and unnecessary. Furthermore, even within Yatabe's system it is unclear what role these conjoint subcategorisation requirement should play besides the obvious reason to avoid unification clashes on the valence list of the shared head domain object: as subcategorisation requirements are satisfied within each conjunct individually, it appears that this new type of conjoint synsem object is fully redundant and, as such, does not really contribute to our understanding of the linguistic issue at stake.

4.4 An asymmetric approach to head/dependent sharing

Instead, I will pursue an essentially asymmetric approach to head sharing: while dependents will require token-identity between the shared domain object on the DOM list of the mother and the corresponding domain objects in either conjunct, with heads, the entire domain object on the mother will be structure-shared with a corresponding domain object in a single conjunct only. Token-identity of PHON and HEAD values, however, is obligatory for all shared material across all conjuncts, irrespective of the head-dependent divide.

The intuition underlying this approach pertains to the observation that CR structures can be divided into a continuous string, which corresponds to the yield of the DOM list of the first conjunct, and a remainder, which only represents the right part of the domain structure of the second conjunct. Thus, it appears that the shared material on the DOM-list of the mother can actually be directly projected from the first conjunct, whereas it has to be "reconstructed" for the second.

Thus, we can establish that left-peripheral sharing proceeds along three dimensions: identity of phonology and category across both conjuncts, projection of the entire domain object from the first conjunct, and token-identity of shared dependents across all conjuncts. We can implement these three different dimensions of sharing straightforwardly by means of three constraints.

The first constraint implements the minimal requirement that sharing in CR entails token-identity of phonologies. To avoid sharing of homophonous, though categorially distinct material, we can further demand identity of HEAD values. In addition, this constraint will guarantee that sharing of phonologies can only involve contiguous peripheral stretches, in this case sharing at the left edge.

(18) $\begin{bmatrix} \textit{phrase} \\ \textsc{dtrs}\ \textit{l-shrd-coord-str} \end{bmatrix} \rightarrow$

$$\begin{bmatrix}
\textsc{dom} \left\langle \begin{bmatrix} \textsc{ph} & \boxed{i_1} \\ \textsc{hd} & \boxed{h_1} \end{bmatrix}, \ldots, \begin{bmatrix} \textsc{ph} & \boxed{i_m} \\ \textsc{hd} & \boxed{h_m} \end{bmatrix} \right\rangle \oplus \left\langle \begin{bmatrix} \textsc{ph}\ \boxed{j_1} \oplus \ldots \oplus \boxed{j_n} \oplus \boxed{k_1} \oplus \ldots \oplus \boxed{k_o} \\ \textsc{ss}\mid\textsc{l}\mid\textsc{cat} \quad \boxed{0} \end{bmatrix} \right\rangle \\
\textsc{ss}\mid\textsc{l}\mid\textsc{cat}\ \boxed{0} \begin{bmatrix} \textsc{spr} & \langle\rangle \\ \textsc{subj} & \langle\rangle \\ \textsc{comps} & \langle\rangle \end{bmatrix} \\
\textsc{conj-dtrs} \left\langle \begin{bmatrix} \textsc{dom} \left\langle \begin{bmatrix} \textsc{ph} & \boxed{i_1} \\ \textsc{hd} & \boxed{h_1} \end{bmatrix}, \ldots, \begin{bmatrix} \textsc{ph} & \boxed{i_m} \\ \textsc{hd} & \boxed{h_m} \end{bmatrix} \right\rangle \oplus \left\langle \left[\textsc{ph}\ \boxed{j_1}\right], \ldots, \left[\textsc{ph}\ \boxed{j_n}\right] \right\rangle \\ \textsc{ss}\mid\textsc{l}\mid\textsc{cat} \quad \boxed{0} \end{bmatrix}, \right. \\
\left. \begin{bmatrix} \textsc{dom} \left\langle \begin{bmatrix} \textsc{ph} & \boxed{i_1} \\ \textsc{hd} & \boxed{h_1} \end{bmatrix}, \ldots, \begin{bmatrix} \textsc{ph} & \boxed{i_m} \\ \textsc{hd} & \boxed{h_m} \end{bmatrix} \right\rangle \oplus \left\langle \left[\textsc{ph}\ \boxed{k_1}\right], \ldots, \left[\textsc{ph}\ \boxed{k_o}\right] \right\rangle \\ \textsc{ss}\mid\textsc{l}\mid\textsc{cat} \quad \boxed{0} \end{bmatrix} \right\rangle
\end{bmatrix}$$

Projection of SYNSEM values from the DOM list of one of the conjuncts is given in (19). Here, all we have to do is specify that all domain objects except one on the DOM list of the entire coordinated structure are structure-shared with domain objects on one of the conjunct daughters. The single domain object that is exempt from this requirement is the compaction of the unshared domain objects, identified by means of the token-identity of its CAT value with that of the coordinated structure ($\boxed{0}$).

(19) $\begin{bmatrix} \textit{phrase} \\ \textsc{dtrs}\ \textit{coord-struc} \end{bmatrix} \rightarrow$

$$\begin{bmatrix}
\textsc{dom} & \boxed{1} \bigcirc \left\langle \left[\textsc{ss}\mid\textsc{l}\mid\textsc{cat}\ \boxed{0}\right] \right\rangle \\
\textsc{ss}\mid\textsc{l}\mid\textsc{cat} & \boxed{0} \\
\textsc{conj-dtrs} & \left\langle \left[\textsc{dom}\ \boxed{1} \bigcirc \textit{list}\right] \right\rangle \bigcirc \textit{list}
\end{bmatrix}$$

As a last step, we can enforce sharing of the SYNSEM value of dependents across all conjuncts: this is achieved by ruling out any situation where there is a domain object with all-saturated valencies on the DOM list of the mother that fails to be structure-shared with a domain object on any of the conjunct daughters. Again, the single domain object that holds the compacted non-shared rest is exempted from this requirement.

(20) $\begin{bmatrix} \textit{phrase} \\ \textsc{dtrs}\ \textit{l-shrd-coord-struc} \end{bmatrix} \rightarrow$

$$\neg \left[\begin{array}{l} \textsc{dom} \quad \left\langle \boxed{1} \left[\textsc{ss} \mid \textsc{l} \mid \textsc{cat} \mid \textsc{val} \begin{bmatrix} \textsc{subj} & \langle\rangle \\ \textsc{spr} & \langle\rangle \\ \textsc{comps} & \langle\rangle \end{bmatrix} \right] \right\rangle \bigcirc \left\langle \left[\textsc{ss} \mid \textsc{l} \mid \textsc{cat} \ \boxed{0} \right] \right\rangle \bigcirc \textit{list} \\ \textsc{ss} \mid \textsc{l} \mid \textsc{cat} \quad \boxed{0} \\ \textsc{conj-dtrs} \quad \left\langle \neg \left[\textsc{dom} \quad \left\langle \boxed{1} \right\rangle \bigcirc \textit{list} \right] \right\rangle \bigcirc \textit{list} \end{array} \right]$$

As such, the technical implementation of the analysis proposed has all the necessary properties identified above: the constraint in (20) is capable of deriving that shared dependents are necessarily interpreted as coreferent, accounting for the contrasts in (10–12). Shared heads, however, are subject to a somewhat weaker constraint that only requires token-identity with a domain object in one of the conjuncts. Still, projecting SYNSEM information of heads from a single conjunct ensures that the corresponding domain object on the mother will contain sufficiently rich information such that order constraints that hold on each of the conjuncts can be satisfied identically on the DOM list of the mother: as the only domain objects transparently represented on the mother are shared heads and shared dependents, and the shared dependents are actually required to be token-identical to those in all the conjuncts, it is clear that even the valency information for shared dependents will be structure-shared on the VAL features of the head, despite the fact that the SYNSEM value of this head is only projected from a single conjunct. The only place where projection from a single conjunct makes a difference is the representation of non-shared dependents. However, for linearisation purposes this does not matter at all, as non-shared dependents are not transparently represented on the DOM list of the entire structure anyway, but rather, they are compacted into a single opaque domain object.

4.4.1 Some qualifications regarding dependent sharing

On closer examination one will observe that even dependents are not all created equal: while the sentences in (21a) and (21b) below necessarily demand identity of individual real-world referents for the (animate) theme or goal argument, the sentence in (21c) can only be interpreted, given world knowledge, in the sense that at least two separate copies of some letter have been posted to their respective destinations.

(21) a. Mary sent a friend a letter on Friday and a postcard on Saturday.

b. I introduced a friend to the head of department on Thursday and to the dean on Friday.

c. Mary sent a letter to Bill on Friday and to Peter on Saturday.

d. ? A letter was posted from Gozo last Saturday and from Tunis this week.

Interestingly enough, this division appears to be independent of both grammatical function and syntactic position: passives, like the one in (21d) above do not pattern all too differently in this respect.

The shared dependents in (21c) or (21d) still require similarity of the letters, e.g. with respect to contents. Apparently, this restriction is beyond what is captured by mere identity of the type of semantic relation. Thus, I would rather interpret these facts as an instance of coercion of *letter* to an abstract notion (which is token-identical across the two conjuncts) that happens to have more than one concrete physical instantiation. Coercion to a single abstract referent, however, can only be enforced, if we impose token-identity syntactically. As a result, the asymmetry between animate dependents, as in (21a,b) and inanimate ones, as in (21c,d) will be related to the ease or difficulty with which such an abstract referent can be given.

But can we also find empirical support for this hypothesis? That is, will shared dependents only receive a distributive interpretation as a result of coercion? A conclusive piece of evidence may be found in the context of intersective modification. As we have observed above, sharing of verbal heads in CR constructions does not entail sharing of event variables. As a consequence, these distinct event variables can easily be modified by different temporal modifiers.

(22) John gave Mary a book on Wednesday and several records on Saturday.

Shared dependents, however, do not seem to pattern with heads in this respect:

(23) a. ?? An email was posted on Friday from an account in China, advertising Viagra, and on Saturday from the States, offering age-reversal remedies.

b. ? Some bulk mailer sent an email from an account in China, advertising Viagra, and from the States, offering age-reversal remedies.

Modification of a shared dependent by different intersective modifiers makes the entire construction awkward, if not unacceptable.[5] Under the assumption that dependent sharing does indeed involve sharing of indices, the unacceptability of the above sentences can readily be explained: if the indices are shared, both modifiers will target one and the same individual variable. Coercion to some abstract notion of bulk email cannot apply in this case, owing to the simple fact that the conflicting modifiers render the interpretation highly specific.

Before we close the discussion of dependent sharing, I would like to address an apparent counter-example to the hypothesis advanced here, brought up by Shalom Lappin (p.c.).

(24) She saw most films in Cannes this year and in Venice last year.

Here, the most likely interpretation is clearly a distributive one. But does the above sentence really constitute counter-evidence to the claim raised here? To decide upon this, we first need to establish that we are actualy dealing with an unambiguous case of conjunction reduction at the sentential level. Attachment of the spacial modifiers, however, is systematically ambiguous between modification of the event and modification of the direct object. Under the salient distributive interpretation, however, the locative modifiers *in Cannes* and *in Venice* are part of the quantifier's restrictor. Similarly, the temporal expressions most probably modify the understood festival, and not just the seeing event. This perspective is also supported by syntactic facts, see (25) below.

(25) Most films in Cannes this year and in Venice last year were actually quite boring.

Thus, what looks superficially like an instance of Conjunction Reduction at the sentential level turns out to be better understood as CR at the NP (or DP) level. Within the NP, the quantifiers are definitely semantic and, arguably, syntactic heads. Assuming a DP approach, the

[5]The degree of acceptability enjoyed by the examples in (23b) above is evidently much higher than that of the corresponding example in (23a). The way I interpret these facts is that in b., but not in a. the verb provides an additional target for modification by the gerund. Once we control for this possibility, e.g., by choosing a relative clause, the acceptability soon degrades.

i. Some bulk mailer sent an email from an account in China, which advertised Viagra, and another one from the States, which offered age-reversal remedies.

ii. ?? Some bulk mailer sent an email from an account in China, which advertised Viagra, and from the States, which offered age-reversal remedies.

apparent counter-example can be reduced to a case of head-sharing CR in the nominal domain.

4.4.2 Peripheral sharing and argument composition

The asymmetrical approach to head/dependent sharing also provides a straightforward account of argument composition verbs taking wide scope over a coordination of complements. As argued by Abeillé and Godard (1994), classical constituency tests provide little or no evidence in favour of a VP constituent in French auxiliary-participle constructions. Instead, they suggest that French auxiliaries form a flat constituent structure with their participial complement, realising all of the complement's arguments as its own complements.

(26)
$$\left[\begin{array}{l}\textsc{ss}\mid\textsc{loc}\left[\begin{array}{ll}\textsc{cat} & \left[\begin{array}{ll}\textsc{head} & \textit{verb}\\ \textsc{arg-st} & \left\langle \boxed{1}, \mathrm{V}\left[\begin{array}{ll}\textsc{vform} & \textit{past-p}\\ \textsc{cont} & \boxed{3}\\ \textsc{arg-st} & \langle\boxed{1}\rangle \oplus \boxed{2}\end{array}\right]\right\rangle \oplus \boxed{2}\end{array}\right]\\ \textsc{cont} & \boxed{3}\end{array}\right]\end{array}\right]$$

Argument composition also provides the key mechanism in HPSG accounts of French cliticisation (Miller and Sag, 1997): as argued convincingly by Miller (1992), French pronominal clitics much more resemble lexical affixes than true post-lexical clitics. In particular they display a high degree of selection towards their host, they feature arbitrary gaps in the set of clitic and clitic-host combinations, and they are subject to morphophonological and semantic idosyncrasies. Nevertheless, they sometimes get realised on a host they are not strictly an argument of, a phenomenon referred to as clitic climbing. Miller and Sag (1997) crucially build on argument composition in deriving non-local realisation of pronominal affixes in a lexicalist fashion.

Coordination data like the one in (27) are probably the only evidence one may find in favour of French auxiliaries taking a VP complement. However, Abeillé and Godard (1994) point out that these data may equally well be understood as an instance of CR. Manning (1997), by contrast, argues that a CR account of argument composition verbs is not entirely straightforward, as the auxiliaries to be shared may differ with respect to their (inherited) argument structure.

(27) a. Paul a parlé avec Marie et compris son erreur.
Paul has spoken with Marie and understood his mistake
'Paul has spoken with Marie and understood his mistake.' (Abeillé and Godard, 1994, p. 159)

b. Paul est arrivé à dix heures et reparti à midi.
Paul is arrived at 10 o'clock and left at noon

'Paul arrived at 10 o'clock and left again at noon.' (Abeillé and Godard, 1994, p. 159)

(28) a. Paul l'a insulté et mis à la porte.
Paul him.has insulted and set to the door
'Paul insulted him and threw him out.' (Manning, 1997, p. 9)

b. La bonne femme les a cuit au four et fait
the woman them has cooked in.the oven and made
manger à son fils.
eat to her son
'The woman has cooked them in the oven and had her son eat them.' (Manning, 1997, p. 9)

(28) obviously presents a challenge to the theories of Abeillé and Godard (1994) and Miller and Sag (1997): on the one hand, placement of the pronominal affix appears to necessitate argument composition, whereas raising of all arguments other than the affixal argument would give rise to unification failure, if sharing of the auxiliary required token-identity of the shared auxiliary across both conjuncts.[6]

Under the asymmetric approach to head/dependent sharing that we have developed above, however, no conflict will arise, as the auxiliary's domain object on the DOM list of the mother need only be token-identical to the corresponding domain object on one of the conjuncts. Likeness of the heads being shared is attributed instead to token-identity of PHON and HEAD values. Again, just like in the English cases discussed earlier, this approach also gives sound results from the point of view of semantics: as witnessed by (27b), sharing of the tensed auxiliary does not entail identity of the event.

4.4.3 Coordination of Unlikes

Probably the only phenomenon which appears to require a generalisation operation or subsumption tests as part of the HPSG formalism is Coordination of Unlikes (cf. the Weak Coordination Principle in Pollard and Sag, 1994), a position that is shared by almost every approach to the issue (Blevins and Sag, 1996, Daniels, 2001, Sag, 2003).

(29) Pat is [[a Republican] and [proud of it]].

Given the standard model-theoretic interpretation of an HPSG theory, generalisation is not entirely unproblematic (see Pollard and Sag, 1994), in that the feature structures denoted by the generalisation will

[6] Here, the situation is even worse than in the English examples discussed above, as the valence lists may even differ in length.

not be totally well-typed and sort-resolved.[7] The proposal by Daniels (2001) represents an exception to this, in that a "coordination type," such as *acc+dat* not only subsumes the two types *acc* and *dat* it is the generalisation of, but also immediately subsumes a maximal type *p-acc+dat*, a "pure type" in Daniels' terminology. Independent of the formal ramifications of subsumption tests and/or generalisation, it is clear that introducing a specific operation for a single construction only does not exactly reflect standard ideas about scientific economy. In other words, a theory of language that can be cast entirely in terms of unification certainly makes a much stronger claim.

In the domain of CR, we have already seen that potential unification clashes simply do not arise, once we adopt a slightly different perspective on the data, namely in terms of an asymmetric surface-oriented theory of reconstruction. Ever since Sag et al. (1985), the underlying intuition was that what makes Coordination of Unlikes acceptable is that each conjunct is actually well-formed when combined individually with the shared rest. In essence, this characterisation of Coordination of Unlikes is highly reminiscent of, if not identical to the one we would give for NCC. Thus, it will be worthwhile to pursue whether coordination of unlikes should not be better understood as a case of CR.

Under this perspective, the auxiliary will combine independently with an NP or AP in each conjunct, thereby guaranteeing that subcategorisationn requirements are always met. Coordination, however, will apply at the level of sentential signs. According to our asymmetric theory of CR detailed above, the DOM list of the entire coordinate structure will consist of a single domain object, which is the compaction of the non-shared strings from both conjuncts, preceded by a list of domain objects. The HEAD and PHON values of these domain objects are constrained to be token-identical to those of the initial domain objects in both conjuncts. Furthermore, our constraint in (19) requires that each domain object on this list corresponding to shared material must be unified as a whole with the corresponding domain objects in the first conjunct, essentially warranting that DOM-LIST of the mother will always be a list of total objects projected from that conjunct which is realised as a contiguous string. Furthermore, for shared dependents, our theory of CR demands unification of entire domain objects across all conjuncts. Applied to the example above, we will find that the valencies of the head auxiliary are already satisfied, and therefore cancelled off, at the point where the phrases combine into a coordinated constituent.

[7] See, however, Sag (2003) for a proposal to do away with sort-resolvedness entirely and regard as the model of a feature structure description the most general satisfier.

Thus, at the level of constituent structure, no conflict can ever arise. At the surface-syntactic level, however, it is exactly valence information relating a shared head to non-shared dependents that provides the source of the conflict, a situation which is fully identical to the one we identified with CR. Our asymmetric theory of reconstruction, however, specifically exempts the head domain object from the stricter requirement of total token-identity across both conjuncts, and limits identity requirements to those features which are necessary to perform reconstruction.[8] To conclude, we find that a simple change in the linguistic perspective enables us to reduce the issue of coordination of unlikes to head-sharing in CR, a phenomenon for which a unification-based solution has already been developed independently.

4.5 Conclusion

In this paper I have suggested an asymmetric surface-oriented theory of reconstruction for Conjunction Reduction (CR) and similar phenomena which obviates, by virtue of a change in linguistic analysis, the need for any special extensions to the underlying unification-based formalism or its model-theoretic interpretation. I have argued that the problem of conflicting CONT values between parallel dependents of a shared head is a distinguishing property of coordinated structures: if the phonological and syntactic representation of a dependent is shared across two conjuncts, it entails identity of semantic information. Heads, however, preserve the potential to distribute over the two conjuncts. I have shown that an adequate account of the semantics of CR favours an asymmetric approach that identifies the shared domain objects on the mother with a sublist of the domain objects of a single conjunct daughter. Dependents, however, observe a stronger restriction, involving token-identity of the domain objects across all conjunct daughters. This latter requirement is sufficient to derive the effect of obligatory coreference of shared dependents.

Furthermore, I have shown that the problem unification-based theories typically face when confronted with Coordination of Unlikes can be avoided, once these constructions are conceived of as a special case of Conjunction Reduction.

[8] Throughout this paper I have contented myself with regarding phonology and basic category as sufficient to fulfil the recoverability demands. Subject to one's sense of humour, the following piece of data may suggest that we want to include the basic semantic relation under SYNSEM—LOC—CONT—KEY—RELN as well:

?? John grew potatoes and weary of it.

I further conjecture that the present asymmetric theory of head and dependent sharing can also be extended quite straightforwardly to cover gapping, the only difference being the mode of indexing. With Conjunction Reduction (and Right Node Raising), potentially shared material is identified by linear position, whereas grammatical function (most notably headhood) will provide the relevant concept in case of gapping. However, despite this difference, it appears that the asymmetric approach — in particular, the relaxation of identity requirements for head domain objects — will provide the necessary prerequisite for permitting shared heads to combine with different dependents in either conjunct. Such an account of gapping can then be understood as a linearisation- and unification-based variant of the type of Categorial Grammar analysis proposed by Steedman (1990).

References

Abeillé, A. and D. Godard. 1994. The complementation of French auxiliaries. In *Proceedings of the 13th West Coast Conference on Formal Linguistics*. Palo Alto: CSLI Publications.

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*, pages 70–76.

Blevins, J. P. and I. Sag. 1996. Constituent sharing in nonconstituent coordination. Ms., University of Alberta and Stanford University.

Crysmann, B. 2000. Clitics and coordination in linear structure. In B. Gerlach and J. Grijzenhout, eds., *Clitics in Phonology, Morphology and Syntax*, vol. 36 of *Linguistik Aktuell/Linguistics Today*, pages 121–159. John Benjamins.

Dalrymple, M. and R. Kaplan. 2000. Feature indeterminacy and feature resolution. *Language* 76(4):759–798.

Daniels, M. 2001. On a type-based analysis of feature neutrality and the coordination of unlikes. In *Proceedings of the 8th International Conference on Head-Driven Phrase Structure Grammar*, pages 137–147. CSLI Online Publications.

Ingria, R. J. P. 1990. The limits of unification. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pages 194–204.

Kathol, A. 1995. *Linearization-Based German Syntax*. Ph.D. thesis, Ohio State University.

Levy, R. and C. Pollard. 2001. Coordination and neutralization in HPSG. In *Proceedings of the 8th International Conference on Head-Driven Phrase Structure Grammar*, pages 221–234. CSLI Online Publications.

Manning, C. 1997. Romance complex predicates: in defence of the right-branching structure. Ms., University of Sydney.

Maxwell, J. T. and C. Manning. 1996. A theory of non-constituent coordination based on finite-state rules. In M. Butt and T. H. King, eds., *Proceedings of the LFG '96 Conference*. CSLI Online Publications.

Miller, P. 1992. *Clitics and Constituents in Phrase Structure Grammar*. Outstanding Dissertations in Linguistics. New York: Garland.

Miller, P. and I. Sag. 1997. French clitic movement without clitics or movement. *Natural Language and Linguistic Theory* 15(3):573–639.

Milward, D. 1994. Non-constituent coordination: Theory and practice. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 935–941.

Pollard, C. and I. Sag. 1994. *Head–Driven Phrase Structure Grammar*. Chicago: University of Chicago Press.

Reape, M. 1994. Domain union and word order variation in German. In J. Nerbonne, K. Netter, and C. Pollard, eds., *German in Head-Driven Phrase Structure Grammar*, no. 46 in Lecture Notes, pages 151–197. Palo Alto: CSLI Publications.

Sag, I., G. Gazdar, T. Wasow, and S. Weisler. 1985. Coordination and how to distinguish categories. *Natural Language and Linguistic Theory* 3:117–171.

Sag, I. A. 2003. Coordination and underspecification. In J.-B. Kim and S. Wechsler, eds., *Proceedings of the 9th International Conference on Head-driven Phrase Structure Grammar*. CSLI Online Publications.

Steedman, M. 1990. Gapping as constituent coordination. *Linguistics and Philosophy* 13:207–264.

Whitman, N. 2002. *Category Neutrality: a Type-Logical Investigation*. Ph.D. thesis, Ohio State University.

Yatabe, S. 2000. The syntax and semantics of left-node raising in Japanese. In *Proceedings of the 7th International Conference on Head-Driven Phrase Structure Grammar*, pages 325–344. CSLI Online Publications.

Yatabe, S. 2003. A linearization-based theory of summative agreement in peripheral-node raising constructions. In J.-B. Kim and S. Wechsler, eds., *Proceedings of the 9th International Conference on Head-driven Phrase Structure Grammar*. CSLI Online Publications.

5

Discourse Plans and Linguistic Meaning

Alexander Dikovsky

Linguistic meaning is presented in this paper as a result of dynamic planning. The plans, we call them *discourse plans*, are tree-like structures composed of primitive situations and establishing referential links between the situations and the context. Linguistic meanings are normal form second order Lambda-terms available through a simple morphism from realizable discourse plans.

5.1 Introduction

There are two different approaches to meaning formalization: one *logical*, another *cognitive*. The logical approach, going back to Frege and developed and explicitly applied to natural language semantics by Montague, is characterized by establishing semantics from primitive semantic structures of *sentential type*. This approach is most consistently represented by type logical grammars (see collections (Gamut 1991, van Benthem and ter Meulen 1997)).

The cognitive approach treats linguistic meaning as a structure (e.g., a graph, a DAG, a feature structure) *encoding the content* of a text in enough detail to represent it at the syntactic, morphological and phonological levels, *independent of the propositional attitude*. This is the way it appears in the theory "Meaning ⇔ Text" (Mel'čuk 1997). Formalized or completely formal descriptive definitions of cognitive meaning are well known: cf. semantic networks of Sowa 1991 and situation theory of Barwise and Perry 1983. However, these definitions construct meaning structures *per se* independent of their realization in natural

Proceedings of FGVienna: The 8th Conference on Formal Grammar.
Gerald Penn (ed.).
Copyright © 2008, CSLI Publications.

language. So in fact, they define an information and not a meaning structure. In this paper, we propose a formal definition of linguistic meaning using specific linguistic realization means. Basically, our definition is functional and object-oriented. All meaning elements may be seen as typed functions, the types forming a hierarchy based on a genericity relation. Any realistic implementation of this definition must be feature driven with feature inheritance. In the theoretical model of this paper the features are abstracted. Our primitive meaning structures are in fact very close to the primitive situations of Barwise and Perry 1983. The difference is that we use specific semantical linguistic types and roles. The basic particularity of our definition is that building complex meanings from primitive situations needs planning. The planning becomes necessary for at least three reasons:

1. Meaning composition may be complicated by type conflicts, so it may need type conversion.

2. Name and reference scoping does not always conform with meaning composition.

3. To realize a particular communicative structure (theme / rheme partition, focus) may need in general specific transformations of situation argument structure (semantic diatheses).

In that way, we come to *discourse plans* [1] - meaning specifications transformed into meanings by a simple morphism under certain realizability conditions.

5.2 Roles, Types and Situations

We start with primitives from which the discourse plans are defined. The first primitives are *cognitive roles* marking arguments of meanings. We choose in this paper the following list of cognitive roles, which is of course far from being complete, but largely suffices for the examples below: **ACT** (action), **EVT** (event), **ST** (state), **TNS** (tense), **LOC** (location), **AGT** (agent), **CAG** (counteragent), **PAT** (patent), **OBJ** (object), **EXP** (experiencer), **ADR** (addresser), **ORG** (origin), **RCP** (recipient), **DST** (destination), **EFF** (effect), **GL** (goal), **CND** (condition), **CSE** (cause), **INS** (instrument), **RES** (result), **CLS** (class), **EL** (element), **PRO** (proprietor), **ATR** (attribute), **QUA**(qualia), **DEF** (definiendum), **STR** (strength), **INT** (intensivity).

Types. Next primitives are linguistic meaning types which serve for classification of meanings and for class inheritance.

[1] We came to the idea of planning and to the notion itself of discourse plan with a knowledge of a literature on first language acquisition in little children. In Dikovsky 2003 we state without details a cognitive hypothesis accounting for dynamic planning of meanings.

Primitive types make a finite lattice $(\mathbf{P}, \prec)$ containing four pairwise incomparable elementary types:
– the type of *nominators* **n** intuitively corresponding to "things" in the most general sense,
– the type of *sententiator* **s** intuitively corresponding to relations between "things"(actions / processes / events, etc.),
– the type of *qualifiers* **q**, intuitively corresponding to meanings qualifying nominators,
– the type of *circumscriptors* **c**, intuitively corresponding to meanings, qualifying qualifiers, sententiators, ad and circumscriptors themselves.

$\prec$ is an instance/generic partial order relation on primitive types. For example, the type of nominators **n** has the instances $\mathbf{n_a}$,$\mathbf{n_{\tilde{a}}} \prec \mathbf{n}$ of *animated/inanimated nominators*, the type of sententiators **s** has the instance $\mathbf{s_{om}} \prec \mathbf{s}$ of *oriented movement sententiators* (e.g., $\mathbf{run_1}$, **mount**), and also the instance $\mathbf{s_{bf}} \prec \mathbf{s}$ of *belief sententiators*, the type **q** of qualifiers has the instance $\mathbf{q_{cp}} \prec \mathbf{q}$ of *comparison qualifiers* (e.g., **better**, **worst**), and the instance $\mathbf{q_{qu}} \prec \mathbf{q}$ of *quantification qualifiers* (e.g., **all**, **neither**, **four**), the type **c** of circumscriptors has the instance $\mathbf{c_{dg}} \prec \mathbf{c}$ of *degree circumscriptors* (e.g., **more**, **especially**).

Let $\bot = \wedge\mathbf{P}$ be the *least* and $\top = \vee\mathbf{P}$ be the *greatest* primitive types.

Complex types. An important particularity of linguistic meaning types is that their definition includes options and iteration.

Option types. $\mathbf{O} = \{\mathbf{u^{(0)}} \mid \mathbf{u} \in \mathbf{P} \setminus \{\bot, \top\}\}$ is the set of *option types*.

Iterative types. $\mathbf{I} = \{\mathbf{u}^{(\omega)} \mid \mathbf{u} \in \mathbf{P} \setminus \{\bot, \top\}\}$ is the set of *iterative types*.

Basic types. $\mathbf{B} = \mathbf{P} \cup \mathbf{O} \cup \mathbf{I}$ is the set of *basic types*. The type instance order $\prec$ is naturally extended to $\mathbf{B} \setminus \{\bot, \top\} : \mathbf{u^{(0)}} \prec \mathbf{v^{(0)}}$ and $\mathbf{u}^{(\omega)} \prec \mathbf{v}^{(\omega)}$ for all $\mathbf{u} \prec \mathbf{v}$ in $\mathbf{B} \setminus \{\bot, \top\}$.

Being combined with nominators, the qualifiers give new nominators. Being combined with qualifiers, the circumscriptors form new qualifiers, and being combined with sententiators, they form new sententiators. From this follows the particularity, that all words have functional meaning types with an iterative subtype. For instance, common nouns have type $(\mathbf{q}^{(\omega)} \rightarrow \mathbf{n})$. They are interpreted by functions from lists of qualifier type meanings to nominator type meanings. Adjectives have type $(\mathbf{c}^{(\omega)} \rightarrow \mathbf{q})$. They are interpreted by functions from lists of circumscriptor type meanings to qualifier type meanings. Adverbs have type $(\mathbf{c}^{(\omega)} \rightarrow \mathbf{c})$. Their meanings are functions from lists of circumscriptor type meanings to circumscriptor type meanings. Non-functional meanings are of only two kinds: $\emptyset^{\mathbf{u^{(0)}}}$ (*empty meaning* of option type $\mathbf{u^{(0)}}$)

and $\mathbf{nil}^{\mathbf{u}^{(\omega)}}$ (*empty list* of type $\mathbf{u}^{(\omega)}$). This is why, we distinguish between iterative type and non-iterative type function arguments, the latter being named *actants*. For instance, the meaning of the majority of transitive verbs has two actants and the value of type **s**. Meanwhile, many words with meanings of other value types also have actants. For instance, the nominator, which is the meaning of the noun '$PART$', has two nominator type actors (*what*) and (*of-what*). So it has the functional type $(\mathbf{q}^{(\omega)}\mathbf{nn} \rightarrow \mathbf{n})$.

Notation. The expression $(\mathbf{u}_1\mathbf{u}_2...\mathbf{u}_k \rightarrow \mathbf{v})$ denotes the type $(\mathbf{u}_1 \rightarrow (\mathbf{u}_2 \rightarrow ... (\mathbf{u}_k \rightarrow \mathbf{v})...))$.

Meanings with actants are the first primitive meaning structures. We call them *situations*.

Situation types. There are four families of *situation types*:
$\mathbf{T^S} = \{(\mathbf{c}^{(\omega)}\mathbf{u}_1...\mathbf{u}_r \rightarrow \mathbf{s}) \mid \mathbf{u}_i \in \mathbf{P} \cup \mathbf{O}, r \geq 0\}$,
$\mathbf{T^n} = \{(\mathbf{q}^{(\omega)}\mathbf{u}_1...\mathbf{u}_r \rightarrow \mathbf{n}) \mid \mathbf{u}_i \in \mathbf{P} \cup \mathbf{O}, r \geq 0\}$,
$\mathbf{T^q} = \{(\mathbf{c}^{(\omega)}\mathbf{u}_1...\mathbf{u}_r \rightarrow \mathbf{q}) \mid \mathbf{u}_i \in \mathbf{P} \cup \mathbf{O}, r \geq 0\}$,
$\mathbf{T^c} = \{(\mathbf{c}^{(\omega)}\mathbf{u}_1...\mathbf{u}_r \rightarrow \mathbf{c}) \mid \mathbf{u}_i \in \mathbf{P} \cup \mathbf{O}, r \geq 0\}$.
$\mathbf{T} = \mathbf{B} \cup \mathbf{T^S} \cup \mathbf{T^n} \cup \mathbf{T^q} \cup \mathbf{T^c}$ is the set of *types*.

Situations are defined in the dictionary by *situation profiles* consisting of the situation's key, of its type, of the basic lexeme and of roles of its actants. For instance, the situation, which is the meaning of the verb lexeme 'PUT_1', has profile

$$\mathbf{sit}(\mathbf{put_1}^{(\mathbf{c}^{(\omega)}\mathbf{n_a}\mathbf{nc} \rightarrow \mathbf{s})}, \text{'}PUT_1\text{'}(\mathbf{AGT}(1), \mathbf{PAT}(2), \mathbf{LOC}(3)))$$

identified by the key $\mathbf{put_1}$, stating that this situation has three obligatory actants: the first actant of type $\mathbf{n_a}$ has the role **AGT**, the second of type **n** has the role **PAT**, the third of type **c** has the role **LOC**. Besides them, it has the standard iterative circumscriptor-type argument. This situation has value type **s**.

In order to be used in complex meanings, the dictionary situations must be abstracted, i.e. transformed into Lambda terms of the corresponding types. For instance, the profile of $\mathbf{put_1}$ induces the Lambda term[2]:

$$\lambda Y^{\mathbf{c}^{(\omega)}} X_1^{\mathbf{n_a}} X_2^{\mathbf{n}} X_3^{\mathbf{c}}.\ \mathbf{put}_1(Y, X_1, X_2, X_3).$$

5.3 Discourse Plans

Building complex meanings from abstracted primitive situations meets with the following three obstacles. First is that the type of the actant of the host situation which should be developed using a new situation may conflict with the value type of the latter. For instance, in order to

[2]We abbreviate $\lambda X_1.\ (...(\lambda X_n.\ X_n.\ t)...)$ by $\lambda X_1...X_n.\ t$.

use a sententiator-type meaning as a qualifier-type argument of a nominator (cf. relative clauses modifying names), the type transformation ($\mathbf{s} \Rightarrow \mathbf{q}$) should be effected somehow. The second problem is more complex and is often due to the intended communicative organization of the complex meaning. It arises when the new situation should be a semantic derivative of some "clue" situation, as it is the case of the voices of verbs. In order to convert the original (i.e. available in the dictionary) situation into its derivative necessary to express a specific communicative structure, some semantic analogues are needed of the voice diatheses (Mel'čuk and Holodovich 1970). The third problem has been the subject of long debate in the literature (see (van Benthem and ter Meulen 1997)). This is the problem of using references to a meaning unit outside of the scope of its definition. For instance, in the example of Geach:

If a farmer has a donkey, he beats it

the anaphorical pronouns corresponding to the actants of situation *beat* reference the corresponding actants of situation *have*, whereas their scopes are disjoint. More complex are implicit tense relations, as in the following Kamp's sentence:

A child was born that will become ruler of the world.

So on the one hand, there should be a structure specifying the intended composition of primitive meanings, and on the other hand, there must be some means of realization of the meaning specifications by complex meanings. The specifications, we call them *discourse plans (DPs)*, are defined by the grammar in Fig. 1.

A DP can be seen as a sequence of *plan points* and also as a hierarchy of *sub-plans*. These two orders induce the corresponding mixed order of sub-plans. Each plan point introduces into DP a new primitive, or a new aggregate, or a situation's derivative defined by a situation converter. They all can be referenced. Long term references are kept in global context Γ, which is never updated. Short-term references added at some points of DP to local context ρ can be used and deleted from ρ later in DP. The scope and visibility of local references in the DP is defined in terms of the mixed order on sub-plans.

Primitives are of four kinds:

(p_1) Empty lists of qualifiers and circumscriptors $\mathbf{nil}^{\mathbf{q}^{(\omega)}}$, $\mathbf{nil}^{\mathbf{c}^{(\omega)}}$ and null values of option types: $\emptyset^{\mathbf{u}^{(\mathbf{0})}}$ (both omitted in examples).

(p_2) Actant-less lexical units, for instance, common nouns such as $`CAT'^{(\mathbf{q}^{(\omega)} \rightarrow \mathbf{n_a})}$.

(p_3) Embedded relations, e.g. tense relations.

(p_4) Context access operators:

DP	::=	$\mathbf{SP}^+$	(discourse plan)
SP	::=	**Prim** ∥	(sub-plan)
		$\mathbf{A_{op}}\{\mathbf{AComps}\}$ ∥	(aggregate)
		Key Conv Mode {**DOs**}	(situation)
DOs	::=	**DO** ∥ **DO**, **DOs**	(development operators)
DO	::=	**Arg** ⇐ **SP**	(development operator)
AComps	::=	**SP**, **SP** ∥ **SP**, **AComps**	(aggregate components)
Prim			(primitive)
Conv			(converter)
Mode			(intonation marker)
Key			(situation identifier)
Arg			(situation argument)
op			(aggregation operator)

FIGURE 1 DP syntax

$(\Uparrow_x)^{\mathbf{u}}$ co-referential with an element of type **u** in the local context;
$(\Downarrow_x^{\mathbf{u}}\ LEX)^{\mathbf{u}}$ creating and adding to the local context a new reference x of type **u** to the lexical unit LEX;
$(\Uparrow^{\mathbf{g}}\ N)^{\mathbf{u}}$ accessing the global context element of type **u** identified by N;
$(\Uparrow_x^{\mathbf{g}}\ N)^{\mathbf{u}}$ creating and adding to the local context a reference x to the global context element identified by N (x has the type of N).

Converters are fundamental means of planning. A converter **conv** used in a development operator $\mathbf{Arg} \Leftarrow \mathbf{sub-plan}$, in which

$$\mathbf{sub-plan} = \mathbf{key\ conv\ mode}\ \{\mathbf{sub-plans}\},$$

determines a semantic derivative $\mathbf{der}(\mathbf{key}, \mathbf{conv})$ of situation **key** to be composed in (i.e. to unfold) the argument Arg of the host situation. So it is a second order operator applied to situation's meaning. For any two types $\mathbf{u}, \mathbf{v} \in \mathbf{T}$, $(\mathbf{u} \Rightarrow \mathbf{v})$ is a *converter type*. Below, we outline three kinds of converters: *TR-Converters, abstractors* and *direct diatheses*.

TR-converters relate to situations and to primitives their inherent attributes determined by roles and constrained by types. Intuitively, the role **R** of a **uR**-converter applied to a DP element E determines an inherent attribute of the meaning of E. The type **u** of this converter is one of the possible types of values of this attribute. For instance, each situation with **s**-type value has a semantic tense attribute **TNS**, and each nominator expressing a physical body has a location attribute **LOC**. Respectively, the tense TR-converter has the form $(\ )^{\mathbf{u}\ \mathbf{TNS}}$,

where $\mathbf{u}$ is a tense circumscriptor type (e.g. $\mathbf{t_{ntr}} \prec \mathbf{t}$: neutral (gnomical) tense, $\mathbf{t_{pnt}} \prec \mathbf{t}$: pointwise tense, $\mathbf{t_{int}} \prec \mathbf{t}$: interval tense), and the location TR-converter has the form $(\ )^{\mathbf{v\ LOC}}$, where $\mathbf{v}$ is a location circumscriptor type. For example, let ρ contain a reference s_2 to the situation:

$$\mathbf{sit}(\mathbf{be_3}^{(\mathbf{c}^{(\omega)}\mathbf{nn} \rightarrow \mathbf{s})}, \text{`}BE_3\text{'}(\mathbf{EL}(1), \mathbf{CLS}(2))).$$

Then in the DP fragment:

$[t]$ $\quad \mathbf{t_{prg}} \Leftarrow \Downarrow_{t_1}^{\mathbf{t_{prg}}} (\Uparrow s_2)^{\mathbf{t_{prg}}\ \mathbf{TNS}},$

$[t+1]$ $\quad \mathbf{t} \Leftarrow ((\Uparrow^{\mathbf{g}}\ \mathbf{now}) \in t_1)^{\mathbf{t}}, \quad \%\ (\Uparrow^{\mathbf{g}}\ \mathbf{now})$: the moment of speech

TR-converter $(\Uparrow s_2)^{\mathbf{t_{prg}}\ \mathbf{TNS}}$ defines the tense attribute of situation $\mathbf{be_3}$ with value type $\mathbf{t_{prg}}$. At point $[t]$, the operator $\Downarrow_{t_1}^{\mathbf{t_{prg}}} (\Uparrow s_2)^{\mathbf{t_{prg}}\ \mathbf{TNS}}$, adds to ρ the reference t_1 to this attribute. Besides this, at this point, $(\Uparrow s_2)^{\mathbf{t_{prg}}\ \mathbf{TNS}}$ becomes a circumscriptor of $\mathbf{be_3}$ of type $\mathbf{t_{prg}}$. At the next point, the reference t_1 is used to introduce a new circumscriptor of $\mathbf{be_3}$, which states the semantic present tense.

Abstractor (**abs**^{x}) has the approximate meaning "*such x that*". Abstractors are used at the plan points of the form:

$$\mathbf{q} \Leftarrow \mathbf{key_1}^{(\bar{\mathbf{u}} \rightarrow \mathbf{v})}\ \mathbf{abs}^x\ \{\mathbf{sub-plans}\},$$

where the situation $\mathbf{key}_1$ develops a qualifier type argument of its host situation $\mathbf{key}_2$. So $\mathbf{abs}^x$ describes type conversions ($\mathbf{v} \Rightarrow \mathbf{q}$).

Direct diathesis (**dth**) determines how the type and the argument structure of the introduced situation should be adapted to the developed argument type and role and to the new positions and roles of its actants. For instance, applied to situation:

$\mathbf{sit}(\mathbf{bear_2}^{(\mathbf{c}^{(\omega)}\mathbf{n_a}\mathbf{n_a} \rightarrow \mathbf{s})}, \text{`}GIVE\_BIRTH\text{'}(\mathbf{AGT}(1)^{\mathbf{n_a}}, \mathbf{PAT}(2)^{\mathbf{n_a}}))$,

the diathesis $\mathbf{dth_1}(\mathbf{bear_2}) = (\mathbf{PAT}(2)^{\mathbf{n_a}}, \mathbf{AGT}(1)^{\mathbf{n_a^{(0)}}})^{\mathbf{s\ EVT}}$ determines the semantic derivative: $\mathbf{der}(\mathbf{bear_2}, \mathbf{dth_1}) =$

$\mathbf{sit}(\mathbf{be\_born}^{(\mathbf{c}^{(\omega)}\mathbf{n_a}\mathbf{n_a^{(0)}} \rightarrow \mathbf{s})}, \text{`}GIVE\_BIRTH\text{'}(\mathbf{PAT}(2), \mathbf{AGT}(1)))$.

Applied to situation:

$\mathbf{sit}(\mathbf{rule_1}^{(\mathbf{c}^{(\omega)}\mathbf{n_a}\mathbf{n} \rightarrow \mathbf{s})}, \text{`}RULE_1\text{'}(\mathbf{AGT}(1)^{\mathbf{n_a}}, \mathbf{PAT}(2)^{\mathbf{n}}))$,

the diathesis $\mathbf{dth_2}(\mathbf{rule_1}) = (\mathbf{PAT}(2)^{\mathbf{n^{(0)}}})^{\mathbf{n_a\ AGT}}$ determines:

$\mathbf{der}(\mathbf{dth_2}, \mathbf{rule_1}) = \mathbf{sit}(\mathbf{ruler\_of}^{(\mathbf{q}^{(\omega)}\mathbf{n} \rightarrow \mathbf{n_a})}, \text{`}RULE_1\text{'}(\mathbf{PAT}(2)))$.

In general, a direct diathesis

$$\mathbf{dth}(\mathbf{key}) = (\mathbf{R'_1}(j_1)^{\mathbf{u'_1}}, ..., \mathbf{R'_k}(j_k)^{\mathbf{u'_k}})^{\mathbf{v'\ R}}$$

applied to a situation **key** with profile

$$\mathbf{sit}(\mathbf{key}^{(\mathbf{u}^{(\omega)}\mathbf{u}_1 ... \mathbf{u}_n \rightarrow \mathbf{v})}, \text{`}LEXEME\text{'}(\mathbf{R_1}(1), ..., \mathbf{R_n}(n)))$$

determines a derivative with profile

$$\mathbf{sit}(\mathbf{key}'^{(\mathbf{u}'^{(\omega)}\mathbf{u}'_1...\mathbf{u}'_k \to \mathbf{v}')}, \text{`}LEXEME\text{'}(\mathbf{R}'_{\mathbf{1}}(j_1), ..., \mathbf{R}'_{\mathbf{k}}(j_k))).$$

This derivative is introduced in sub-plan of type $\mathbf{v}'$ for role $\mathbf{R}$ by development operators:

$[t_1]$ $\quad \mathbf{Arg} \Leftarrow \mathbf{key}^{(\mathbf{u}^{(\omega)}\mathbf{u}_1...\mathbf{u}_n \to \mathbf{v})}$ **dth** {

...

$[t_2]$ $\quad (j_1)^{\mathbf{u}'_1} \Leftarrow \{sub{-}plan_1\}^{\mathbf{u}'_1\mathbf{R}'_1}$,

...

$[t_3]$ $\quad (j_k)^{\mathbf{u}'_k} \Leftarrow \{sub{-}plan_k\}^{\mathbf{u}'_k\mathbf{R}'_k}$ }

where each $\{sub{-}plan_i\}^{\mathbf{u}'_i\ \mathbf{R}'_i}$ is a sub-plan of type $\mathbf{u}'_\mathbf{i}$ for role $\mathbf{R}'_\mathbf{i}$.

Co-reference and scope. ρ is a bounded resource memory. Adding to ρ a local reference may cause deletion of some other references. Reference deletion may occur not only for the reason of bounded memory size, but also because this memory cannot keep at the same plan point two "similar" "independent" references. We propose the following notion of reference similarity. When a local reference x is created in expression $\Downarrow_x^{\mathbf{u}}$ **key Conv** $\{Args\}$, it is assigned the *format* $fm(x) = (\mathbf{u}, \mathbf{R})$, where $\mathbf{u}$ is the type of x and $\mathbf{R}$ is the role assigned to the sub-plan **key Conv** $\{Args\}$ by the development operator in which it is used. If the role is not assigned, this will be the most general *undefined role* $\Re$. A reference x is *visible* at point $[i]$ of DP if $x \in \rho$ at this point. Visibility is determined by the following rules ($\leq_c$ is the infix order of sub-plans):

($\mathbf{v_1}$) There is a constant κ limiting the number of visible references in ρ Adding to ρ $(\kappa+1)$-th reference causes **deletion** from ρ of the earliest[3] reference.

($\mathbf{v_2}$) Expressions

$$\Downarrow_x^{\mathbf{u}} \mathbf{key\ Conv}\ \{\mathbf{Args}\} \text{ and } (\Uparrow_x^{\mathbf{g}} \mathbf{key}^{\mathbf{u}})$$

add the reference x [4] to ρ and at the same time **remove** from ρ every similar local reference y, on which x does not c-depend: $y \not\leq_c x$.

($\mathbf{v_3}$) Local context access/update operators are executed in the order of the plan points where they are used.

($\mathbf{v_4}$) In operators $\mathbf{q} \Leftarrow \mathbf{SP}$, $\mathbf{c} \Leftarrow \mathbf{SP}$ developing in DP π a qualifier/circumscriptor argument of a situation (of a primitive) S, every reference $\Uparrow x$ used in **SP** is c-dependent only on those elements in π on which S c-depends (in particular, on S itself).

($\mathbf{v_5}$) A reference to a situation argument is removed together with the references to all this argument's qualifiers/circumscriptors.

[3] I.e. with the least DP point number.

[4] Never used at the preceding plan points.

Communicative structure. Semantic diatheses are related with communicative structure. Namely, to become the *theme* of a situation S, its actant should be promoted to the first actant position in a derivative of S. Somewhat simplifying, the *rheme* of S will be the set of all other *locally referenced arguments* in the derivative. Intuitively, the rheme consists of those arguments related with the theme through S, which can be passed as parameters to situations introduced later in the DP. Another communicative structure component, which can also entail a diathesis, is the (main) focus. In this paper, somewhat simplifying the real facts, we mark by the focus $\odot$ a unique DP point. The DP element introduced at the focalized point must be locally referenced.

Let us plan the meaning of the Kamp's sentence using the two diatheses above.

Discourse plan π:

$$
\begin{array}{ll}
[1] & \Downarrow^{\mathbf{s}}_{s_1} \mathbf{bear_2}^{(\mathbf{c}^{(\omega)}\mathbf{n_a}\mathbf{n_a} \to \mathbf{s})}\ \mathbf{dth_1}\ \{ \\
[2] & \quad \mathbf{t_{pnt}} \Leftarrow \Downarrow^{\mathbf{t_{pnt}}}_{t_1} (\Uparrow s_1)^{\mathbf{t_{pnt}}}\ \mathbf{TNS}, \\
[3] & \quad \mathbf{t} \Leftarrow (\Uparrow t_1 \triangleleft (\Uparrow^{\mathbf{g}}\ \mathbf{now}))^{\mathbf{t}}, \qquad \% \text{ before now} \\
[4]\odot & \quad (2)^{\mathbf{n_a}} \Leftarrow \Downarrow^{\mathbf{n_a}}_{n_1} \text{`}CHILD\text{'}^{(\mathbf{q}^{(\omega)} \to \mathbf{n_a})}\{ \\
[5] & \qquad \mathbf{q} \Leftarrow \Downarrow^{\mathbf{q}}_{q_1} \Downarrow^{\mathbf{s}}_{s_2} \mathbf{be_3}^{(\mathbf{c}^{(\omega)}\mathbf{nn} \to \mathbf{s})}\ \mathbf{abs}^{n_1}\ \{ \\
[6] & \qquad\quad \mathbf{t_{pnt}} \Leftarrow \Downarrow_{t_2} (\Uparrow s_2)^{\mathbf{t_{pnt}}}\ \mathbf{TNS}, \\
[7] & \qquad\quad \mathbf{t} \Leftarrow (\Uparrow t_1 \triangleleft \Uparrow t_2)^{\mathbf{t}}, \qquad \% \ t_2 \text{ after } t_1 \\
[8] & \qquad\quad (1)^{\mathbf{n_a}} \Leftarrow (\Uparrow n_1)^{\mathbf{n_a}}, \\
[9] & \qquad\quad (2)^{\mathbf{n}} \Leftarrow \Downarrow^{\mathbf{n}}_{n_2} \mathbf{rule}^{(\mathbf{c}^{(\omega)}\mathbf{n_a}\mathbf{n} \to \mathbf{s})}\ \mathbf{dth_2}\ \{ \\
[10] & \qquad\qquad (2)^{\mathbf{n}} \Leftarrow \text{`}WORLD_1\text{'}^{(\mathbf{q}^{(\omega)} \to \mathbf{n})} \\
 & \quad \}\ \ \}\ \ \}\ \ \}
\end{array}
$$

This DP is consistent in the following sense:

Definition 1 *A DP is* s(cope)-consistent *if all local references used in this plan are visible at the points, where they are used* [5].

Moreover, π is *realizable*, in the sense that all the planned diatheses have the corresponding semantic derivatives in English (see above).

5.4 Meanings

Meanings are Lambda terms derived from realizable DPs by means of the following simple translation τ defined by induction on sub-plans S.

[5] There is a real-time algorithm checking this property.

Definition 2 $(\mathbf{m_1})$ $\tau(S) = S$ *if* S *is a primitive sub-plan.*
$(\mathbf{m_2})$ $\tau(\odot S) = \odot\tau(S)$.
$(\mathbf{m_3})$ $\tau((S_1, S_2)) = (\tau(S_1), \tau(S_2))$.
$(\mathbf{m_4})$ $\tau(\mathbf{A_{op}}\{Comps\}) = \mathbf{A_{op}}\{\tau(Comps)\}$.
$(\mathbf{m_5})$ *Let* S *be a sub-plan of the form:*

$[t_0]$ **key Converter Mode** {
$[t_1]$ $\mathbf{u}_1 \Leftarrow S_{0,1}$,
...
$[t_l]$ $\mathbf{u}_l \Leftarrow S_{0,l}$,
$[t_m]$ $(j_1)^{\mathbf{v}_1} \Leftarrow S_1$,
...
$[t_k]$ $(j_k)^{\mathbf{v}_k} \Leftarrow S_k$ }

and let

$$\mathbf{der}(\mathbf{key}, \mathbf{Converter}) = \mathbf{key_{der}}^{(\mathbf{u}^{(\omega)}\mathbf{v}_1 \ldots \mathbf{v}_n \to \mathbf{v})},\ \mathbf{u}_m \preceq \mathbf{u}\ (1 \le m \le l).$$

be the derivative determined in the dictionary for **key** *by the* **Converter**. *Then, using the abstract situation*

$$absder(\mathbf{key}, \mathbf{Converter}) = \lambda X_0^{\mathbf{u}^{(\omega)}} X_1^{\mathbf{v}_1} \ldots X_k^{\mathbf{v}_k}.\ \mathbf{key_{der}}(X_0, X_1, ..., X_k),$$

the translation of S *is defined as :*

$$\tau(S) = \mathbf{sit\ Mode}\ (absder(\mathbf{key}, \mathbf{Converter})\ [\tau(S_{k+1,1}), ..., \tau(S_{k+1,l})], \mathbf{act}_1(\tau(S_1)), ..., \mathbf{act}_k(\tau(S_k))).$$

The meaning derived from a realizable DP π *is the normal form term* M *such that* $\tau(\pi) \twoheadrightarrow M$ *(see below the point* **Reducibility***).*

Proposition 1 *1. If* π *is a realizable DP, then there exists a Lambda term* $\tau(\pi)$. *Moreover, for each sub-plan* S *of* π *of a primitive type* $\mathbf{u}$, $\tau(S)$ *also has type* $\mathbf{u}$.
2. If π *is s-consistent, then the meaning derivable from* π *is also s-consistent.*

The meaning of the sentence of Kamp derivable from π has the form:

(1) $\Downarrow_{s_1}^{\mathbf{s}}\ \mathbf{sit}(\mathbf{be\_born}^{(\mathbf{c}^{(\omega)}\mathbf{n_a}\mathbf{n_a^{(0)}} \to \mathbf{s})}$
(2) $[(\Downarrow_{t_1}^{\mathbf{t_{pnt}}} \Uparrow s_1)^{\mathbf{t_{pnt}}}$,
(3) $(\Uparrow_{t_1} \triangleleft (\Uparrow^{\mathbf{g}}\ \mathbf{now}))^{\mathbf{t}}]^{\mathbf{c}^{(\omega)}}$,
(4)$\odot$ $\mathbf{act}_2(\Downarrow_{n_1}^{\mathbf{n_a}}\ `CHILD'^{(\mathbf{q}^{(\omega)} \to \mathbf{n_a})}($
(5) $[\Downarrow_{q_1}^{\mathbf{q}} \Downarrow_{s_2}^{\mathbf{s}}\ \mathbf{sit}(\mathbf{der}(\mathbf{be_3}, \mathbf{abs}^{n_1})$
(6) $[(\Downarrow_{t_2}^{\mathbf{t_{pnt}}} \Uparrow s_2)^{\mathbf{t_{pnt}}}$,

(7) $(\Uparrow_{t_1} \triangleleft \Uparrow_{t_2})^{\mathbf{t}}]\mathbf{c}^{(\omega)}$,
(8) $\mathbf{act}_1((\Uparrow n_1)^{\mathbf{n_a}})$,
(9) $\mathbf{act}_2(\Downarrow^{\mathbf{n}}_{n_2}\ \mathbf{sit}(\mathbf{ruler\_of}^{(\mathbf{q}^{(\omega)}\mathbf{n} \rightarrow \mathbf{n_a})}$
(10) $\mathbf{act}_2(\text{`}WORLD_1\text{'}^{(\mathbf{q}^{(\omega)} \rightarrow \mathbf{n})})$
$))\quad)]^{\mathbf{q}^{(\omega)}}\quad))\quad)$

Language of linguistic meanings $\mathcal{MT}^{\mathbf{lt}}$ (for space reasons, we don't include aggregates, modes, roles and foci).

Variables. We suppose that there are disjoint countable sets $\mathbf{V}^{\mathbf{u}}$ of *object variables* of type $\mathbf{u}$, $\mathbf{u} \in (\mathbf{B} \setminus \{\top\})$, and $\mathbf{R}^{\mathbf{u}}$ of *context variables* of type $\mathbf{u}$, for $\mathbf{u} \in \mathbf{T}$. We use upper-case latin indexed letters for the former and lower-case latin indexed letters for the latter.

Constants. For each type $\mathbf{u} \in \mathbf{T}$, $\mathbf{C}^{\mathbf{u}}$ is a countable set of *constants* of type $\mathbf{u}$. For each primitive type $\mathbf{u} \in (\mathbf{P} \setminus \{\bot, \top\})$, $\mathbf{C}^{\mathbf{u}^{(\mathbf{0})}} = \{\emptyset^{\mathbf{u}^{(\mathbf{0})}}\}$ ($\emptyset^{\mathbf{u}^{(\mathbf{0})}}$ is the *empty element* of type $\mathbf{u}^{(\mathbf{0})}$) and $\mathbf{C}^{\mathbf{u}^{(\omega)}} = \{\mathbf{nil}^{\mathbf{u}^{(\omega)}}\}$ ($\mathbf{nil}^{\mathbf{u}^{(\omega)}}$ is the *empty list* of type $\mathbf{u}^{(\omega)}$).

Converters. For each converter type $\mathbf{v} = (\mathbf{u}_1 \Rightarrow \mathbf{u}_2), \mathbf{u}_1, \mathbf{u}_2 \in \mathbf{T}$, $\mathbf{\Omega}^{\mathbf{V}}$ is a countable set of *converter constants* of converter type $\mathbf{v}$.

Context operators. $\mathbf{\Phi}$ is a countable set of *context operator names*.

Terms. The set $\mathcal{MT}^{\mathbf{lt}} =_{df} \bigcup\limits_{\mathbf{u} \in \mathbf{T}} \mathcal{T}^{\mathbf{u}}$ of typed terms is the least set verifying the following conditions:

($\mathbf{t_0}$) If $t = X \in \mathbf{V}^{\mathbf{u}}$, then $t \in \mathcal{T}^{\mathbf{u}}$ and $FV(t) = \{X\}$.
($\mathbf{t_1}$) If $\mathbf{k} \in \mathbf{C}^{\mathbf{u}}$, then $\mathbf{k} \in \mathcal{T}^{\mathbf{u}}$ and $FV(\mathbf{k}) = \emptyset$.
($\mathbf{t_2}$) If $\phi \in \mathbf{\Phi}$ and $x \in \mathbf{R}^{\mathbf{u}}$, then $t = \phi_x \in \mathcal{T}^{\mathbf{u}}$ and $FV(t) = \emptyset$.
($\mathbf{t_3}$) If $\mathbf{u}^{(0)} \in \mathbf{O}$, then $\mathcal{T}^{\mathbf{u}} \cup \{\emptyset^{\mathbf{u}^{(\mathbf{0})}}\} \subseteq \mathcal{T}^{\mathbf{u}^{(\mathbf{0})}}$ and $FV(\emptyset^{\mathbf{u}^{(\mathbf{0})}}) = \emptyset$.
($\mathbf{t_4}$) If $\mathbf{u} \in \mathbf{P}$ and $\mathbf{u}_1, ..., \mathbf{u}_k \preceq \mathbf{u}$, $k > 0$, then:
(*i*) $\mathbf{nil}^{\mathbf{u}^{(\omega)}} \in \mathcal{T}^{\mathbf{u}^{(\omega)}}$, $FV(\mathbf{nil}^{\mathbf{u}^{(\omega)}}) = \emptyset$, and
(*ii*) $t = [t_1, ..., t_k] \in \mathcal{T}^{\mathbf{u}^{(\omega)}}$ and $FV(t) = \bigcup\limits_{i=1}^{k} FV(t_i)$ for any $t_1 \in \mathcal{T}^{\mathbf{u}_1}, ..., t_k \in \mathcal{T}^{\mathbf{u}_k}$, such that $FV(t_i) \cap FV(t_j) = \emptyset$, $1 \leq i \neq j \leq k$.
($\mathbf{t_5}$) If $\gamma \in \mathbf{\Omega}^{(\mathbf{u} \Rightarrow \mathbf{v})}$ and $t \in \mathcal{T}^{\mathbf{u}}$, then $t_1 = \gamma\{t\} \in \mathcal{T}^{\mathbf{V}}$ and $FV(t_1) = FV(t)$.
($\mathbf{t_6}$) If $t_0 \in \mathcal{T}^{(\mathbf{u} \rightarrow \mathbf{v})}$, $t_1 \in \mathcal{T}^{\mathbf{u}_1}$ for some $\mathbf{u}_1 \preceq \mathbf{u}$, and $FV(t_0) \cap FV(t_1) = \emptyset$, then $t = (t_0\ t_1) \in \mathcal{T}^{\mathbf{V}}$ and $FV(t) = FV(t_0) \cup FV(t_1)$.
($\mathbf{t_7}$) If $t_0 \in \mathcal{T}^{\mathbf{V}}$, $(\mathbf{u} \rightarrow \mathbf{v}) \in \mathbf{T} \setminus \mathbf{B}$, and $X \in \mathbf{V}^{\mathbf{u}} \cap FV(t_0)$, then $t = \lambda X.\ t_0 \in \mathcal{T}^{(\mathbf{u} \rightarrow \mathbf{v})}$ and $FV(t) = FV(t_0) \setminus \{X\}$. □

Let $t_0 \in \mathcal{T}^{\mathbf{u}}$ be a subterm of a term $t = C[t_0] \in \mathcal{T}^{\mathbf{V}}$ identified by the *typed context* $C[\ ]^{\mathbf{u}}$. Given some other term $t_1 \in \mathcal{T}^{\mathbf{u}}$, if $C[t_1]$, i.e. the result of replacement of t_0 by t_1 in $C[\ ]^{\mathbf{u}}$, does not violate the convention of free variables in the definition of terms then, clearly, $C[t_1] \in \mathcal{T}^{\mathbf{V}}$. We say that a binary relation R on terms is *closed under typed contexts*, if

$t_1\ R\ t_2$ and $C[t_1] \in \mathcal{T}^{\mathbf{V}}$ implies $C[t_2] \in \mathcal{T}^{\mathbf{V}}$ and $C[t_1]\ R\ C[t_2]$.

Reducibility. The *immediate reducibility* of terms being the relation

$$(\lambda X^{\mathbf{u}}.\ t_1)\ t_2 \to_\beta t_1[t_2/X]$$

between terms $t = (\lambda X^{\mathbf{u}}.\ t_1)\ t_2$ and $t_0 = t_1[t_2/X]$, $t, t_0 \in \mathcal{T}^{\mathbf{v}}$, such that $\lambda X^{\mathbf{u}}.\ t_1 \in \mathcal{T}^{(\mathbf{u} \to \mathbf{v})}$, $t_2 \in \mathcal{T}^{\mathbf{u}_1}$ for some $\mathbf{u}_1 \preceq \mathbf{u}$, and $X \in \mathbf{V}^{\mathbf{u}} \cap FV(t_1)$, the *reducibility relation* $\twoheadrightarrow$ is the least preorder containing $\to_\beta$ and closed under typed contexts $C[\,]^{\mathbf{u}}$ and renaming of bound variables. A term t_0 is a *normal form* of a term t if $t \twoheadrightarrow t_0$ and t_0 is $\twoheadrightarrow$-minimal.

By the classical Church-Rosser theorem (see (Barendregt 1981)), the reducibility $\twoheadrightarrow$ is confluent and terminal. Then it is not difficult to prove that each term $t \in \mathcal{T}^{\mathbf{u}}$ has a unique normal form $t_0 : t \twoheadrightarrow t_0$ and $t_0 \in \mathcal{T}^{\mathbf{u}}$.

5.5 Discussion

The basic difference between the classical logical type system and that of the linguistic meanings is in the way the adnominal/adverbial modifiers are treated. The classical way (which can be traced back to Aristotle) is to interpret them as properties or, better to say, as conjunctive constraints to the intention of the modified unit: $|(mod\ U)| = |mod| \wedge |U|$. This is why, they obtain functional recursive types $(T/T), (T\backslash T)$. Through the Curry-Howard isomorphism, the meanings in the conventional system are isomorphic to derivations in formal systems, in which implication elimination corresponds to the function application and implication introduction corresponds to Lambda-abstraction. This leads directly to the Lambek calculus and its generalizations (see Buszkowski 1997). The iterative and non-recursive type system of linguistic meanings syntactically separates the linguistic semantics stricto sensu (argument structure, inheritance, semantic dependencies, co-reference, communicative structure) from the property / relation based factual semantics. At the same time, it can serve as a formal interface between the two semantics:

$$|(Nominator^{(\mathbf{q}^{(\omega)} \to \mathbf{n})}\ [m_1, \ldots, m_k]^{\mathbf{q}^{(\omega)}})|_{fact} =$$
$$\mathbf{modify}(|Nominator|_{fact}, [|m_1|_{fact}, \ldots, |m_k|_{fact}]),$$

where:

$$\begin{array}{lcl} \mathbf{modify}(N, \mathbf{nil}) & = & N \\ \mathbf{modify}(N, [M|R]) & = & M\ \wedge\ \mathbf{modify}(N, R). \end{array}$$

On the other hand, the linguistic meaning types are directly translated into the corresponding semantic dependencies. This translation can be described in the form of the archetype dependency grammar in Fig. 2, in which the underlined nonterminals correspond to the heads. The dependencies defined by this grammar have the same orientation as the corresponding surface syntax dependencies. Due to this direct translation, the linguistic meaning can be naturally linked with the surface

s	→	$\underline{\mathbf{verb}} \oplus \{\mathbf{arg} \oplus\}^i \mathbf{c} \quad (1 \leq i \leq 5)$
arg	→	**n** | **q** | **c**
n	→	$\mathbf{q} \oplus \underline{\mathbf{n}}$ | **name**
q	→	$\mathbf{c} \oplus \underline{\mathbf{q}}$ | **adj**
c	→	$\mathbf{c} \oplus \underline{\mathbf{c}}$ | **adv**

FIGURE 2 Archetype grammar ($\oplus$ stands for the standard word order, which is a language dependent parameter).

dependency syntax. Such direct translation is impossible for classical logical types because the dependencies corresponding to them are oriented from modifiers to modified units [6].

One more aspect relating the linguistic meaning with some existing theories of linguistic semantics is its *dynamicity* (cf. Muskens et al. (1997)). There is however a great difference between the way the planned linguistic meaning transforms local contexts and the way the grammatical sentences of DRT Kamp (1981) transform boxes. The former only establishes anaphorical bindings and communicative structure relations between meaning units in disjoint sub-plans, whereas the latter construct and updates models. So the DRT-like dynamic systems are destined for knowledge acquisition through linguistic form. As such, they can, for instance, be a means of the global context control, which we don't take into account. Another difference between the two models is in the nature of the dynamism. In the planned linguistic meaning model, the local context control is resource sensitive, whereas in the factual dynamic systems there are no dynamic memory size limits.

We think that it is more appropriate from the linguistic point of view to define logical semantics (be it type logical, or game-theoretic, or DRT, etc.) on top of the linguistic meanings than to do it directly from sentences or from their syntactic structures.

APPENDIX: Set Theoretic Semantics of $\mathcal{MT}^{\mathbf{lt}}$

Domains. For each primitive type $\mathbf{u} \in \mathbf{P}$, let a domain $D^{\mathbf{u}}$ be chosen so that the following conditions were met:

($\mathbf{d_1}$) $D^{\perp} = \{\varepsilon\}$ for some object ε.

($\mathbf{d_2}$) If $\mathbf{u} \prec \mathbf{v}$, then $D^{\mathbf{u}} \subset D^{\mathbf{v}}$.

($\mathbf{d_3}$) $D^{\top} = \bigcup_{\mathbf{u} \prec \top} D^{\mathbf{u}}$.

For each option type $\mathbf{u^{(0)}}$, a *null element* $\epsilon^{\mathbf{u^{(0)}}}$ of type $\mathbf{u^{(0)}}$ is selected

[6] The natural dependencies can be simulated through proofs in multimodal extensions of Lambek calculus Moortgat 1997.

and for each iterative type $\mathbf{u}^{(\omega)}$, a special object $[\,]^{\mathbf{u}^{(\omega)}}$ is selected called *empty list of type* $\mathbf{u}^{(\omega)}$, and the domains of option and iterative types are defined by:

($\mathbf{d_4}$) If $\mathbf{u}^{(\mathbf{0})} \in \mathbf{O}$, then $D^{\mathbf{u}^{(\mathbf{0})}} = D^{\mathbf{u}} \cup \{\epsilon^{\mathbf{u}^{(\mathbf{0})}}\}$.

($\mathbf{d_5}$) If $\mathbf{u}^{(\omega)} \in \mathbf{I}$, then $D^{\mathbf{u}^{(\omega)}} = \mathbf{list}(D^{\mathbf{u}})$, where $\mathbf{list}(D^{\mathbf{u}})$ is the set of all finite lists of objects in $D^{\mathbf{u}}$ with $[\,]^{\mathbf{u}^{(\omega)}}$ being the empty list. Somewhat more precisely, $\mathbf{list}(D^{\mathbf{u}})$ is the least set L containing $[\,]^{\mathbf{u}^{(\omega)}}$ and containing the pair $[e|l]$ for any $e \in L$ and $l \in L$.

($\mathbf{d_6}$) For any $\mathbf{u}, \mathbf{v} \in \mathbf{B} \setminus \{\bot, \top\}$, if $\mathbf{u} \prec \mathbf{v}$, then $\epsilon^{\mathbf{u}^{(\mathbf{0})}} = \epsilon^{\mathbf{v}^{(\mathbf{0})}}$ and $[\,]^{\mathbf{u}^{(\omega)}} = [\,]^{\mathbf{v}^{(\omega)}}$.

($\mathbf{d_7}$) If $(\mathbf{u} \to \mathbf{v}) \in \mathbf{T} \setminus \mathbf{B}$ and domains $D^{\mathbf{u}}$, $D^{\mathbf{v}}$ are defined, then the domain $D^{(\mathbf{u} \to \mathbf{v})}$ is defined as the set $(D^{\mathbf{v}})^{(D^{\mathbf{u}})}$ of all total functions from $D^{\mathbf{u}}$ to $D^{\mathbf{v}}$.

Interpretations. An *interpretation* $\iota = (\sigma, \tau, \omega, \Xi)$ consisting of *variables assignment* σ, *constants assignment* τ, *converters assignment* ω, and *contexts assignment* Ξ, is defined as follows:

($\mathbf{i_1}$) σ is a total function from $\bigcup_{\mathbf{u} \in (\mathbf{B} \setminus \{\top\})} \mathbf{V}^{\mathbf{u}}$ to $D^{\top}$ such that $\sigma(X) \in D^{\mathbf{u}}$ for $X \in \mathbf{V}^{\mathbf{u}}$.

($\mathbf{i_2}$) τ is a total function from constants to objects of corresponding types (i.e. $\tau(c) \in D^{\mathbf{v}}$ for $c \in \mathbf{C}^{\mathbf{v}}$).

($\mathbf{i_3}$) ω is a total function from converter constants to second order operators transforming $\mathbf{T}$-type functions to $\mathbf{T}$-type functions such that for each $\gamma \in \Omega^{(\mathbf{u} \Rightarrow \mathbf{v})}$, $\omega(\gamma) \in (D^{\mathbf{v}})^{(D^{\mathbf{u}})}$.

($\mathbf{i_4}$) Ξ is a total function from terms, context operators and typed context variables into objects in $D^{\top}$: $\Xi(t, \phi, x) \in D^{\mathbf{u}}$ for any term t, context operator $\phi \in \mathbf{\Phi}$ and context variable $x \in \mathbf{R}^{\mathbf{u}}$.

For an interpretation $\iota = (\sigma, \tau, \omega, \Xi)$, an object variable $X \in \mathbf{V}^{\mathbf{u}}$ and an object $d \in D^{\mathbf{u}}$, we denote by $\iota\{X := d\}$ the interpretation $(\sigma_1, \tau, \omega, \Xi)$, in which $\sigma_1(Y) = \sigma(Y)$ for all $Y \neq X$ and
$\sigma_1(X) = d$.

Term values. For an *interpretation* $\iota = (\sigma, \tau, \omega, \Xi)$ and a term $t_0 \in \mathcal{T}^{\mathbf{u}}$, the *value* of t_0 in interpretation ι is denoted $|t_0\,\|^{\mathbf{u},\iota}$ and defined by $|t_0\,\|^{\mathbf{u},\iota} =_{df} |t_0\,\|^{t_0,\mathbf{u},\iota}$, where the value $|t\,\|^{t_0,\mathbf{u},\iota}$ of sub-terms t of term t_0 is defined recursively as follows:

($\mathbf{s_0}$) $|t\,\|^{t_0,\mathbf{u},\iota} = \sigma(X)$ for $t = X \in \mathbf{V}^{\mathbf{u}}$.

($\mathbf{s_1}$) $|t\,\|^{t_0,\mathbf{u},\iota} = \tau(\mathbf{k})$ for $t = \mathbf{k} \in C^{\mathbf{u}}$.

($\mathbf{s_2}$) Let $t = \phi_x$ for some $\phi \in \mathbf{\Phi}$ and $x \in \mathbf{R}^{\mathbf{u}}$. Then $|t\,\|^{t_0,\mathbf{u},\iota} = \Xi(t_0, \phi, x)$.

($\mathbf{s_3}$) Let $\mathbf{u} = \mathbf{v}^{(\mathbf{0})} \in \mathbf{O}$. Then $|t\,\|^{t_0,\mathbf{u},\iota} = \varepsilon^{\mathbf{v}^{(\mathbf{0})}}$ if $t = \emptyset^{\mathbf{v}^{(\mathbf{0})}}$, and $|t\,\|^{t_0,\mathbf{u},\iota} = |t\,\|^{t_0,\mathbf{v},\iota}$ if $t \neq \emptyset^{\mathbf{v}^{(\mathbf{0})}}$.

($\mathbf{s_4}$) Let $\mathbf{u} = \mathbf{v}^{(\omega)}$ and $\mathbf{v}^{(\omega)} \in \mathbf{I}$. Then

(i) $|t\ \|^{t_0,\mathbf{u},\iota} = [\ ]^{\mathbf{v}^{(\omega)}}$, if $t = \mathbf{nil}^{\mathbf{v}^{(\omega)}}$,

(ii) $|t\ \|^{t_0,\mathbf{u},\iota} = [|t_1\ \|^{t_0,\mathbf{v}_1,\iota}, ..., |t_k\ \|^{t_0,\mathbf{v}_k,\iota}]$, if $t = [t_1, ..., t_k]$ and $t_i \in \mathcal{T}^{\mathbf{v}_i}$, for some $\mathbf{v}_i \prec \mathbf{v}$ and all $1 \leq i \leq k$.

$(\mathbf{s_5})$ $|t\ \|^{t_0,\mathbf{u},\iota} = (\omega(\gamma))(|t_1\ \|^{t_0,\overline{\mathbf{u}},\iota})$, if $t = \gamma\{t_1\}$.

$(\mathbf{s_6})$ Let $t = (t_1\ t_2)$, where $t_1 \in \mathcal{T}^{(\mathbf{u}_1\ \to\ \mathbf{u})}$ and $t_2 \in \mathcal{T}^{\mathbf{v}_1}$ for some $\mathbf{v}_1 \preceq \mathbf{u}_1$. Then

$$|t\ \|^{t_0,\mathbf{u},\iota} = |t_1\ \|^{t_0,(\mathbf{u}_1\ \to\ \mathbf{v}),\iota}\ (|t_2\ \|^{t_0,\mathbf{v}_1,\iota}).$$

$(\mathbf{s_7})$ Let $\mathbf{u} = (\mathbf{u}_1\ \to\ \mathbf{v})$, $t = \lambda X.\ t_1 \in \mathcal{T}^{(\mathbf{u}_1\ \to\ \mathbf{v})}$ for some $t_1 \in \mathcal{T}^{\mathbf{v}}$ and $X \in \mathbf{V}^{\mathbf{u}_1} \cap FV(t_1)$. Then $|t\ \|^{t_0,\mathbf{u},\iota} = f$, where f is the function defined by:

$$f(d) = |t_1\ \|^{t_0,\mathbf{u},\iota\{X:=d\}} \text{ for each } d \in D^{\mathbf{u}_1}. \ \square$$

It is not difficult to prove that this definition is correct.

Proposition 2 *$|t\ \|^{\mathbf{u},\iota} \in D^{\mathbf{u}}$ for any type $\mathbf{u}$, any term $t \in \mathcal{T}^{\mathbf{u}}$ and any interpretation ι.*

Besides this, term values are invariant with respect to reducibility.

Proposition 3 *If $t_1 \twoheadrightarrow t_2$, then $|t_1\ \|^{\mathbf{u},\iota} = |t_2\ \|^{\mathbf{u},\iota}$ for any type $\mathbf{u}$ and interpretation ι.*

References

Barendregt, H. P. 1981. *The Lambda Calculus. Its Syntax and Semantics*. Amsterdam: North Holland Publishing Co.

Barwise, J. and J. Perry. 1983. *Situations and Attitudes*. Cambridge: MIT Press.

Buszkowski, W. 1997. Mathematical linguistics and proof theory. In van Benthem and ter Meulen (1997), chap. 12, pages 683–736.

Dikovsky, A. 2003. Discourse plan → meaning. In *Proceedings of the First International Conference on Meaning-Text Theory (MTT2003)*, pages 299–308. Paris, France.

Gamut, L. T. F., ed. 1991. *Intensional Logic and Logical Grammar. Vol. 2*. Chicago: University of Chicago Press.

Kamp, H. 1981. A theory of truth and semantic representation. In J. Groenendijk, T. Janssen, and M. Stokhoff, eds., *Formal Methods in the Study of Language*. Dordrecht: Foris.

Mel'čuk, I. 1997. Vers une linguistique ≪Sens–Texte≫ . Leçon inagurale au Collège de France .

Mel'čuk, I. and A. A. Holodovich. 1970. K teorii grammaticheskogo zaloga [Towards a theory of grammatical voice]. *Narody Afriki i Azii [Nations of Africa and Asia]* (4).

Moortgat, M. 1997. Categorial type logics. In van Benthem and ter Meulen (1997), chap. 2, pages 93–177.

Muskens, R., J. van Benthem, and A. Visser. 1997. Dynamics. In van Benthem and ter Meulen (1997), chap. 10, pages 587–648.

Sowa, J., ed. 1991. *Principles of Semantic Networks*. San Mateo: Morgan Kaufmann.

van Benthem, J. and A. ter Meulen, eds. 1997. *Handbook of Logic and Language*. Amsterdam: North-Holland Elsevier, MIT Press.

6

A Learnable Class of Classical Categorial Grammars from Typed Examples

Daniela Dudau-Sofronie, Isabelle Tellier, Marc Tommasi

6.1 Introduction

Categorial Grammars are well known lexicalized formalisms, often used to model natural languages. Their main interest is their expressivity and the fact that they allow good connections with formal semantics in Montague's tradition. The simplest instance of this large family is known as AB-Categorial Grammars, or Classical Categorial Grammars (CCG in the following). Although too rudimentary to model subtle linguistic phenomena, this family is interesting to study because some formal learnability results (in Gold's model) have recently been proved for large subclasses of CCGs (Kanazawa, 1998). But these results do not provide tractable learning algorithms, as most of the problems to be solved are NP-hard (Costa-Florêncio, 2001, 2002). The only favorable case is when rigid CCGs are to be learned from Structural Examples. But the class of rigid grammars has poor expressive power with respect to natural language. Moreover, learning algorithms must have access to structural examples which correspond to derivation trees of sentences and it seems to us that this requirement is not natural.

We define a new subclass of CCGs with good properties from a language-theoretic point of view. Our main result is that for every CCG,

Proceedings of FGVienna: The 8th Conference on Formal Grammar.
Gerald Penn (ed.).
Copyright © 2008, CSLI Publications.

another CCG producing the same structure language as (i.e. strongly equivalent with) the first one and belonging to this new subclass can be built. This new subclass is then proved learnable from Typed Examples. Typed Examples are sentences enriched with lexicalized information which can be interpreted as coming from semantics, and are thus more "naturally available" than Structural Examples.

The availability of the Typed Examples can be argued firstly from a theoretical point of view, exploiting the compositionality and secondly by the efforts that are done to build up a natural language resource (corpus) from texts enriched with semantic information. As a matter of fact, the connection of Categorial Grammars with semantics relies on a formal statement of the well known Principle of Compositionality that states : *"the meaning of a compound expression is a function of the meaning of its parts and of the syntactic rules by which they are combined"* (Partee, 1990). If the *"parts"* are assimilated with words and the *"compound expressions"* with phrases, this formulation implies that words have individual meanings and the semantics of a phrase (and thus of a sentence) only depends of the meaning of its words and of its syntactic structure. We believe that this Principle is still under-exploited in formal models of grammatical inference. This paper presents a new way of considering learning Categorial Grammars from semantic knowledge. We make the hypothesis that semantic types, in the usual sense, are general information making a distinction between facts, entities and properties satisfied by entities. Most knowledge representation formalisms use this notion, so types can be supposed to be directly extracted from the environment. Types can also be considered as *lexicalized structural information*.

For practical reasons we need corpora of typed texts. Such corpora are not available and they have to be built. As semantic types are lexicalized, simpler resources like lexical taggers are of great help. A tagger is able to recognize proper nouns, common nouns and other lexical items whose lexical tag is easily transformable in a lexical type (for verbs, for example some post-treatment needs to be done, as well as for conjunctions, etc.). We are working to produce a clean version of a typed corpus in French of almost 100,000 words that will be used for experiments.

This paper is organized in five sections. The second section introduces the preliminary notions: CCGs, canonical semantic types and the definition of the new introduced subclass of CCGs. The third section presents the main result of strong equivalence and the fourth section is about the learnability from typed examples. The fifth section concludes.

6.2 A New Subclass of Classical Categorial Grammars

6.2.1 Classical Categorial Grammars

Let $\mathcal{B}$ be a countably infinite set of basic categories containing a distinguished category $S \in \mathcal{B}$, called the axiom. We note $Cat(\mathcal{B})$ the term algebra built over the two binary symbols $/$, $\backslash$ and the set $\mathcal{B}$: $Cat(\mathcal{B})$ is the smallest set such that $\mathcal{B} \subset Cat(\mathcal{B})$ and for any $A \in Cat(\mathcal{B})$ and $B \in Cat(\mathcal{B})$ we have: $/(A, B) \in Cat(\mathcal{B})$ and $\backslash(A, B) \in Cat(\mathcal{B})$.

Let Σ be a fixed alphabet called vocabulary. A **categorial grammar over** Σ is any finite relation between Σ and $Cat(\mathcal{B})$, i.e. $G \subset \Sigma \times Cat(\mathcal{B})$ and G is finite. For a symbol $a \in \Sigma$ and a category $A \in Cat(\mathcal{B})$ if $\langle a, A\rangle \in G$, we say that the category A is assigned to a.

In the general framework of categorial grammars, the language $L(G)$ of a grammar G is the set of finite concatenations of elements of the vocabulary for which there exists an assignment of categories that can be *reduced* to the axiom S. For **Classical Categorial Grammars (or CCGs)**, the admitted reduction rules for any categories A and B in $Cat(\mathcal{B})$ are[1]:

- forward application $FA : /(A, B).A \to B$;
- backward application $BA : A.\backslash(A, B) \to B$

We denote by $\mathcal{G}$ the set of every Classical Categorial Grammar and for any integer $k \geq 1$, $\mathcal{G}_k$ is the set of k-valued CCGs, i.e. the set of CCGs assigning at most k different categories to each member of its vocabulary.

As usual in term algebras, a context is a category with exactly one occurrence of a distinguished constant (not in $\mathcal{B}$). We denote $C[]$ a context and $C[A]$ is the category obtained by replacing the distinguished constant by the category A. Forward and backward rules justify that for any category $X = C[/(A, B)]$ (or $X = C[\backslash(A, B)]$) we say that A occurs in X at an argument position and B occurs in X at a result position.

Any mapping Φ defined from $\mathcal{B}$ to $Cat(\mathcal{B})$ can be extended to contexts and elements of $Cat(\mathcal{B})$ in the following way: for every $A \in Cat(\mathcal{B})$ and $B \in Cat(\mathcal{B})$, $\Phi(\backslash(A, B)) = \backslash(\Phi(A), \Phi(B))$ and $\Phi(/(A, B)) = /(\Phi(A), \Phi(B))$. For any CCG G, we can also define $\Phi(G) = \{\langle a, \Phi(A)\rangle \mid \langle a, A\rangle \in G\}$.

[1]These rules justify the fractional notations of the category-building operators $/$ and $\backslash$ usually used in the literature. Categories, often written B/A (resp. $A\backslash B$), can be considered as functors expecting as argument the category A and providing as result the category B. In this paper, we do not use this notation because of some constructions. We use instead $/(A, B)$ and $\backslash(A, B)$ where A is always the argument and B the functor.

We denote by $\mathcal{B}(G)$ the (finite) set of basic categories that occur in G and $Cat(\mathcal{B}(G)) \subset Cat(\mathcal{B})$ is the set of categories that occur in G.

6.2.2 Canonical Types

For any countable set of basic categories $\mathcal{B}$, we define the set of canonical types $Types(\mathcal{B})$ as the smallest set such that $\mathcal{B} \subset Types(\mathcal{B})$ and for any $U \in Types(\mathcal{B})$ and $V \in Types(\mathcal{B})$ we have $(U, V) \in Types(\mathcal{B})$. The type (U, V) is to be read as a functor taking as argument the type U and providing as result the type V. Canonical Types can thus be seen as non oriented categories, *i.e.* categories where both operators / and \ are erased. We call the Canonical Typing Function h the unique function from $Cat(\mathcal{B})$ to $Types(\mathcal{B})$ recursively defined by: (1) $h_{|\mathcal{B}} = Id_{\mathcal{B}}$ where $Id_{\mathcal{B}}$ denotes the identity function on the set $\mathcal{B}$; (2) for any $A \in Cat(\mathcal{B})$ and $B \in Cat(\mathcal{B})$ we have: $h(/(A, B)) = h(\backslash(A, B)) = (h(A), h(B))$. The Canonical Typing Function simply transforms categories into the corresponding Canonical Types by deleting the operators. h can also be naturally extended to contexts of categories.

6.2.3 The Class $\mathcal{G}_{Type}$

Definition 13 For any vocabulary Σ and any set of basic categories $\mathcal{B}$, we note $\mathcal{G}_{Type}$ the set of CCGs G on Σ satisfying the property:

$$\forall \langle a, A\rangle \in G,\ \langle a, A'\rangle \in G \quad h(A) = h(A') \Rightarrow A = A'. \tag{6.7}$$

In other words, in the CCGs of $\mathcal{G}_{Type}$, the types corresponding with the categories of a single member of the vocabulary are all distinct. Different categories assigned to the same symbol of the vocabulary can be distinguished looking at their type. We can compare the class $\mathcal{G}_{Type}$ with the classes $\mathcal{G}_k$ of k-valued CCGs. Interestingly, the two notions do not coincide.

Example 1.

Let $\Sigma = \{a, b\}$ and let us consider a CCG grammar G_0 recognizing the language a^*ba^*. G_0 is defined by the assignments $\{\langle b, S\rangle, \langle a, /(S, S)\rangle, \langle a, \backslash(S, S)\rangle\}$. It is worth noting that G_0 belongs to the class $\mathcal{G}_2$ of 2-valued CCGs and therefore belongs to any $\mathcal{G}_k$, $k \geq 2$ but is not in $\mathcal{G}_{Type}$.

Let $(X_i)_{i\geq 0}$ be a sequence of categories defined by $X_0 = A$, $X_1 = /(A, S)$ and $X_i = /(A, X_{i-1})$ for $i > 1$. Let us consider the grammar G_k defined by: $G_k = \{\langle a, X_i\rangle | 0 \leq i \leq k\}$. For $k \geq 1$, the language recognized by G_k is $L(G_k) = \{a^i \mid 2 \leq i \leq k + 1\}$. For every $k \geq 1$, G_k is in $\mathcal{G}_{Type}$ and G_k is $(k + 1)$-valued but not k-valued.

Proposition 1 *$\mathcal{G}_1 \subset \mathcal{G}_{Type}$ and $\forall k > 1$, $\mathcal{G}_{Type} \cap \mathcal{G}_k \neq \emptyset$, $\mathcal{G}_k \not\subseteq \mathcal{G}_{Type}$ and $\mathcal{G}_{Type} \not\subseteq \mathcal{G}_k$.*

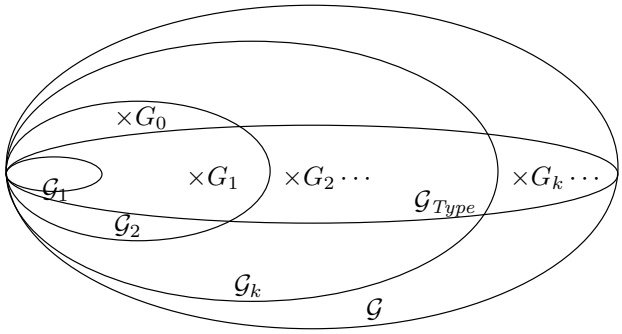


FIGURE 1 Comparisons between classes of k-valued categorial grammars and the $\mathcal{G}_{Type}$ class.

Proof.

The fact that $\mathcal{G}_1 \subset \mathcal{G}_{Type}$ is obvious and other properties are easily proved using the grammars given in Example 1. □

Therefore, the situation is the one displayed in Figure 1.

The set of languages generated (or recognized) by CCGs is the set of ϵ-free context-free languages, where ϵ is the empty word (Gaifman's theorem in Bar-Hillel et al. (1960)). Now that we have defined a subclass of $\mathcal{G}$, it is natural to wonder if its members allow the generation of the same set of languages or only of a subset of it.

Proposition 2 *For every fixed sets Σ and $\mathcal{B}$ and every ϵ-free context-free language $L \subset \Sigma^*$ there exists a CCG, $G \in \mathcal{G}_{Type}$ so that $L(G) = L$.*

Proof. It has long been recognized that Gaifman's proof is equivalent with the existence of the strong Greibach normal form for ϵ-free context-free languages. That is, for every ϵ-free context-free language $L \subset \Sigma^*$ there exists a phrase structure grammar $G' = \langle \Sigma, NT, P, S \rangle$ (where Σ is the set of terminal symbols, NT the set of nonterminal symbols, P the set of production rules and $S \in NT$ the axiom) in strong Greibach normal form satisfying $L(G') = L$. This normal form specifies that every production rule in P is of the form $X \longrightarrow a$ or $X \longrightarrow aX_1$ or $X \longrightarrow aX_1X_2$ with X, X_1 and X_2 elements of NT and a element of Σ. To build a CCG G that is strongly equivalent with G', proceed as follows: (1) define a bijection f between NT and a part of $\mathcal{B}$ satisfying $f(S) = S$; (2) for every rule of the form $X \longrightarrow a$ in P, let $\langle a, f(X) \rangle \in G$; (3) for every rule of the form $X \longrightarrow aX_1$ in P, let $\langle a, /(f(X_1), f(X)) \rangle \in G$; (4) for every rule of the form $X \longrightarrow aX_1X_2$ in P, let $\langle a, /(f(X_1), /(f(X_2), f(X))) \rangle \in G$. We have $L(G) = L(G') = L$. Because f is a bijection and, by construction, types are pairwise distinct

in G, it is trivial to check that $G \in \mathcal{G}_{Type}$. □

Though constructive, this proof is not satisfying, as the CCG obtained produces syntactic analysis trees of a very peculiar form, where only the rule *FA* is used. The next section will take into account the structures of the analysis trees produced by CCGs and provide a much more interesting (because structure-preserving) result of the same kind.

6.3 Structure Languages of $\mathcal{G}_{Type}$

A functor-argument structure over an alphabet Σ is a binary-branching tree whose leaf nodes are labeled by elements of Σ and whose internal nodes are labeled either by *BA* or *FA*. The set of functor-argument structures over Σ is denoted Σ^F. Let G be a CCG. A *Structural Example* for G is an element of Σ^F obtained from the parse tree of a sentence w in $L(G)$ by deleting categories. The *Structure Language* of G, denoted by $FL(G)$, is the set of Structural Examples of G.

The notion of Structural Example is of crucial importance as it is the basis of every learning result about CCGs (Buszkowski and Penn, 1990, Kanazawa, 1998). Furthermore, it allows the connection between syntax and semantics (Tellier, 1999). In the domain of CCGs, two grammars can be said strongly equivalent if they share the same Structure Language.

Example 2.

Let us consider the grammar G_0 of Example 1. The sentence *aba* has two parses in this grammar and therefore there are two Structural Examples for G and *aba*: $FA(a, BA(b, a))$ and $BA(FA(a, b), a)$.

6.3.1 The Main Result

Theorem 3 *For any vocabulary Σ and any set of basic categories $\mathcal{B}$, for every CCG $G \in \mathcal{G}$, there exists a CCG $G' \in \mathcal{G}_{Type}$ so that $FL(G) = FL(G')$.*

The theorem states that the class $\mathcal{G}_{Type}$ has the same expressive power in a strong sense as the entire class $\mathcal{G}$ of Classical Categorial Grammar. That is to say that the restriction imposed by Eq. (6.7) has no incidence on the expressive power of the grammars in $\mathcal{G}_{Type}$.

The proof of this theorem is constructive, having as support the two algorithms respectively named: algorithm 1 G to Gtype, page 88 and algorithm 2 Transform, page 89. We construct a finite chain of grammars $G_0, G_1, \ldots G_m \in \mathcal{G}$ such that $G_0 = G$, $G_m = G'$ and $\forall k < m$, $FL(G_k) = FL(G_{k+1})$ every time performing some transformations over some pairs of categories that contradict Eq. (6.7). To transform assignments that do not fulfill Eq. (6.7), we need new basic categories and

we must also introduce new assignments. Therefore, the counterpart of this result is that the grammar G' can be much larger w.r.t. the number of assignments than the grammar G. We illustrate the construction by an example.

Example 3.

$$\text{Let } G_0 = \left\{ \begin{array}{lll} \langle a, /(B,S)\rangle, & \langle a, \backslash(B,S)\rangle, & (1) \\ \langle b, /(A,B)\rangle, & \langle b, \backslash(A,B)\rangle, & (2) \\ \langle e, /(/(B,S),S)\rangle, & & \\ \langle d, /(B,S)\rangle, & \langle c, A\rangle & \end{array} \right\}$$

For this grammar G_0 we have 2 pairs of assignments (denoted by (1) and (2)) that contradict Eq. (6.7). We begin by applying a transformation on the first pair: we introduce two basic categories $B_f, B_b \in \mathcal{B}$ in order to replace occurrences of B in the argument position of the two categories. The two basic categories B_f, B_b are new categories for G_0, that is to say that they do not occur in any assignment of G_0. In order to memorize which categories have been introduced and their corresponding category in the initial grammar we define a mapping Φ that maps B_f and B_b to B, $\Phi(B_f) = B$, $\Phi(B_b) = B$.

But this replacement has two consequences. Firstly, if a sequence $w \in \Sigma^*$ is reduced by G_0 into B then wa aw are in $L(G_0)$. Therefore, to preserve the set of correct sentences, for every category in the grammar where B occurs at a result position, we add new assignments where these occurrences are replaced respectively by B_f and B_b. Secondly, if a sequence $w \in \Sigma^*$ is reduced by G_0 into $/(B,S)$, then ew is also in $L(G_0)$. Again, to preserve the set of correct sentences, for every assignment $\langle l, C\rangle$ in G_0, $l \in \Sigma$, $C \in Cat(\mathcal{B}(G_0))$ where $/(B,S)$ occurs in C, we need to add a new assignment $\langle l, C'\rangle$ where $/(B,S)$ has been replaced by $/(B_f,S)$ (the same stands for $\backslash(B,S)$). This leads to the grammar:

$$G_1 = \left\{ \begin{array}{lll} \langle a, /(B_f,S)\rangle, & \langle a, \backslash(B_b,S)\rangle, & \\ \langle b, /(A,B)\rangle, & \langle b, \backslash(A,B)\rangle, & (2a) \\ \langle b, /(A,B_f)\rangle, & \langle b, \backslash(A,B_f)\rangle, & (2b) \\ \langle b, /(A,B_b)\rangle, & \langle b, \backslash(A,B_b)\rangle, & (2c) \\ \langle e, /(/(B,S),S)\rangle, & \langle e, /(/(B_f,S),S)\rangle & \\ \langle d, /(B,S)\rangle, & \langle d, /(B_f,S)\rangle, \quad \langle c, A\rangle & \end{array} \right\}$$

Now, the replacement process repeats with the grammar G_1. In G_1 there are no more assignments for the symbol a that contradicts Eq. (6.7), but the transformation has introduced four assignments for the symbol b that also contradicts Eq. (6.7). Fortunately, pairs of assignments (2a), (2b) and (2c) can be processed simultaneously, *i.e.* only two new basic categories A_f and A_b are necessary. This is true because

the three pairs in G_1 have been obtained from the unique pair (2) of G_0 and this can be checked using Φ. This trick is essential in our proof to obtain the termination of the replacement process.

$$G_2 = \left\{ \begin{array}{l} \langle a, /(B_f, S)\rangle, \quad \langle a, \backslash(B_b, S)\rangle, \\ \langle b, /(A_f, B)\rangle, \quad \langle b, \backslash(A_b, B)\rangle, \\ \langle b, /(A_f, B_f)\rangle, \quad \langle b, \backslash(A_b, B_f)\rangle, \\ \langle b, /(A_f, B_b)\rangle, \quad \langle b, \backslash(A_b, B_b)\rangle, \\ \langle e, /(/(B, S), S)\rangle, \quad \langle e, /(/(B_f, S), S)\rangle \\ \langle d, /(B, S)\rangle, \quad \langle d, /(B_f, S)\rangle, \quad \langle c, A\rangle, \quad \langle c, A_f\rangle, \quad \langle c, A_b\rangle \end{array} \right\}$$

and we obtain that G_2 is in $\mathcal{G}_{Type}$.

We now enlighten the properties satisfied by pairs of assignments that must be processed simultaneously. In the algorithm we will apply a transformation step for sets that are *type-indistinguishable* w.r.t Φ, G and maximal in size.

Definition 14 Let G and G' be two CCGs and let Φ be a mapping from $\mathcal{B}(G')$ into $Cat(\mathcal{B}(G))$ such that $\Phi(G') = G$. Let Γ be a set of pairs of assignments $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\}$ such that $\alpha_i, \beta_i \in G'$ for every $1 \leq i \leq n$. We say that Γ is *type-indistinguishable* w.r.t Φ, G if there exists:

$a \in \Sigma$, two contexts C, C' and $A, B, A', B' \in Cat(\mathcal{B}(G))$, and

$\forall i \in \{1, \ldots, n\}$, $A_i, B_i, A'_i, B'_i \in Cat(\mathcal{B}(G'))$, and $2 \times n$ contexts C_i, C'_i

such that

$h(C) = h(C'), h(A) = h(A'), h(B) = h(B')$, and

$\langle a, C[/(A, B)]\rangle \in G$ and $\langle a, C[\backslash(A', B')]\rangle \in G$

$\forall i \in \{1, \ldots, n\}$, $\alpha_i = \langle a, C_i[/(A_i, B_i)]\rangle$,

$\beta_i = \langle a, C'_i[\backslash(A'_i, B'_i)]\rangle$

$$\begin{array}{lll} h(C_i) = h(C'_i), & & \Phi(C_i) = C,\ \Phi(C'_i) = C', \\ h(A_i) = h(A'_i), & \text{and} & \Phi(A_i) = A,\ \Phi(A'_i) = A', \\ h(B_i) = h(B'_i) & & \Phi(B_i) = B,\ \Phi(B'_i) = B'. \end{array}$$

Algorithm 1 G to GType.

Require: a CCG G_i

1: $\Phi = Id_{Cat(\mathcal{B})}$ is the identity on $Cat(\mathcal{B})$.
2: $G' = G_i$
3: **while** There exists Γ in G', a maximal type-indistinguishable set w.r.t Φ, G_i **do**
4: (G', Φ) = Transform(G', G_i, Φ, Γ)
5: **end while**

Ensure: G'

Algorithm 2 Transform.

Require: Two grammars G and G_i, a mapping Φ and Γ a *type-indistinguishable* set w.r.t Φ, G_i.
{With Γ we assume as indicated in Def. 14 $a \in \Sigma$, two contexts C, C' and $A, B, A', B' \in Cat(\mathcal{B}(G_i))$, and $\forall i \in \{1, \ldots, n\}$, $A_i, B_i, A'_i, B'_i \in Cat(\mathcal{B}(G))$, and $2 \times n$ contexts C_i, C'_i.}

$G' = G - \bigcup_i \{\langle a, C_i[/(A_i, B_i)]\rangle, \langle a, C'_i[\backslash(A'_i, B'_i)]\rangle\}$;
Let A_f and A_b be two basic categories not in $\mathcal{B}(G)$;
$G' = G' \cup \bigcup_i \{\langle a, C_i[/(A_f, B_i)]\rangle, \langle a, C'_i[\backslash(A_b, B'_i)]\rangle\}$;
$REM = G'$; {REM is the set of assignments that remain to be processed}
while $REM \neq \emptyset$ **do**
  {Result position}
  Pick and remove $\langle b, D[W]\rangle$ from REM;
  if $W = f(W', A_i)$ for some $i \in \{1, \ldots, n\}$ and $f \in \{\backslash, /\}$ **then**
    Add $\langle b, D[f(W', A_f)]\rangle$ to G' and to REM;
  end if
  if $W = f(W', A'_i)$ for some $i \in \{1, \ldots, n\}$ and $f \in \{\backslash, /\}$ **then**
    Add $\langle b, D[f(W', A_b)]\rangle$ to G' and to REM;
  end if
  if $D[W] = A_i$ **then**
    Add $\langle b, A_f\rangle$ to G' and to REM;
  end if
  if $D[W] = A'_i$ **then**
    Add $\langle b, A_b\rangle$ to G' and to REM;
  end if
  {Any position }
  if $W = /(A_i, B_i)$ **then**
    Add $\langle b, D[/(A_f, B_i)]\rangle$ to G' and to REM;
  end if
  if $W = \backslash(A'_i, B'_i)$ **then**
    Add $\langle b, D[\backslash(A_b, B'_i)]\rangle$ to G' and to REM;
  end if
end while
$\Phi'(A_f) = A$, $\Phi'(A_b) = A'$ and $\Phi'(X) = \Phi(X)$ for any $X \notin \{A_f, A_b\}$.

Ensure: G' and Φ'.

Proof.(of Theorem 3)
We need to prove three things: given as input an initial grammar $G_i \in \mathcal{G}$ the algorithm terminates, it ultimately outputs a final grammar $G_f \in \mathcal{G}_{Type}$ and $FL(G_i) = FL(G_f)$.

The algorithm iteratively builds a sequence of grammars $G_0, \ldots, G_m$, where $G_0 = G_i$ and $G_m = G_f$. Let n_k be the number of disjoint sets Γ maximal in size in G_k that are *type-indistinguishable* w.r.t Φ, G_i.

n_0 can be easily calculated w.r.t. the initial grammar. Termination proof relies on the fact that n_k decreases at each step of the iteration. The Transform algorithm only introduces new assignments such that, if they contradict Eq. (6.7), then they increase the size of some type-indistinguishable sets w.r.t Φ, G_i. That is to say that no such new sets are introduced by the transformation algorithm. Moreover, when Transform applies, it suppresses such a type-indistinguishable set from the input grammar.

To prove that G_f is in $\mathcal{G}_{Type}$, we use the fact that if $n_k = 0$ for some grammar G_k in the sequence above, then G_k is in $\mathcal{G}_{Type}$.

For the last point, we first prove that $FL(G_i) \supseteq FL(G_f)$ using the fact that $\Phi(G_f) = G_i$. Indeed, the equality is invariant with the Transform algorithm and trivially true at the beginning. Then, using properties of substitutions in CCGs, to which our mappings can be assimilated (see Buszkowski and Penn (1990)) we obtain the inclusion. The proof for the reverse inclusion is more technical and relies on case analysis. We develop the proof ideas in one case in this sketch of proof, other cases being similar. Since some assignments have been removed we must check that the same derivations are still possible using new assignments. Consider that G' is obtained from G by the Transform algorithm. Following notations in the Transform algorithm, $\langle a, C_i[/(A_i, B_i)]\rangle$ has been removed and replaced by $\langle a, C_i[/(A_f, B_i)]\rangle$. Consider a parse tree τ in G where A_i is not useless. Then there are two sibling nodes defining subtrees τ_1 and τ_2 labelled by A_i and $/(A_i, B_i)$ in τ. Because we have added assignments that put the basic category A_f in every category where A_i occurs in a result position, we can do the same derivation tree as τ_1 and label it by A_f. Every time $/(A_i, B_i)$ occurs in a category, a new assignment with $/(A_f, B_i)$ is introduced, therefore we can do the same derivation tree as τ_2 and label it by $/(A_f, B_i)$. Using these properties, we are able to prove that a sentence w is associated with a category C in G' if and only if it is associated with a category $\Phi(C)$ in the initial grammar G_i and both derivation trees are identical up to Φ. Hence structural languages $FL(G_i)$ and $FL(G_f)$ are identical. □

6.4 Learnability of $\mathcal{G}_{Type}$ from Typed Examples

The previous section proved that the class $\mathcal{G}_{Type}$ has good properties relatively to the entire class $\mathcal{G}$ as it allows to produce every possible Structure Language generated by a CCG. But the initial motivation for introducing this class was that of learnability. We now justify the interest of $\mathcal{G}_{Type}$ in terms of formal learning theory and we argue for its plausibility to model natural language learning.

6.4.1 Grammar Systems and Learnability

For any CCG G, a Canonical Typed Example for G is a sequence of couples $\langle word, type \rangle$ where the first items of the couples compose a sentence of G and the second items are the types corresponding with the categories assigned to each word and allowing the syntactic analysis in G. We define $TL : \mathcal{G} \rightarrow pow((\Sigma \times Types(\mathcal{B}))^*)$ the function that maps every CCG G with the set $TL(G)$ of the Canonical Typed Examples it generates, also called the Canonical Typed Language of G:

$TL(G) = \{\langle u_1, \tau_1 \rangle ... \langle u_n, \tau_n \rangle | \forall i \in \{1, ..., n\}, \exists c_i$ so that $\langle u_i, c_i \rangle \in G$, $\tau_i = h(c_i)$ and $c_1 ... c_n \rightarrow^* S\}$.

To deal with questions of learnability, Kanazawa introduces in (Kanazawa, 1998) the notion of grammar system. This allows a reformulation of the classical Gold's model of *identification in the limit from positive examples* (Gold, 1967). We recall this notion here.

A grammar system is a triple $\langle \Omega, \Lambda, L \rangle$ where Ω is the hypothesis space (in our context, Ω will be a set of formal grammars), the sample space Λ is a recursive subset of A^*, for some fixed alphabet A (elements of Λ are sentences and subsets of Λ are languages) and L is a naming function that maps elements of Ω into languages i.e. $L : \Omega \rightarrow pow(\Lambda)$. The universal membership problem, i.e. the question of whether $s \in L(G)$ holds between $s \in \Lambda$ and $G \in \Omega$, is supposed computable.

Let $\langle \Omega, \Lambda, L \rangle$ be a grammar system and $\phi : \bigcup_{k \geq 1} \Lambda^k \rightarrow \Omega$ be a computable function. We say that ϕ converges to $G \in \Omega$ on a sequence $\langle s_i \rangle_{i \in \mathbb{N}}$ of elements of Λ if $G_i = \phi(\langle s_0, ..., s_i \rangle)$ is defined and equal to G for all but finitely many $i \in \mathbb{N}$ - or equivalently if there exists $n_0 \in \mathbb{N}$ such that for all $i \geq n_0$, G_i is defined and equal to G. Such a function ϕ is said to learn $\mathcal{G} \subseteq \Omega$ and $\mathcal{G}$ is said learnable if for every language L in $L(\mathcal{G}) = \{L(G) | G \in \mathcal{G}\}$ and for every infinite sequence $\langle s_i \rangle_{i \in \mathbb{N}}$ that enumerates the elements of L (i.e. so that $\{s_i | i \in \mathbb{N}\} = L$), there exists some G in $\mathcal{G}$ such that $L(G) = L$ and ϕ converges to G on $\langle s_i \rangle_{i \in \mathbb{N}}$.

Kanazawa has proved that $\forall k \geq 1, \mathcal{G}_k$ is learnable both in the grammar system $\langle \mathcal{G}, \Sigma^F, FL \rangle$ (i.e from Structural Examples) and in the grammar system $\langle \mathcal{G}, \Sigma^*, L \rangle$ (i.e. from string examples).

In the formal learning model of Gold, learnability results for CCGs become trivial when typed examples are given in input. Indeed, there are a bounded number of grammars compatibles with any finite presentation as soon as all elements of the lexicon have been presented. Membership is decidable and therefore any simple enumerative algorithm of compatible grammars can be easily transformed into a learner. But this is not satisfactory and more interesting remarks can be done for $\mathcal{G}_{Type}$ grammars. Indeed, we notice that to learn $\mathcal{G}_{Type}$ in the grammar sys-

tem $\langle \mathcal{G}, (\Sigma \times \mathit{Types}(\mathcal{B}))^*, TL \rangle$, it is enough to be able to learn $\mathcal{G}_1$ in the grammar system $\langle \mathcal{G}, (\Sigma \times \mathit{Types}(\mathcal{B}))^*, L \rangle$. As a matter of fact, grammars G in $\mathcal{G}_{Type}$ are such that for all pairs $\langle u, \tau \rangle \in \Sigma \times \mathit{Types}(\mathcal{B})$ belonging to a member of $TL(G)$, there exists only one c so that $\langle u, c \rangle \in G$ and $h(c) = \tau$ and are thus one to one distinct. On the vocabulary $\Sigma \times \mathit{Types}(\mathcal{B})$, these grammars are thus rigid.

This suggests a learning algorithm which would be an adaptation of the one that learns $\mathcal{G}_1$ from strings. Unfortunately, this strategy would not be efficient, since learning $\mathcal{G}_1$ from strings is NP-hard (Costa-Florêncio, 2002).

Another candidate is the learning strategy proposed in Dudau-Sofronie, Tellier, and Tommasi (2001), taking advantage of the functional nature of types and of their closeness with categories. The only remaining problem is that we still do not know if the inclusion of Typed Languages is decidable for CCGs. The answer to that open problem can have a great influence on the computational complexity of the strategy.

6.4.2 Semantic Interpretation of Types

Learning subclasses of CCGs from text (i.e. strings of words) is intractable. Richer input data need to be provided to help the learning process. The strategy investigated so far consisted in providing indications about the analysis structure underlying a given string, under the form of a Structural Example. On the contrary, we focus here on providing additional *lexical information*. The argument of learnability used in the last proof also applies for any kind of lexical item associated with a word (i.e. a member of the vocabulary) and given as input provided that two categories assigned to a same word in a grammar always give rise to two different items. This property defines the class of CCGs learnable from non-ambiguous lexical information (in the sense of ambiguity used here). A special case of lexical information, especially for natural languages, is lexical semantics. Furthermore, the types used here are inspired by those used in Montague's typed lambda calculus to represent natural language semantics. This strategy is then both more "natural to justify" and hopefully more efficient than the one making use of Structural Examples.

6.5 Conclusion

The main contribution of this paper is the definition of a precise subclass of $\mathcal{G}$ which is both learnable from typed examples and has good properties from a language-theoretic point of view, as it has the same expressive power than $\mathcal{G}$ in a strong sense. So, $\mathcal{G}_{Type}$ is *representative*

of $\mathcal{G}$ and grammars in this class can be considered as CCGs of a special normal form. The learnability of $\mathcal{G}_{Type}$ is guaranteed because unambiguous types are provided, which is a strong condition. But this result can be compared for example, with Sakakibara's 1992 result which states that every context-free language can be generated by a reversible context-free grammar and that the set of reversible context-free grammars is learnable from skeletons, i.e. from syntactic analysis trees where non-terminal symbols are deleted. This result is interesting but limited because transforming a plain context-free grammar into a reversible context-free grammar recognizing the same string language does not preserve the corresponding set of skeletons. On the contrary, our transformation is structure-preserving, which is a crucial condition in a natural language context.

Acknowledgements

This research was partially supported by: "CPER 2000-2006, Contrat de Plan état - région Nord/Pas-de-Calais: axe TACT, projet TIC"; fonds européens FEDER "TIC - Fouille Intelligente de données - Traitement Intelligent des Connaissance" OBJ 2-phasing out - 2001/3 - 4.1 - n 3; ARC INRIA Gracq et Maison des Sciences de l'Homme-Institut International Erasme.

References

Bar-Hillel, Y., C. Gaifman, and E. Shamir. 1960. On categorial and phrase structure grammars. *Bulletin of the Research Council of Israel* 9F.

Buszkowski, W. and G. Penn. 1990. Categorial grammars determined from linguistic data by unification. *Studia Logica* 49:431–454.

Costa-Florêncio, C. 2001. Consistent identification in the limit of any of the classes k-valued is NP-hard. In *Logical Aspects of Computational Linguistics, LNAI 2099*, pages 125–134. Springer.

Costa-Florêncio, C. 2002. Consistent identification in the limit of rigid grammars from strings is NP-hard. In P. Adriaans, H. Fernau, and M. V. Zaanen, eds., *Grammatical Inference: Algorithms and Applications, LNAI 2484*, pages 49–62. Springer.

Dudau-Sofronie, D., I. Tellier, and M. Tommasi. 2001. Learning categorial grammars from semantic types. In *Proceedings of the 13th Amsterdam Colloquium*, pages 79–84.

Gold, E. M. 1967. Language identification in the limit. *Information and Control* 10:447–474.

Kanazawa, M. 1998. *Learnable Classes of Categorial Grammars*. Palo Alto: CSLI Publications.

Partee, B. 1990. *Mathematical methods in Linguistics*. Linguistics and Philosophy 30. Kluwer.

Sakakibara, Y. 1992. Efficient learning of context-free grammars from positive structural examples. *Information and Computation* 97(1):23–60.

Tellier, I. 1999. Towards a semantic-based theory of language learning. In *Proceedings of the 12th Amsterdam Colloquium*, pages 217–222.

7

A Note on Countercyclicity and Minimalist Grammars

Hans-Martin Gärtner and Jens Michaelis

7.1 Introduction

Minimalist grammars (MGs), as introduced in Stabler (1997), have proven a useful instrument in the formal analysis of syntactic theories developed within the minimalist branch of the principles–and–parameters framework (cf. Chomsky 1995, 2000). In fact, as shown in Michaelis (2001), MGs belong to the class of mildly context–sensitive grammars. Interestingly, without there being a rise in (at least weak) generative power, (extensions and variants of) MGs accommodate a wide variety of (arguably) "odd" items from the syntactician's toolbox, such as *head movement* (Stabler 1997, 2001), *affix hopping* (Stabler 2001), *(strict) remnant movement* (Stabler 1997, 1999), *adjunction* (Frey and Gärtner 2002), and (to some extent) *scrambling* (Frey and Gärtner 2002).[1]

Here, we would like to explore the possibility of enriching MGs with another controversial mechanism, namely, *countercyclic operations*. These operations allow structure building at any node in the tree instead of just at the root.[2] We will first discuss countercyclic ad-

[1] A strictly formal proof showing that at least the weak generative capacity is unaffected seems to be straightforward but is still outstanding for the corresponding formalizations of adjunction and scrambling. An empirically fully satisfactory MG–treatment of scrambling, however, requires further research.

[2] Note that affix hopping and head movement as formalized in the mentioned works can be considered to be countercyclic in the weaker sense that these operations

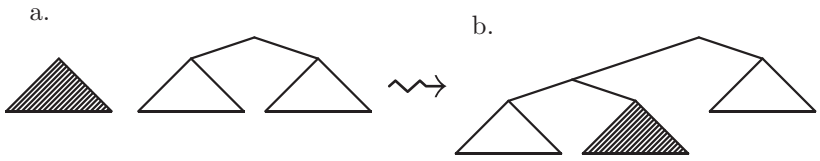
Proceedings of FGVienna: The 8th Conference on Formal Grammar.
Gerald Penn (ed.).
Copyright © 2008, CSLI Publications.

junction, which has repeatedly been postulated in the syntactic literature, especially in analyses of binding phenomena (Section 7.2.1). Then we sketch an extension of MGs that captures countercyclic adjunction (Section 7.2.2). As further discussed in Section 7.2.3, it turns out that, while weak and (even) strong generative capacity seem to remain essentially unaffected by this modification, there is an effect on what can be called *derivational generative power*, a category earlier introduced by Becker et al. (1992), which is considered to be "orthogonal" to the dimension of strong generative power. This is due to the fact that the latter is about derived structures while the former concerns derivation structures. In Section 7.3 we give an outlook on further variants of countercyclicity.

7.2 Countercyclic Adjunction

Let us briefly illustrate the crucial property of countercyclic operations, i.e. the capability of expanding the tree at a non–root position. Thus, a transition from (1a) to (1b) is countercyclic.

(1)



7.2.1 Adjuncts and Binding

Countercyclic adjunction has been argued for among others by Lebeaux (1991) on the basis of contrasts like the following.

(2) a. *She$_i$ denied the claim that Mary$_i$ fell asleep
b. *She$_i$ liked the book that Mary$_i$ read
c. *Which claim that Mary$_i$ fell asleep did she$_i$ deny
d. Which book that Mary$_i$ read did she$_i$ like

Lebeaux' account rests on the assumption that (2a) and (2b) are ruled out by Principle C of the Binding Theory (Chomsky 1981), according to which an R–expression like *Mary* must not be c–commanded by any coindexed constituent, such as *she* in our examples. The contrast in (2c)/(2d) would then follow, if there is a stage in the derivation of (2c) where such an illicit c–command relation holds, while there is no such stage in the derivation of (2d). Concretely put, (3a), i.e. the

simply do not lead to any "proper" tree expansion at any node.

stage before wh–movement applies to yield (3b), displays a Principle C violation.

(3) a. $[_{C'}$ did $[_{IP}$ she$_i$ $[_{VP}$ deny $[_{DP}$ which claim $[_{CP}$ that Mary$_i$ fell asleep]]]]]
b. $[_{CP}[_{DP}$ which claim $[_{CP}$ that Mary$_i$ fell asleep]] $[_{C'}$ did $[_{IP}$ she$_i$ $[_{VP}$ deny t]]]]

Crucially, things would be different for (2d), if instead of cyclic adjunction leading to stage (4a) and ultimately (4c) via the illicit stage (4b), adjuncts like relative clauses were allowed to be introduced "late," i.e. countercyclically. This alternative is illustrated in (5a)–(5c).

(4) a. $[_{DP}[_{DP}$ which book] $[_{CP}$ that Mary$_i$ read]]
⋮
b. $[_{C'}$ did $[_{IP}$ she$_i$ $[_{VP}$ like $[_{DP}$ $[_{DP}$ which book] $[_{CP}$ that Mary$_i$ read]]]]]
c. $[_{CP}$ $[_{DP}$ $[_{DP}$ which book] $[_{CP}$ that Mary$_i$ read]] $[_{C'}$ did $[_{IP}$ she$_i$ $[_{VP}$ like t]]]]

(5) a. $[_{C'}$ did $[_{IP}$ she$_i$ $[_{VP}$ like $[_{DP}$ which book]]]]
b. $[_{CP}$ $[_{DP}$ which book] $[_{C'}$ did $[_{IP}$ she$_i$ $[_{VP}$ like t]]]]
c. $[_{CP}$ $[_{DP}$ $[_{DP}$ which book] $[_{CP}$ that Mary$_i$ read]] $[_{C'}$ did $[_{IP}$ she$_i$ $[_{VP}$ like t]]]]

In derivation (5) there is no stage at which *Mary* is c–commanded by *she* and thus Principle C is complied with as desired.[3]

7.2.2 An MG–Treatment of Countercyclic Adjunction

An MG–account of countercyclic adjunction, subsuming the transition from (5b) to (5c) above, is straightforward. We have to adopt a variant of the *adjoin*–operation introduced into the MG–formalism by Frey and Gärtner (2002). This requires that in addition to *(basic) categorial features*, *m(erge)–selectors*, *(move–)licensees* and *(move–)licensors*, denoted in the form `x`, `=x`, `-x` and `+x`, respectively, an MG is equipped with *a(djoin)-selectors*, denoted in the form `≈x`.

As for further notation and use of symbols: appearing within a given string of features, `#` serves to mark the substring of features to its right as "unchecked." `>` denotes "right constituent projects over left one," and `<` denotes "left constituent projects over right one."[4] Furthermore,

[3] We have to sidestep many ramifications of this account such as criticisms and alternatives in terms of reconstruction and copies, not to speak of alternative approaches to binding.

[4] Recall that a minimalist tree is always binary (branching), cf. Definition 15 from the appendix.

for minimalist trees τ, τ_1 and τ_2 such that τ_1 is a subtree of τ, $\tau\{\tau_1/\tau_2\}$ represents the result of replacing τ_1 by τ_2 in τ, and

(6) τ_1 is a *maximal projection* in τ in case it is identical to τ, or it is projected over by its sister constituent (i.e. in case each subtree of τ which is a proper supertree of τ_1 has a head other than the head of τ_1).

Assuming that υ and τ are minimalist trees such that υ displays $\approx$x, and there is at least one maximal projection χ in τ fulfilling condition (7), *(right) adjunction* for the pair $\langle\upsilon,\tau\rangle$, i.e. (right) adjunction of υ to τ, can be defined as in (8).[5,6]

(7) The head–label of χ is of the form β#xβ' or βxβ'#β'' for some $\beta,\beta',\beta''\in \mathit{Feat}^*$.

(8) $\mathit{adjoin}(\upsilon,\tau)$
$= \{\tau\{\chi/[_{<}\,\chi,\upsilon']\}\mid \chi$ maximal projection in τ obeying (7) $\}$,
where υ' results from υ by interchanging the instances of # and $\approx$x, the latter immediately following the former within the head–label of υ.[7]

In order to sketch our treatment of countercyclic adjunction in (5) we choose the following small MG–lexicon.[8]

(9) a. #.n.*book* b. #.d.*she* c. #.=d.v.*like*
d. #.=n.d.-wh.*which* e. #.=v.=d.i.$\emptyset$ f. #.=i.$\approx$d.*that*
g. #.=i.+wh.c.*did* h. #.i.*Mary_read*

(10a)–(10c) below correspond to (5a)–(5c), respectively.

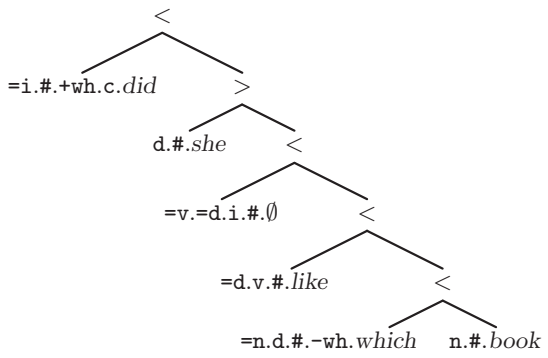
[5]Left adjunction can be defined analogously. Note that, in contrast with its counterpart in Frey and Gärtner (2002), *adjoin* as defined here does not necessarily map a pair of minimalist trees from its domain to a unique minimalist tree, there being potentially multiple "adjunction sites." The operation adopts the attractive type–preserving "x/x–approach" from *categorial grammar*. It should not be confused with the more general *(tree) adjoining* operation familiar from *tree adjoining grammar (TAG)* (cf. e.g. Joshi and Schabes 1997). The latter operation allows countercyclicity quite generally.

[6]A minimalist tree τ *displays* feature f if an instance of f starts the substring of unchecked features within τ's head–label. For any two minimalist trees υ and χ, $[_{<}\chi,\upsilon]$ (respectively, $[_{>}\chi,\upsilon]$) denotes the minimalist tree whose root immediately dominates subtrees χ and υ such that χ's root precedes υ's root, and such that χ's root projects over (respectively, is projected over by) υ's root.

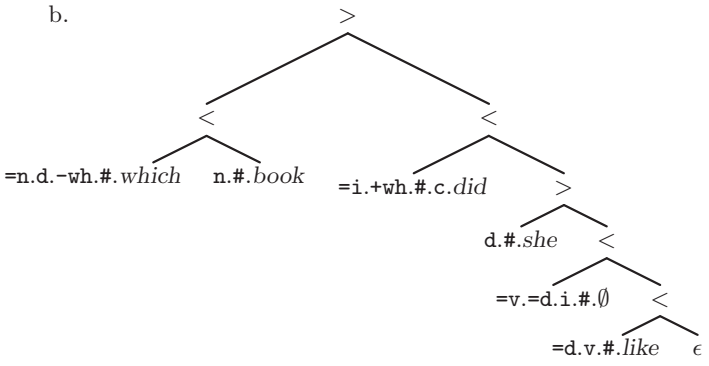
[7]Cf. (1) from above, and also Figure 3 from the appendix with ϕ instead of τ.

[8]Our treatment of relative clauses has been radically simplified for the sake of brevity. We take $\emptyset$ to denote a string of non–syntactic features without phonetic content.

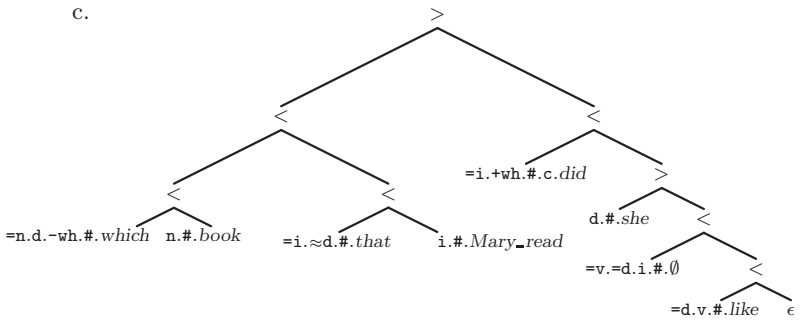
(10) a.



b.



c.



Note that the transition from (10b) to (10c) crucially involves availability of a checked feature instance, i.e. the instance of `d` on *which*, for *adjoin* to be able to apply to *which book did she like* "late." Resorting to this representational option is what distinguishes MG enriched by (countercyclic) *adjoin* from classical MGs, where checked features are radically inert and therefore deleted instantaneously.

7.2.3 Derivational Generative Capacity

Let us now turn to the question as to what the addition of late adjunction implies for the generative capacity of MGs. In the light of the type of example discussed earlier, there is no difference between early and late adjunction for finally resulting trees, or to put it differently, abstracting away from the binding phenomenon the system displays a "Church-Rosser-like" behavior. Therefore neither the weak nor the strong generative capacity of the formalism is affected by adding this type of late adjunction. Yet, this holds only insofar as the adjuncts do not introduce unchecked instances of (move–)licensees that allow subsequent extraction (out) of these adjuncts at some later derivation step. At the same time it is important to note that the kind of restriction required to enforce this actually yields welcome results, since without it we would be able to derive locality violations, such as shown in (11).

(11) *When$_i$ did John wonder [who$_j$ Mary met t_j t_i]

These considerations aside, what will under any circumstances be affected by the introduction of late adjunction is, what could — in the spirit of Becker et al. (1992) — be called the *derivational generative capacity*.[9] As already mentioned, our definition of *adjoin* crucially requires features not to be deleted even after they have been checked by an application of a structure building operation. In order to allow late(r) adjunction in full generality these features have to be present throughout the derivation.[10] In fact, this prevents us from adopting the methods which, in particular, led to the succinct, "chain–based" MG–reformulation (reducing MGs to their "bare essentials") presented in Stabler and Keenan (2000). There, a minimalist tree is represented as a finite sequence of triples of finite strings such that only maximal subtrees with unchecked syntactic features are represented as components. More concretely, only those subtrees are represented which can still be operated on by the operations *merge* or *move*, and each of these subtrees is represented in a highly reduced form: indicating a) the unchecked syntactic features of the subtree's head–label, b) whether the subtree has already a feature checked off or not (denoted by : and

[9]Note that, introducing their notion of derivational generative capacity, Becker et al. restrict their interest to predicate–argument relations, which they formally account for by means of a "coindexing" of predicates and their arguments. This coindexing is straightforwardly realized by assuming that a predicate and its arguments are introduced as dependent on each other, and such a dependence is initialized in a single derivation step. Of course, in this sense, an adjunct is not derivationally dependent on the constituent it modifies, and vice versa.

[10]This is true at least for plain categorial features.

::, respectively),[11] and c) the (narrow) non–syntactic yield of the subtree. Thus, MGs in exactly the sense of this succinct MG–reformulation can be seen as a severe reduction of "classical" MGs. This holds not only w.r.t. the strong generative capacity, but also the elusive notion of derivational complexity, since we are left with just one option for adjunction, namely, the earliest one. In other words, what is a familiar notational shortcut in syntactic representations (cf. item (9h) in our lexicon) becomes an essential part of the theory in the succinct MG–formulation. See (12a)–(12c), which correspond to (10a)–(10c) respectively.

(12) a. $\Big\langle \langle \texttt{+wh.c} , : , \textit{did\_she\_like} \rangle , \langle \texttt{-wh} , : , \textit{which\_book} \rangle \Big\rangle$

b. $\Big\langle \langle \texttt{c} , : , \textit{which\_book\_did\_she\_like} \rangle \Big\rangle$

c. $\Big\langle \langle \texttt{c} , : , \textit{which\_book\_that\_Mary\_read\_did\_she\_like} \rangle \Big\rangle$

Note that, as long as we do not permit late adjunction and restrict our interest to convergent derivations, there is a general finite upper bound on the number of components which must be available. But, to allow for unrestricted late(r) adjunction in such a representation, i.e. adjunction to any maximal projection at any stage of the derivation, an unbounded number of components must be essentially available.

One of the questions arising at this point is the following: when we take into account (only) the "Lebeaux cases" of late adjunction, is it then possible to finitely restrict the number of nodes to which late adjunction can apply without reducing too much the derivational capacity which seems to be necessary for an adequate description of the phenomena discussed? For example, is it possible to revise the definition of *adjoin* given above such that adjunction is only allowed to τ or one of its immediate daughters? The usefulness of such a restriction is straightforward, since it would (re)open the possibility for a succinct MG–formulation again, which in its turn makes the formalism directly amenable to polynomial–time parsing methods (see e.g. Harkema 2000).[12]

[11] That is, "::" serves to denote exactly the unaffected instances of lexical items.

[12] In a thematically related paper dealing with adjuncts in the TAG–framework, Schabes and Shieber (1994) suggest a modified notion of derivation, called *independent derivation*. Contrary to standard *dependent derivations* as defined in Vijay–Shanker (1987), this new mechanism effectively allows multiple adjunction at one and the same node ν. Thus, adjunction constraints valid at ν affect all constituents adjoined to ν irrespective of their ultimate hierarchical order in derived trees. The same effect is automatically captured in MGs as defined here and in Frey and Gärtner (2002), since (i) the MG–operation *adjoin* checks features against the head

7.3 Further Outlook

There are two obvious directions in which to pursue these issues further.[13]

First, it would be important to find out the consequences for the generative capacity of MGs with late adjunction that do not impose the restriction on adjuncts we looked at in Section 7.2.3. This is particularly interesting for MGs enriched with a mechanism for the treatment of relative clause extraposition. If, as we assume, the latter is analyzed in terms of a *(rightward) scrambling* operation, definable in analogy to its "leftward" counterpart from Frey and Gärtner (2002), we seem to be forced to lift the ban on multiple occurrences of competing *scramble–licensees* that was assumed there, following the corresponding constraint on move–licensees. Otherwise, there exist cases of multiple extraposition the structurally adequate derivation of which is possible in the relevant MGs *with* late adjunction but not in those MGs *without*. This means that these two types of MGs would differ not only in terms of their derivational generative capacity but also in terms of (at least) their strong generative capacity. (13) exemplifies such a case.

(13) [[[[Only those papers t_i]$_k$ did [everyone t_j] read t_k] [who was on the committee]$_j$] [which deal with adjunction]$_i$]

A step by step derivation of this example is provided in (14), where α and β are placeholders for *who was on the committee* and *which deal with adjunction*, respectively.

(14) ⋮

a. did [everyone] read [only those papers]
b. [only those papers]$_i$ did [everyone] read t_i
c. [only those papers]$_i$ did [[everyone] [α]] t_i read
d. [only those papers]$_i$ did [everyone t_j] t_i read [α]$_j$
e. [[only those papers] [β]]$_i$ did [everyone t_j] t_i read [α]$_j$
f. [only those papers t_k]$_i$ did [everyone t_j] t_i read [α]$_j$ [β]$_k$

Secondly, the issue of countercyclicity can be systematically further complicated by considering (a) countercyclic *move* and (b) counter-

of the constituent adjoined to, and (ii) the head of the constituent adjoined to is identical to the head of the resulting tree.

[13] As for semantics, it seems that the introduction of countercyclicity is neutral w.r.t. the question as to whether derivation trees or derived trees are interpreted. In case of the former, one would, in order to preserve compositionality, have to employ the "open–property–variable–approach" to restrictive relative clauses introduced by Bach and Cooper (1978) and discussed by Janssen (1982). (Thanks to Shalom Lappin for having raised this question.)

cyclic *merge* in addition. Countercyclic *move* seems to be necessary for the approach to Malagasy adverb placement by Rackowski and Travis (2000), in order to circumvent so-called "freezing violations," i.e. extraction from constituents that have undergone movement at an earlier stage (Thiersch p.c.). Collins (1994), however, provides arguments against this kind of approach. Likewise, it is easy to see that allowing countercyclic *merge* in addition to countercyclic adjunction would jeopardize the Lebeaux–account presented above. Nevertheless, from a formal point of view it would be attractive to find out whether there is a hierarchy in terms of derivational generative capacity ordering these different countercyclic operations.

Appendix

Throughout we let $\neg Syn$ and Syn be a finite set of *non–syntactic features* and a finite set of *syntactic features*, respectively, in accordance with (F1)–(F3) below. We take *Feat* to be the set $\neg Syn \cup Syn$.

(F1) $\neg Syn$ is disjoint from Syn and partitioned into the sets *Phon* and *Sem*, a set of *phonetic features* and a set of *semantic features*, respectively.

(F2) Syn is partitioned into five sets:[14]

Base		a set of *(basic) categories*
M-Select	$= \{ \texttt{=x} \mid \texttt{x} \in \mathit{Base} \}$	a set of *m(erge)–selectors*
A-Select	$= \{ \texttt{≈x} \mid \texttt{x} \in \mathit{Base} \}$	a set of *a(djoin)–selectors*
Licensees	$= \{ \texttt{-x} \mid \texttt{x} \in \mathit{Base} \}$	a set of *licensees*
Licensors	$= \{ \texttt{+x} \mid \texttt{x} \in \mathit{Base} \}$	a set of *licensors*

(F3) *Base* includes at least the category `c`.

Definition 15 An *expression (over Feat)*, also referred to as a *minimalist tree (over Feat)*, is a 5–tuple $\langle N_\tau, \lhd_\tau^*, \prec_\tau, <_\tau, label_\tau \rangle$ obeying (E1)–(E3).

(E1) $\langle N_\tau, \lhd_\tau^*, \prec_\tau \rangle$ is a finite, binary (ordered) tree defined in the usual sense: N_τ is the finite, non–empty set of *nodes*, and $\lhd_\tau^*$ and $\prec_\tau$ are the respective binary relations of *dominance* and *precedence* on N_τ.[15]

(E2) $<_\tau \subseteq N_\tau \times N_\tau$ is the asymmetric relation of *(immediate) projection* that holds for any two siblings, i.e., for each $x \in N_\tau$

[14] Elements from *Syn* will usually be typeset in typewriter mode.

[15] Thus, $\lhd_\tau^*$ is the reflexive–transitive closure of $\lhd_\tau \subseteq N_\tau \times N_\tau$, the relation of *immediate dominance* on N_τ.

different from the root of $\langle N_\tau, \lhd_\tau^*, \prec_\tau \rangle$ either $x <_\tau \mathit{sibling}_\tau(x)$ or $\mathit{sibling}_\tau(x) <_\tau x$ holds.[16]

(E3) label_τ is the *leaf–labeling function* from the set of leaves of $\langle N_\tau, \lhd_\tau^*, \prec_\tau \rangle$ into $\mathit{Syn}^*\{\texttt{\#}\}\mathit{Syn}^*\mathit{Phon}^*\mathit{Sem}^*$.[17]

We take $\mathit{Exp}(\mathit{Feat})$ to denote the class of all expressions over *Feat*.

Let $\tau = \langle N_\tau, \lhd_\tau^*, \prec_\tau, <_\tau, \mathit{label}_\tau \rangle \in \mathit{Exp}(\mathit{Feat})$.[18]

For each $x \in N_\tau$, the *head of x (in τ)*, denoted by $\mathit{head}_\tau(x)$, is the (unique) leaf of τ with $x \lhd_\tau^* \mathit{head}_\tau(x)$ such that each $y \in N_\tau$ on the path from x to $\mathit{head}_\tau(x)$ with $y \neq x$ projects over its sibling, i.e. $y <_\tau \mathit{sibling}_\tau(y)$. The *head of τ* is the head of τ's root. τ is said to be a *head* (or *simple*) if N_τ consists of exactly one node, otherwise τ is said to be a *non–head* (or *complex*).

An $\upsilon = \langle N_\upsilon, \lhd_\upsilon^*, \prec_\upsilon, <_\upsilon, \mathit{label}_\upsilon \rangle \in \mathit{Exp}(\mathit{Feat})$ is a *subexpression of τ* in case $\langle N_\upsilon, \lhd_\upsilon^*, \prec_\upsilon \rangle$ is a subtree of $\langle N_\tau, \lhd_\tau^*, \prec_\tau \rangle$, $<_\upsilon = <_\tau \restriction_{N_\upsilon \times N_\upsilon}$, and $\mathit{label}_\upsilon = \mathit{label}_\tau \restriction_{N_\upsilon}$. Such a subexpression υ is a *maximal projection (in τ)* if its root is a node $x \in N_\tau$ such that x is the root of τ, or such that $\mathit{sibling}_\tau(x) <_\tau x$. $\mathit{MaxProj}(\tau)$ is the set of maximal projections in τ.

An $\upsilon \in \mathit{MaxProj}(\tau)$ is said to *have*, or *display*, *(open) feature f* if the label assigned to υ's head by label_τ is of the form $\beta\texttt{\#}f\beta'$ for some $f \in \mathit{Feat}$ and some $\beta, \beta' \in \mathit{Feat}^*$.[19]

τ is *complete* if its head–label is in $\mathit{Syn}^*\{\texttt{\#}\}\{\texttt{c}\}\mathit{Phon}^*\mathit{Sem}^*$, and each of its other leaf–labels is in $\mathit{Syn}^*\{\texttt{\#}\}\mathit{Phon}^*\mathit{Sem}^*$. Hence, a complete expression over *Feat* is an expression that has category `c`, and this instance of `c` is the only instance of a syntactic feature within all leaf–labels which is preceded by an instance of `#`.

The *phonetic yield of τ*, denoted by $Y_{\mathit{Phon}}(\tau)$, is the string which results from concatenating in "left–to–right–manner" the labels assigned via label_τ to the leaves of $\langle N_\tau, \lhd_\tau^*, \prec_\tau \rangle$, and replacing all instances of non–phonetic features with the empty string, afterwards.

---

[16] $\mathit{sibling}_\tau(x)$ denotes the (unique) sibling of any given $x \in N_\tau$ different from the root of $\langle N_\tau, \lhd_\tau^*, \prec_\tau \rangle$. If $x <_\tau y$ for some $x, y \in N_\tau$ then x is said to *(immediately) project over y*.

[17] For each set M, M^* is the Kleene closure of M, including ϵ, the empty string. For any two sets of strings, M and N, MN is the product of M and N w.r.t. string concatenation. Further, `#` denotes a new symbol not appearing in *Feat*.

[18] Note that the leaf–labeling function label_τ can easily be extended to a total labeling function ℓ_τ from N_τ into $\mathit{Feat}^*\{\texttt{\#}\}\mathit{Feat}^* \cup \{\texttt{<}, \texttt{>}\}$, where `<` and `>` are two new distinct symbols: to each non–leaf $x \in N_\tau$ we can assign a label from $\{\texttt{<}, \texttt{>}\}$ by ℓ_τ such that $\ell_\tau(x) = \texttt{<}$ iff $y <_\tau z$ for $y, z \in N_\tau$ with $x \lhd_\tau y, z$, and $y \prec_\tau z$. In this sense a concrete $\tau \in \mathit{Exp}(\mathit{Feat})$ is depictable in the way done in (10a)–(10c).

[19] Thus, e.g., the expression depicted in (10a) has feature `+wh`, while there is a maximal projection which has feature `-wh`.

For any $\upsilon, \phi \in \mathit{Exp}(\mathit{Feat})$, $[_{<}\upsilon, \phi]$ (respectively, $[_{>}\upsilon, \phi]$) denotes the complex expression $\chi = \langle N_\chi, \lhd^*_\chi, \prec_\chi, <_\chi, \mathit{label}_\chi \rangle \in \mathit{Exp}(\mathit{Feat})$ for which υ and ϕ are those two subexpressions such that $r_\chi \lhd_\chi r_\upsilon$, $r_\chi \lhd_\chi r_\phi$ and $r_\upsilon \prec_\chi r_\phi$, and such that $r_\upsilon <_\chi r_\phi$ (respectively $r_\phi <_\chi r_\upsilon$), where r_υ, r_ϕ and r_χ are the roots of υ, ϕ and χ, respectively.

For any $\upsilon, \phi, \chi \in \mathit{Exp}(\mathit{Feat})$ such that ϕ is a subexpression of υ, $\upsilon\{\phi/\chi\}$ is the expression which results from substituting χ for ϕ in υ.

Definition 16 A *minimalist grammar with generalized adjunction (*abbr. MG^{adj}*)* is a five–tuple $G = \langle \neg\mathit{Syn}, \mathit{Syn}, \mathit{Lex}, \Omega, \mathtt{c} \rangle$ with Ω being the operator set consisting of the structure building functions *merge*, *move* and *adjoin* defined w.r.t. *Feat* as in (me), (mo) and (ad) below, respectively, and with *Lex* being a *lexicon (over Feat)*, i.e., *Lex* is a finite set of simple expressions over *Feat*, and each lexical item $\tau \in \mathit{Lex}$ is of the form $\langle \{r_\tau\}, \lhd^*_\tau, \prec_\tau, <_\tau, \mathit{label}_\tau \rangle$ such that $\mathit{label}_\tau(r_\tau)$ is an element from $\{\mathtt{\#}\}(\mathit{M\text{-}Select} \cup \mathit{Licensors})^*(\mathit{Base} \cup \mathit{A\text{-}Select})\, \mathit{Licensees}^*\, \mathit{Phon}^*\, \mathit{Sem}^*$.

The operators from Ω build larger structure from given expressions by succesively checking "from left to right" the instances of syntactic features appearing within the leaf–labels of the expressions involved. The symbol `#` serves to mark which feature instances have already been checked by the application of some structure building operation.

(me) *merge* is a partial mapping from $\mathit{Exp}(\mathit{Feat}) \times \mathit{Exp}(\mathit{Feat})$ into $\mathit{Exp}(\mathit{Feat})$. For any $\upsilon, \phi \in \mathit{Exp}(\mathit{Feat})$, $\langle \upsilon, \phi \rangle$ is in $\mathit{Domain}(\mathit{merge})$ if for some category $\mathtt{x} \in \mathit{Base}$ and $\alpha, \alpha', \beta, \beta' \in \mathit{Feat}^*$, conditions (i) and (ii) are fulfilled:

(i) the head–label of υ is $\alpha\mathtt{\#=x}\alpha'$ (i.e. υ has m–selector `=x`), and
(ii) the head–label of ϕ is $\beta\mathtt{\#x}\beta'$ (i.e. ϕ has category `x`).

Then,

(me.1) $\mathit{merge}(\upsilon, \phi) = [_{<}\upsilon', \phi']$ if υ is simple, and
(me.2) $\mathit{merge}(\upsilon, \phi) = [_{>}\phi', \upsilon']$ if υ is complex,

where υ' and ϕ' result from υ and ϕ, respectively, just by interchanging the instance of `#` and the instance of the feature directly following the instance of `#` within the respective head–label (cf. Fig. 1).

(mo) *move* is a partial mapping from $\mathit{Exp}(\mathit{Feat})$ into $\mathit{Exp}(\mathit{Feat})$. An $\upsilon \in \mathit{Exp}(\mathit{Feat})$ is in $\mathit{Domain}(\mathit{move})$ if for some $\mathtt{-x} \in \mathit{Licensees}$ and $\alpha, \alpha' \in \mathit{Feat}^*$, (i) and (ii) are true:

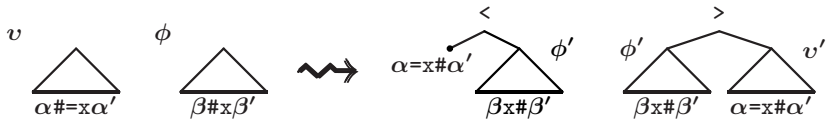


FIGURE 1 *merge*(υ, ϕ) according to (me).

(i) the head–label of υ is α#+xα' (i.e. υ has licensor +x), and
(ii) there is exactly one $\phi \in \mathit{MaxProj}(\upsilon)$ with head–label β#−xβ' for some $\beta, \beta' \in \mathit{Feat}^*$ (i.e. there is exactly one $\phi \in \mathit{MaxProj}(\upsilon)$ displaying −x).

Then,

$$\mathit{move}(\upsilon) = [_{>}\phi', \upsilon'] \,,$$

where $\upsilon' \in \mathit{Exp}(\mathit{Feat})$ results from υ by interchanging the instance of # and the instance of +x directly following it within head–label of υ, while the subtree ϕ is replaced by a single node labeled ϵ. $\phi' \in \mathit{Exp}(\mathit{Feat})$ arises from ϕ by interchanging the instance of # and the instance of −x next to its right within the head–label of ϕ (cf. Fig. 2).

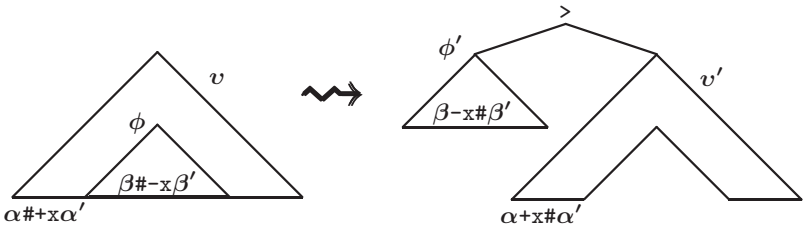


FIGURE 2 *move*(υ) according to (mo).

(ad) *adjoin* is a partial mapping from $\mathit{Exp}(\mathit{Feat}) \times \mathit{Exp}(\mathit{Feat})$ to $\mathcal{P}_{\mathrm{fin}}(\mathit{Exp}(\mathit{Feat}))$.[20] A pair $\langle \upsilon, \phi \rangle$ with $\upsilon, \phi \in \mathit{Exp}(\mathit{Feat})$ belongs to $\mathit{Domain}(\mathit{adjoin})$ if for some category x $\in \mathit{Base}$ and $\alpha, \alpha' \in \mathit{Feat}^*$, conditions (i) and (ii) are fulfilled:

(i) the head–label of υ is α#≈xα' (i.e. υ has a–selector ≈x), and
(ii) there is some $\chi \in \mathit{MaxProj}(\phi)$ with head–label β#xβ' or βxβ'#β'' for some $\beta, \beta', \beta'' \in \mathit{Feat}^*$

[20] $\mathcal{P}_{\mathrm{fin}}(\mathit{Exp}(\mathit{Feat}))$ is the class of all finite subsets of $\mathit{Exp}(\mathit{Feat})$.

Then,

$$adjoin(\upsilon,\phi) = \left\{ \phi\{\chi/[_{<}\chi,\upsilon']\} \;\middle|\; \begin{array}{l} \chi \in MaxProj(\phi) \text{ with head–label } \beta\mathtt{\#x}\beta' \text{ or } \beta\mathtt{x}\beta'\mathtt{\#}\beta'' \text{ for some } \beta,\ \beta',\ \beta'' \in Feat^* \end{array} \right\},$$

where υ' results from υ by interchanging the instances of # and $\approx$x, the latter directly following the former within the head–label of υ. (cf. Fig. 3).

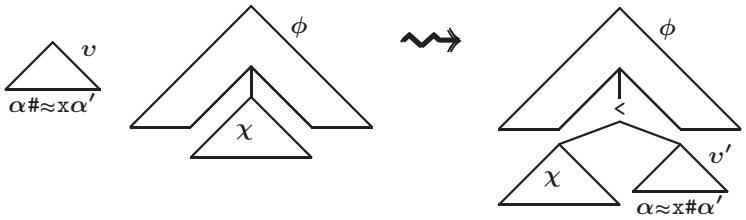


FIGURE 3 Expression from $adjoin(\upsilon,\phi)$ according to (ad).

For each $\mathrm{MG}^{\mathrm{adj}}$ $G = \langle \neg Syn, Syn, Lex, \Omega, \mathtt{c}\rangle$, the *closure of* G, $CL(G)$, is the set $\bigcup_{k\in\mathbb{N}} CL^k(G)$,[21] where $CL^0(G) = Lex$, and for $k \in \mathbb{N}$, $CL^{k+1}(G) \subseteq Exp(Feat)$ is recursively defined as the set

$$\begin{aligned} & CL^k(G) \\ & \cup \{merge(\upsilon,\phi) \mid \langle\upsilon,\phi\rangle \in Domain(merge) \cap CL^k(G) \times CL^k(G)\} \\ & \cup \{move(\upsilon) \mid \upsilon \in Domain(move) \cap CL^k(G)\} \\ & \cup \bigcup\nolimits_{\langle\upsilon,\phi\rangle \in Domain(adjoin)\cap CL^k(G)\times CL^k(G)} adjoin(\upsilon,\phi) \end{aligned}$$

The set $\{\tau \mid \tau \in CL(G) \text{ and } \tau \text{ complete}\}$, denoted by $T(G)$, is the *minimalist tree language derivable by* G. The set $\{Y_{Phon}(\tau) \mid \tau \in T(G)\}$, denoted by $L(G)$, is the *minimalist (string) language derivable by* G.

References

Bach, E. and R. Cooper. 1978. The NP–S analysis of relative clauses and compositional semantics. *Linguistics and Philosophy* 2:145–150.

Becker, T., O. Rambow, and M. Niv. 1992. The derivational generative power of formal systems or Scrambling is beyond LCFRS. Technical Report IRCS-92-38, Institute for Research in Cognitive Science, University of Pennsylvania.

Chomsky, N. 1981. *Lectures on Government and Binding*. Dordrecht: Foris.

[21] $\mathbb{N}$ is the set of all non–negative integers.

Chomsky, N. 1995. *The Minimalist Program*. Cambridge: MIT Press.

Chomsky, N. 2000. Minimalist inquiries. The framework. In R. Martin, D. Michaels, and J. Uriagereka, eds., *Step by Step. Essays on Minimalist Syntax in Honor of Howard Lasnik*, pages 89–155. Cambridge: MIT Press.

Collins, C. 1994. Economy of derivation and the Generalized Proper Binding Condition. *Linguistic Inquiry* 25:45–61.

Frey, W. and H.-M. Gärtner. 2002. On the treatment of scrambling and adjunction in minimalist grammars. In *Proceedings of the Conference on Formal Grammar*, pages 41–52.

Harkema, H. 2000. A recognizer for minimalist grammars. In *Proceedings of the 6th International Workshop on Parsing Technologies*, pages 111–122.

Janssen, T. M. V. 1982. Compositional semantics and relative clause formation in Montague grammar. In J. A. G. Groenendijk, T. M. V. Janssen, and M. B. J. Stokhof, eds., *Formal Methods in the Study of Language*, pages 237–276. Dordrecht: Foris.

Joshi, A. K. and Y. Schabes. 1997. Tree adjoining grammars. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages*, vol. 3, pages 69–124. Berlin, Heidelberg: Springer.

Lebeaux, D. 1991. Relative clauses, licensing, and the nature of the derivation. In S. D. Rothstein, ed., *Perspectives on Phrase Structure. Heads and Licensing*, pages 209–239. New York: Academic Press.

Michaelis, J. 2001. Derivational minimalism is mildly context–sensitive. In M. Moortgat, ed., *Logical Aspects of Computational Linguistics, LNAI 2014*, pages 179–198. Berlin: Springer. Also available at `http://www.ling.uni- potsdam.de/~michael/papers.html`.

Rackowski, A. and L. Travis. 2000. V–initial languages. X or XP movement and adverbial placement. In A. Carnie and E. Guilfoyle, eds., *The Syntax of Verb Initial Languages*, pages 117–141. New York: Oxford University Press.

Schabes, Y. and S. M. Shieber. 1994. An alternative conception of tree–adjoining derivation. *Computational Linguistics* 20:91–124.

Stabler, E. P. 1997. Derivational minimalism. In C. Retoré, ed., *Logical Aspects of Computational Linguistics, LNAI 1328*, pages 68–95. Berlin: Springer.

Stabler, E. P. 1999. Remnant movement and complexity. In G. Bouma, G. M. Kruijff, E. Hinrichs, and R. T. Oehrle, eds., *Constraints and Resources in Natural Language Syntax and Semantics*, pages 299–326. Palo Alto: CSLI Publications.

Stabler, E. P. 2001. Recognizing head movement. In P. de Groote, G. Morrill, and C. Retoré, eds., *Logical Aspects of Computational Linguistics, LNAI 2099*, pages 245–260. Berlin: Springer.

Stabler, E. P. and E. L. Keenan. 2000. Structural similarity. In *Proceedings of the 16th Twente Workshop on Language Technology (TWLT 16) joint with the 2nd AMAST Workshop on Algebraic Methods in Language Processing*, pages 251–265. Iowa City.

Vijay–Shanker, K. 1987. *A Study of Tree Adjoining Grammars*. Ph.D. thesis, University of Pennsylvania.

8

Graph Properties of HPSG Feature Structures

Stephan Kepser and Uwe Mönnich

A general principle for grammars for languages is that they must use *finite* means to produce the infinitely many structures of the languages. When the term *production* is understood algebraically, there are at least two notions of finiteness involved. Firstly in each set of structures each structure is generated, i.e., the result of *finitely* many applications of operators. Secondly, the set of operators as such is also *finite*. For algebraic theories, this is eo ipso the case. But there are linguistic frameworks that are not generating but rather licensing linguistic structures such as Head-Driven Phrase Structure Grammar or GB-Theory. Within these, structures are not generated. Rather only general properties of admissible structures are stated. But these structures must be provided somehow. Thus one may ask if there are algebraic approaches to licensing theories.

We show here that HPSG is non-algebraic in the following sense. There is no finite family of operations such that the members of every finite set of feature structures can be constructed by this finite family of operations. We also show that HPSG feature structures are abstractly recognisable in the sense that for every set of finite HPSG feature structures there exists a (many-sorted) algebra with *finite* sort sets of infinitely many operations such that every feature structure in that set is recognised by the algebra (seen as an abstract automaton). In other words, the first finiteness demand can be fulfilled, the second one cannot.

Proceedings of FGVienna: The 8th Conference on Formal Grammar.
Gerald Penn (ed.).
Copyright © 2008, CSLI Publications.

8.1 Introduction

Formal frameworks for linguistic theories like tree adjoining grammars, minimalist grammars, and context-free tree grammars belong to the generative-enumerative paradigm. This paradigm can be considered as a branch of applied recursive functions theory. Languages, under this view, consist of a set of finite concrete structures that are enumerable by an algorithmic device. The Chomsky hierarchy and its refinements is one of the success stories of this approach towards a general syntactic meta-theory.

Recent years have seen the emergence of a competing paradigm which is not so much inspired by a general theory of algorithms but by methods developed within finite model theory (Ebbinghaus and Flum, 1995). According to this approach languages are to be viewed as sets of (finite) relational structures that satisfy a family of constraints which are stated in a suitable logical specification language. Grammatical structures are not the result of an enumerative process, but they comprise those structures that satisfy all the constraints of a language. Prominent incarnations of this paradigm are the Government and Binding Theory (Chomsky, 1981) and Head-driven Phrase Structure Grammar (HPSG (Pollard and Sag, 1987, 1994)). A systematic comparison between the generative-enumerative and the model theoretic framework can be found in an article by Pullum and Scholz (2001).

What has not been noticed before is the fact that the difference between the generative and the model-theoretic licensing paradigm is parallelled by a concomitant distinction of data structures. To put it in a nutshell: whereas the notion of an initial abstract data structure is fundamental for the generative-enumerative framework, the dual notion of a final abstract object structure plays a similar role within the model-theoretic framework. The definition of an initial abstract data structure specifies in which ways elements of that structure may be generated. The central concepts of this type of data are those of an algebra of a signature, of a congruence relation and of induction. Familiar examples are provided by strings and trees. The dual type of data is characterised by the corresponding notions of coalgebra, bisimulation relation and coinduction. The elements of a final object structure are not described in terms of a representation through their recursive buildup, but via the attributes and methods that characterise their behaviour. The most prominent example for such a structure is that of a labeled graph.

The difference between these two types of data structures is the reason for the association of strings and trees with the generative-enumerative paradigm and of labeled graphs with the model-theoretic

paradigm of syntactic frameworks. Trees can be uniquely characterised by specifying their hierarchical structure, they can be enumerated by means of a production system and they can be fed into an abstract automaton. For graphs no suitable notion of finite automaton has been proposed. Similarly, graph grammars do not generate their output in a way that is patterned by an independently given hierarchical structure which would be intrinsic to this data type.

The notion of generation often entails the existence of a finite set of operators such that every element of the structure under consideration is the result of finitely many applications of these operations on a finite set of constants (nullary operations). One way of interpreting this technical statement is to say that a grammar constructs a language with *finite* means, where there are two aspects of finiteness to be distinguished. Firstly, each structure can be constructed by finitely many applications of operations (this is the notion of generation in algebra). Secondly, there are only finitely many different operators. These properties may be desirable for licensing theories as well. The fact that the data structures underlying licensing theories are graphs does not as such preclude such a possibility. Bauderon and Courcelle (1987) present an algebraic approach to graphs by defining a (generally infinite) set of graph operations such that every finite graph can be constructed by finitely many applications of these graph operations. Terms consisting of graph operation symbols are graph *expressions*, they describe how a graph is constructed. The value of such an expression is of course the graph the construction of which it describes. Thus there is a constructive, algebraic approach to graphs. The important question is then, whether the set of graph operators needed for a family of grammatically well-formed structures is finite or not.

For the particular licensing theory HPSG we will show here that the classes of finite graphs defined by a finite HPSG signature and grammar are indeed such that they cannot be generated by a finite set of graph operations. The set of graph expressions for finite HPSG graphs can in general not be generated by a regular or even context-free tree grammar. This result is independent of the particular logic chosen for HPSG. It just relies on standard assumptions on HPSG signatures and grammars, as they are spelled out, e.g., by Pollard and Sag (1994).

We also show that HPSG is abstractly recognisable. This means that for every set of HPSG feature structures defined by a grammar there exists a (many-sorted) algebra over the possibly infinite set of graph operations where all sort sets are *finite*. This algebra can be seen as an abstract automaton. The automaton is abstract, because there are infinitely many graph operations and hence infinitely many different

sorts. In other words, the states of the automaton are sorted. For each sort, there are only finitely many states, but the set of all states of all sorts is infinite. This is why the automaton cannot really be used for recognition or generation. The notion *recognisable* remains abstract. But at least this finiteness property can be established for HPSG.[1]

8.2 HPSG-Style Feature Structures

The data structures underlying HPSG are so-called feature structures. The probably simplest approach to them, and the one we will follow here, is to regard them as relational structures. On this view, sorts are unary predicates and features are binary functional predicates. HPSG feature structures are required to be *totally well-typed* and *sort resolved*. Sort-resolvedness, demanding that the sorts partition the universe, is of no particular relevance here. Well-typedness restricts the admissible correlations between sorts and features. Each sort is correlated with a set of admissible features. And for each such feature there is an indication listing the admissible set of sorts on the target node. *Total* well-typedness additionally requires each admissible feature to be present. Thus a signature for HPSG is a triple $\Delta = (\mathcal{S}, \mathcal{F}, A)$ where $\mathcal{S}$ is a set of sorts (unary predicate symbols), $\mathcal{F}$ is a set of features (binary predicate symbols), and $A : \mathcal{S} \times \mathcal{F} \to \wp(\mathcal{S})$ is an *appropriateness function*. We will restrict our attention to *finite* signatures, i.e., signatures with $\mathcal{S}$ and $\mathcal{F}$ being finite sets. The restrictions imposed by a finite HPSG signature can be expressed as a first-order sentence $\mathcal{A}_\Delta$, as was shown by Aldag (1997) in his master's thesis:

$$
\begin{aligned}
&\forall x \bigvee_{S\in\mathcal{S}} S(x)\\
&\wedge \bigwedge_{S_1,S_2\in\mathcal{S},S_1\neq S_2} \forall x \neg(S_1(x) \wedge S_2(x))\\
&\wedge \bigwedge_{F\in\mathcal{F}} \forall x,y,z((F(x,y) \wedge F(x,z)) \to y = z)\\
&\wedge \bigwedge_{S\in\mathcal{S},F\in\mathcal{F},A(S,F)\neq\emptyset} \forall x(S(x) \to \exists y\, F(x,y) \wedge \bigvee_{S'\in A(S,F)} S'(y))\\
&\wedge \bigwedge_{S\in\mathcal{S},F\in\mathcal{F},A(S,F)=\emptyset} \forall x(S(x) \to \neg\exists y\, F(x,y))
\end{aligned}
$$

One would now expect that a grammar is a first-order sentence over the signature. But a typical HPSG principle is somewhat simpler. It is a boolean combination of path equality expressions and sort statements. A path equality expression is a formula of the following kind:

$$
\begin{aligned}
\exists y_1,\ldots y_n \exists z_1,\ldots z_k \;\; & F_1(x,y_1) \wedge \cdots \wedge F_n(y_{n-1},y_n) \wedge\\
& G_1(x,z_1) \wedge \cdots \wedge G_k(z_{k-1},z_k) \wedge y_n = z_k
\end{aligned}
$$

where $F_1,\ldots,F_n,G_1,\ldots,G_k \in \mathcal{F}$. Such a formula stipulates that the

[1] Note that the fact that each single finite graph can be constructed by finitely many applications of graph operations does in no way imply that every (possibly infinite) *set* of feature structures is abstractly recognisable.

two feature paths $F_1 \dots F_n$ and $G_1 \dots G_k$ lead to the same node. A sort statement is a formula of the following kind:

$$\exists x_1, \dots x_n \; F(x, x_1) \wedge \dots \wedge F(x_{n-1}, x_n) \wedge S(x_n)$$

where $F_1, \dots, F_n \in \mathcal{F}$ and $S \in \mathcal{S}$. It states that the node at the end of the path $F_1 \dots F_n$ is of sort S. An HPSG principle is a boolean combination of these kinds of formulae with the free variable x being universally quantified since the principle is to be true at every node. A grammar is simply a conjunction of principles. We note that for the purpose of the present paper it is immaterial if a grammar is of the limited formula class just described or if it is any first-order sentence.

A relational structure $\mathfrak{A}$ over a signature Δ is a model for a grammar Γ, iff $\mathfrak{A} \models \mathcal{A}_\Delta \wedge \Gamma$. We will only consider *finite* structures. This is linguistically justified if one keeps in mind that such a structure is the linguistic analysis of an utterance. It seems unclear what an infinite analysis is supposed to mean.

The two most prominent alternatives for a formalisation of HPSG are feature logics (see, e.g., Rounds (1997) for a survey) or certain modal logics (see, e.g., (Blackburn and Spaan, 1993, Kracht, 1995)). Since proponents of both alternatives typically propose a weak logic, there often is a translation into first-order logic available. For the particular example of Speciate Reeantrant Logic by King (1989) this was shown by Aldag (1997). For modal logics, several such translation methods are provided by Ohlbach, Nonnengart, de Rijke, and Gabbay (2001).

8.3 Graphs as Logical Structures

Almost from the beginning, feature structures were considered as particular types of graphs. Graph-theoretically they are rooted directed hypergraphs where the sorts are unary edges. We will now define hypergraphs following the proposal by Courcelle (1990a,b) and quoting it freely where appropriate. Since all the graphs we are dealing with are hypergraphs, we will omit the prefix *hyper-*. A signature or ranked alphabet Σ is a finite set L of labels together with a function $\rho : L \to \mathbb{N}$ equipping each label with an arity. In the case of HPSG, the signature consists of the unary sort symbols $\mathcal{S}$ and the binary feature symbols $\mathcal{F}$.

Definition 17 Let Σ be a signature. A *concrete graph* is a quintuple $G = \langle V, E, lab, inc, prt \rangle$ where

- V is a set whose elements are the vertices of the graph;
- E is a set whose elements are the edges;
- $lab : E \to L$ is an edge labelling function;

- $inc : E \to V^*$ associates with each edge e the sequence of its vertices, a sequence of length $\rho(lab(e))$;
- prt is a sequence in V^* of vertices, the *ports*. The length of this sequence is the type of the graph G.

Thus a concrete graph consists of a set of vertices and a set of labelled edges between the vertices. Vertices are labelled by unary edges. A sort of HPSG turns out to be a unary labelled edge that is attached to a vertex. The type of an edge e is the arity of its label, i.e, $type(e) = \rho(lab(e))$. The ports are needed for technical purposes only. A *graph* is the equivalence class of all isomorphic concrete graphs. The set of graphs over a signature Σ is denoted by $\mathcal{G}_\Sigma$.

A graph is *finite*, iff both V and E are finite. As explained before, we restrict our attention to finite graphs. This definition of graphs is very general. Edges are for example not restricted to be binary, they may connect more than two vertices. For a model of HPSG-style feature structures, this generality is certainly not needed. Rather the subset of graphs that are HPSG-style feature structures are *directed multi graphs*. Multi graphs are graphs where the range of ρ is the set $\{0, 1, 2\}$, i.e., each edge is at most binary. A graph is directed if the range of inc is generally not symmetric, i.e., edges have a start and an end vertex. HPSG feature structures are also rooted, but this property is of no relevance for us.

Bauderon and Courcelle (1987) define three families of graph operations to turn a set of graphs into a many-sorted algebra of graphs. (Other such complete families of graph operations are presented, e.g., by Engelfriet (1997) or Courcelle (1997).) The first family of operations is *disjoint sum*. Let G and H be two graphs of types n and n'. We can assume their sets of vertices and edges to be disjoint. Then $G \oplus H$ is $\langle V_G \cup V_H, E_G \cup E_H, lab_G \cup lab_H, inc_G \cup inc_H, prt_G \hat{} prt_H \rangle$ of type $n+n'$.

The second operation is the *port redefinition*. This operation renames or "forgets" ports. If G is a graph of type n and $\alpha : \{1, \ldots, k\} \to \{1, \ldots, n\}$ then the graph after port redefinition is $\langle V_G, E_G, lab_G, inc_G, prt_G(\alpha) \rangle$ of type k.

The third operation is the *port fusion*. It fuses port vertices, i.e., the operation identifies groups of vertices. For every equivalence relation R on the set $\{1, \ldots, n\}$ there is a mapping fu_R taking a graph G of type n and returning G' that is obtained from G by identifying the ports that are in the same equivalence class of R.

Note that each of the three families of operations contains denumerably many operators. The set of all graphs together with these three families of operations forms a many-sorted algebra, where the sorts are

just the types of the graphs. Bauderon and Courcelle (1987) also define a (many-sorted) algebra of so-called *graph expression* $\mathcal{GE}$ in the following way. It is the term algebra of the (sorted) operation symbols of the above described graph operations together with the following constants: 0 (denoting the empty graph), 1 (denoting the graph consisting of a single vertex), a for each label $a \in L$ (denoting the graph consisting of a single edge of type $\rho(a)$ of label a together with its $\rho(a)$ vertices). An element of this term algebra is called a *graph expression*. It can and should be seen as an instruction for constructing a graph. The graph is the value of the expression: Since the term algebra is the free algebra of this signature of graph operations there exists a unique homomorphism from the algebra of graph expressions to the algebra of graphs. The value of a graph expression is exactly the value of this homomorphism. It is now interesting to see that the homomorphism is surjective.

Proposition 4 (Bauderon and Courcelle, 1987) *Every finite graph is the value of a graph expression.*

A graph expression is a term, hence a tree. We can therefore use graph expressions to define particular sets of graphs.

Definition 18 A set of graphs is *context-free* iff the set of graph expressions denoting this set is regular. A set of graphs is *extended context-free* iff the set of graph expressions denoting it is context-free.

As usual, a set of trees is regular iff it is generated by a regular tree grammar or, equivalently, accepted by a tree automaton. A set of trees is context-free iff it is generated by a context-free tree grammar (Rounds, 1970). The notion of *context-freeness* for graphs stems from context-free graph grammars such as hyperedge replacement grammars. It is well known that a set of graphs is the language of a context-free graph grammar just in case it is the value of a regular set of graph expressions (Lautemann, 1988).

We need two notions for the size of a graph. The first one, *width*, is roughly the size of the signature of the expression for that graph. The second one, *treewidth*, is a measure on how tree-like a graph is.

Definition 19 Let g be a graph expression. The *width* $wd(g)$ is the maximal sort of a symbol of the graph expression algebra occurring in g. The *width* of a graph G is $wd(G) := min\{wd(g) \mid val(g) = G\}$.

Definition 20 Let G be a graph. A *tree decomposition* of G is a pair (T, f) where T is an unrooted unoriented tree and $f : V_T \to \wp(V_G)$ is a mapping such that
(1) $V_G = \bigcup\{f(i) \mid i \in V_T\}$;

(2) for every edge e of G there is a set $f(i)$ such that all vertices of e are in $f(i)$;
(3) if $v \in f(i) \cap f(j)$, then $v \in f(k)$ for every k belonging to the unique loop-free path from i to j in T.

The width of a tree decomposition is defined as $max\{|f(i)| \mid i \in V_T\} - 1$, and the *treewidth* of G is the smallest width of a tree decomposition of G.

As was shown by Courcelle, the treewidth of a directed multi graph provides a lower bound for its width.

Proposition 5 (Courcelle, 1992) *Let G be a finite directed multi graph. Then $twd(G) < wd(G)$.*

A natural and indeed very powerful choice of a logical language for graphs is monadic second-order logic of vertices and and edges (MS_2). Monadic second-order logic extends first-order logic by the addition of set variables and quantification over set variables. Thus the language offers existential and universal quantification over vertices, edges, sets of vertices, and sets of edges. Many sets of graphs can be defined in MS_2 such as, e.g., planar graphs, 3-colourable graphs, or graphs with a Hamiltonian cycle. The translation of the appropriateness function and a grammar of a finite HPSG signature into MS_2 is a simple task, in which no second order quantification is needed.

A set C of graphs of sort n is called *abstractly recognisable*, iff there exists a many-sorted algebra $\mathfrak{A}$ over the possibly infinite signature of graph operations with *finite* universes (sort sets), a homomorphism $h : \mathcal{GE} \to \mathfrak{A}$, and a finite subset FS of A_n such that $C = h^{-1}(FS)$. The triple $(h, \mathfrak{A}, FS)$ is called an automaton, the set FS is the set of final states. The homomorphism h is uniquely given due to the fact that $\mathcal{GE}$ is the free algebra over the signature of graph operations.

Proposition 6 (Courcelle, 1990b) *Every MS_2-definable set of graphs is abstractly recognisable.*

8.4 Graph Properties of HPSG Feature Structures

We can now present a finiteness result for HPSG, namely that for each set of HPSG feature structures defined by a grammar there exists a graph algebra over a possibly infinite signature of operations where each sort set is finite such that each feature structure is generated by this algebra. In other words

Theorem 7 *For every finite HPSG signature and grammar, the set of models of that signature and grammar is abstractly recognisable.*

As shown in the previous sections, every finite HPSG signature and every grammar are MS_2-definable. So, let $\mathcal{A}_\Delta$ be the MS_2-sentence defining the signature and Γ be the MS_2-sentence defining the grammar. By Proposition 6, the set of models of $\mathcal{A}_\Delta \wedge \Gamma$ is abstractly recognisable.

This finiteness result is quite a weak one, because due to the infinite signature there are infinitely many sort sets or universes. That means that the algebra as a whole is not finite. It is only the universe for each sort that is finite.

It would therefore be desirable to obtain a stronger finiteness result, namely one in which the whole algebra is finite. But such a result cannot be established, as we will show now. We demonstrate that there is a set of finite feature structures such that there is no graph algebra with a *finite* graph operator signature which generates this set. To do so, let us consider a special set of graphs, namely grids. Grids are special types of planar graphs. They are the result of gluing squares in lines and columns, forming a rectangular shape. An example of a 6×3 grid is given in Figure 1.

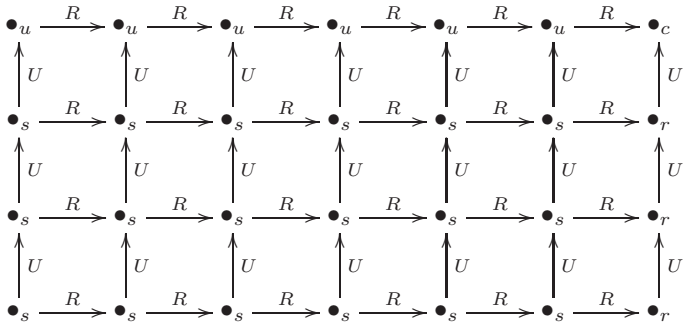


FIGURE 1 A 6×3 grid.

It is not difficult to see that HPSG formalisations allow the construction of grids. Consider the following signature $\langle \{s, u, r, c\}, \{U, R\} \rangle$ with the appropriateness conditions

$$\begin{array}{llll} s & U & s, u & \qquad u \quad R \quad u, c \\ s & R & s, r & \qquad r \quad U \quad r, c \end{array}$$

where $s \quad U \quad s, u$ reads, "sort s has obligatory feature U and the sort at the end vertex of U may be s or u." We suppose the grammar to be empty. The set $\mathcal{GR}$ of all finite graphs that are models of this signature contains all finite grids such as the one in Figure 1. It does, of course, contain many more graphs that are not grids. But this is not

of importance for us. If desired, the set of graphs can be restricted by adding principles so that only grid-like graphs are in the set. Indeed, Courcelle (1997) shows how to define finite grids in monadic second-order logic.

Theorem 8 *The set $\mathcal{GR}$ has unbounded width.*

This is shown as follows. $\mathcal{GR}$ contains all finite grids. The treewidth of an $n \times k$ grid is $min(n, k)$ (see (Bodlaender, 1998)). Therefore there is no bound on the treewidth of $\mathcal{GR}$. Since all feature structures are multi graphs, there is by Proposition 5 no bound on the width of $\mathcal{GR}$, either.

This theorem expresses that the signature for the graph expressions generating $\mathcal{GR}$ is infinite.

Proposition 9 *The set $\mathcal{GR}$ of graphs is neither context-free nor extended context-free.*

This is a simple consequence of the fact that every tree grammar has a finite set of productions. Thus the set of operators in terms is finite. But the width of $\mathcal{GR}$ is unbounded.

Corollary 10 *HPSG feature structures are in general neither context-free nor extended context-free.*

Unfortunately this also means that standard techniques from universal algebra or automata theory cannot be applied. It is a philosophical questions whether graphs similar to grids should be regarded as appropriate for linguistic analyses. There are good reasons to believe that HPSG grammarians do not have grid-like feature structures in mind when they think about models of their grammars. From the point of view of a licensing theory this would mean that additional principles have to be added that generally state desirable graph properties of the intended model. But most logics for HPSG, including (R)SRL and modal logics, are not powerful enough to formulate principles that exclude grid-like feature structures (and only these).

There is an interesting analogon in formalisations of GB-theory. Rogers (1998) showed that under the assumption of free indexation, an assumption that is standard in this theory, the construction of grids cannot be excluded either.

One obvious way to remedy the problem for HPSG is to demand of a class of models to be generable by a context-free graph grammar. Such a demand cannot be postulated as a principle but would have to be extrinsic. An interesting consequence of such a demand would be that treebanks for HPSG could be queried in time *linear* in the size of the treebank using MS_2 as a query language.

References

Aldag, B. 1997. *A Proof Theoretic Investigation of Prediction in HPSG*. Master's thesis, Seminar für Sprachwissenschaft, University of Tübingen.

Bauderon, M. and B. Courcelle. 1987. Graph expressions and graph rewritings. *Mathematical Systems Theory* 20:83–127.

Blackburn, P. and E. Spaan. 1993. A modal perspective on the computational complexity of attribute value grammar. *Journal of Logic, Language, and Information* 2:129–169.

Bodlaender, H. L. 1998. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science* 209:1–45.

Chomsky, N. 1981. *Lectures on Government and Binding*. Dordrecht, Holland: Foris Publications.

Courcelle, B. 1990a. 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.

Courcelle, B. 1990b. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation* 85:12–75.

Courcelle, B. 1992. The monadic second-order logic of graphs III: tree-decompositions, minors and complexity issues. *Informatique Théoretique et Applications* 26:257–286.

Courcelle, B. 1997. The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, ed., *Handbook of Graph Grammars and Computing by Graph Transformation*, pages 313–400. World Scientific Publishing.

Ebbinghaus, H.-D. and J. Flum. 1995. *Finite Model Theory*. Springer-Verlag.

Engelfriet, J. 1997. Context-free graph grammars. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages, Vol III: Beyond Words*, pages 125–213. Springer.

King, P. J. 1989. *A Logical Formalism for Head-Driven Phrase Struture Grammar*. Ph.D. thesis, University of Manchester.

Kracht, M. 1995. Is there a genuine modal perspective on feature structures? *Linguistics and Philosophy* 18:401–458.

Lautemann, C. 1988. Decomposition trees: Structured graph representation and efficient algorithms. In M. Dauchet and M. Nivat, eds., *CAAP '88, 13th Colloquium on Trees in Algebra and Programming*, *LNCS 299*, pages 28–39.

Ohlbach, H. J., A. Nonnengart, M. de Rijke, and D. Gabbay. 2001. Encoding two-valued nonclassical logics in classical logic. In A. Robinson, ed., *Handbook of Automated Reasoning*, pages 1403–1486. North Holland.

Pollard, C. and I. A. Sag. 1987. *Information Based Syntax and Semantics, Vol. 1: Fundamentals*. No. 13 in Lecture Notes. CSLI.

Pollard, C. and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

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. Springer-Verlag.

Rogers, J. 1998. *A Descriptive Approach to Language-Theoretic Complexity*. CSLI Publications.

Rounds, W. 1970. Mappings and grammars on trees. *Mathematical Systems Theory* 4(3):257–287.

Rounds, W. 1997. Feature logics. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language*, pages 475–533. Elsevier.

9

At First Blush on Tenterhooks: About Selectional Restrictions Imposed by Nonheads[1]

Jan-Philipp Soehn and Manfred Sailer

9.1 Introduction

Selection is one of the major mechanisms in formal grammar, such as HPSG (Pollard and Sag, 1994), and is used for two purposes which are usually applied in the same contexts: First, syntactic and semantic arguments are specified and their relation to the selector is indicated. This is usually expressed in terms of thematic role assignment. Second, the properties of elements that may be combined are restricted. When a verb selects its complements, it imposes both kinds of restrictions at the same time. If on the other hand the selector is a nonhead, as in the case of modifiers, the question of thematic role assignment is less clear. When it comes to complementizers which select a clause, only the second function of selection appears to be relevant. In this paper we will present evidence that all nonheads, including complements, can impose this kind of selectional restrictions on the heads they combine with.

As a starting point we will look at PPs that contain a *unique nominal*

[1]This paper originated within the Special Research Center 441 at the University of Tübingen. The full data and a more comprehensive discussion of the approach is given in Soehn (2003). We are grateful to Koenraad Kuiper, Stefan Müller, Frank Richter, Beata Trawiński and to the FGVienna reviewers for helpful comments and to Guthrun Love and Carmella Payne for help with the challenges of English.

Proceedings of FGVienna: The 8th Conference on Formal Grammar.
Gerald Penn (ed.).
Copyright © 2008, CSLI Publications.

complement (UNC), i.e., a noun that only occurs in combination with a particular preposition. English examples are *on tenterhooks* (*nervous, worried*), or *in a trice* (*in an instant*). Note that the underlined nouns cannot be used in any other environment. For German a large number of PPs with UNCs have been identified. We will show that the data can be adequately described if we allow nonheads to select heads. Thus we will generalize the HPSG treatment of adjuncts (which select by MOD) and of functional words such as determiners and complementizers (which select by SPEC) to all nonheads. This analysis requires only minor changes to the HPSG architecture, whereas alternative attempts either fail to account for the data or would require a far stronger mechanism. In this context we will discuss a constructional analysis (Riehemann, 2001) and a collocational analysis (Sailer and Richter, 2002). While the need for constructions for other phenomena is indisputable (Fillmore et al., 1988, Jackendoff, 1995), we will demonstrate that our new architecture can not only account for the data that Sailer and Richter have used to motivate their collocational approach, but also implies a number of restrictions on the kinds of collocational effects that may arise. These predictions are consolidated by a data base of approximately 300 highly collocationally restricted lexical items in German.

9.2 Data

The data we use as motivation for our analysis are *unique nominal complements* (UNC) of prepositions. They can only occur within a PP, as illustrated by the example *auf Anhieb* (*at first go*). To use Tseng (2000)'s terminology: *Anhieb* is an *internal trigger* of the preposition *auf*. In the following examples, the unique element is underlined and the preposition is printed in bold face.

(15) Das "PC-Kummerbuch" ist **auf** Anhieb auf großes Interesse
the PC-sorrow-book is at first-go on great interest
gestoßen.
hit
'The "PC Troubleshooting Companion" triggered interest right away.'

It is important to note that combining another preposition with *Anhieb* is ungrammatical, see (16) for comparison to a non-unique noun.

(16) **auf**/***bei** Anhieb (at first go) vs.
auf den/beim ersten Versuch (at first attempt)

Similar English expressions are *by rote* (*mechanical*) or *in a trice* (*as quickly as possible*). For German, we have compiled about 300 highly

collocationally restricted lexical elements from the phraseological literature (Dobrovol'skij, 1988, Dobrovol'skij and Piirainen, 1994, Fleischer, 1989, 1997). This list contains about 90 PPs with UNCs. These comprise:

(17) **im** Brustton (der Überzeugung) (with utter conviction), **zum** Steinerweichen (bitterly), **von**/**durch** Geisterhand (by magic), **um** Haaresbreite (by a hair's breadth), **im** Nachhinein (with hindsight), **aus**/**nach** Herzenslust (ad libitum), **ohne** Unterlass (without intermission)

These expressions have a highly restricted environment, as only one or a few prepositions are possible.

We are now faced with the question of how these expressions can be analyzed. One possibility would be to put the entire P+UNC sequence in the lexicon as a multiword lexeme (such as *of course*). This would not be adequate for three reasons: Firstly, the whole expression is syntactically formed like a regular PP and can be interpreted compositionally (see 18). Following Fillmore et al. (1988) we want to adopt phrasal lexical entries only for phrases with idiosyncratic properties. Secondly, the complements of the preposition can be modified (see 19). However this is not a property of all UNCs and the modifiability has to be examined case by case. Thirdly, a UNC can be coordinated with another UNC or even with a free noun (see 20 and 21). From here it follows that the relevant PPs are not unstructured entities. Rather they are formed like regular PPs and even their semantics is decomposable. Consequently, regarding them as multiword lexemes would postulate too much irregularity of the language.

(18) Sie erhalten dieses Modell **auf** Anfrage.
you get this model on request
'This model is available upon request.'

(19) Er nahm an dem Gewinnspiel **aus** reinem Daffke teil.
He took at the lottery ex pure fun part
'He took part in the lottery just for fun.'

(20) Das Mädchen schmetterte ein Lied **aus** [Herzenslust und
the girl belted out a song from heart.delight and
Leibeskräften].
body.power
'The girl belted out a song to the top of her bent and with all her heart.'

(21) Damit brachte er sie **in** [Teufels Küche und jede Menge
with-it brougth he her in devil's kitchen and every amount
Schwierigkeiten].
trouble
'With that he got her in hot water and big trouble.'

The next step is to make sure that the UNC combines with (and only with) the correct preposition. Moreover, where a UNC cannot be modified, the possibility of modification has to be excluded. In the following section we will take a closer look at various approaches to analyzing the data.

9.3 Alternative Analyses

9.3.1 Preposition selecting the NP

In all grammar frameworks, selection or subcategorization is a central issue, which is realized within HPSG by valence lists in the CATEGORY value of a sign. Subcategorization is rather general: a typical domain would be for example a verb subcategorizing a subject and an object, both noun phrases being capable of receiving a particular theta-role. To account for the vast variety of grammatical expressions, values of subcategorized signs are specified only to a certain extent. Underspecification is the mechanism which allows the selection of many different signs and keeps subcategorization to the required degree of generality.

If we were to use a preposition to select a UNC as its complement, we would encounter problems. First of all, even if we introduced the possibility of selecting particular lexemes, we could not prevent the UNC from occurring elsewhere, because of underspecification. Additionally, if we defined a preposition that selected a particular UNC we would have to postulate a certain amount of homophonous prepositions, where each one subcategorized for a different UNC. This postulation would be rather counterintuitive. Even worse, the appearance of a UNC in coordination, such as in (20) and (21) would be rendered ungrammatical.

Thus the present selection mechanism for HPSG is not sufficient, as selection of the argument by the preposition would not be able to correctly restrict the occurrence of a UNC to a particular preposition and it would wrongly exclude the coordination data above.

9.3.2 Constructional Approach

Riehemann (2001) offers a concrete analysis of idioms within HPSG based on the ideas of Construction Grammar (Fillmore et al., 1988, Sag, 1997). She describes idioms as units of several words, i. e. constructions. One of her central ideas is to relate each word in an idiom

with its literal counterpart. For our data this approach is not adequate because the lack of a literal lexical counterpart is a defining feature of UNCs such as *Anhieb* or *Daffke*. If we were to introduce lexical entries such as these, we would not be able to prevent the words from occurring freely in other contexts. Alternatively, we can adopt a constructional analysis in which the UNCs are *idiomatic words* not related to any literal counterpart.

Riehemann (2001) introduces a complex mechanism to restrict the distribution of idiomatic words to the correct contexts, defining new sorts and attributes which keep track globally of all parts of idiomatic expressions that occur in an utterance. The components of this mechanism are not independently motivated, and, as Riehemann (p. 207) acknowledges, the occurrence of pronominalized parts of idioms cannot be handled.

We conclude that the UNC data does not fit Riehemann's basic assumptions on idioms, and thus does not justify the application of her complicated apparatus.

9.3.3 Collocation

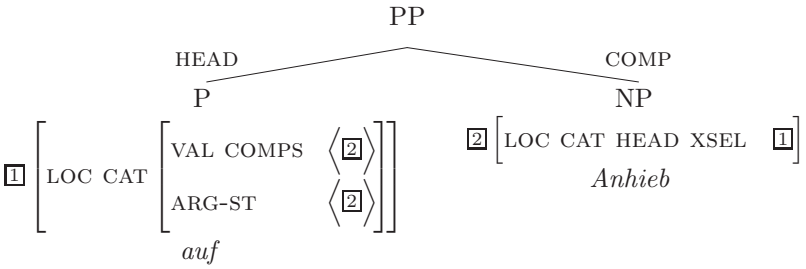
Sailer (2003) and Sailer and Richter (2002) propose a collocation module, introducing the COLL feature. The value of this feature is a list. This list is empty for non-lexical signs, and contains a single sign for each lexical sign. The authors define a grammar principle which imposes the following constraint: the overall utterance must be identical with the sign in the COLL lists of all the lexical signs for this utterance. Thus the entire utterance is available at the COLL value of every lexical sign, so that occurrence restrictions can be specified. This means that any property of the utterance can be mentioned in the lexical entry of such a sign. For example, the word *Anhieb* specifies in its COLL list a sign dominating a PP with the head *auf*, *Anhieb* being the complement of *auf*.

This module is explicitly designed for elements with limited distribution. However it is extremely powerful, because the element of the COLL list makes available the structure of the entire expression available. Therefore the mechanism does not impose any principled constraints on the kinds of occurrence restrictions that may be found. For our data, i.e. very local dependencies, the mechanism seems far too powerful and requires quite a few changes within the grammar. In the next section we will argue for a different approach, using an existing mechanism in HPSG and making only minor changes.

9.4 External Selection

As indicated already in the introduction, there is a mechanism of external selection provided within the HPSG framework. External selection means that nonheads select their heads. This has been assumed for adjuncts (which select by MOD) and for functional words such as determiners and complementizers (selecting by SPEC). We will now generalize this mechanism and apply it to all nonheads by specifying a feature XSEL (for *external selection*) for the sort *head*. By XSEL the features MOD and SPEC are subsumed and become dispensable. Pollard and Sag (1994, p. 55) themselves state that these two features are analogous in their use. The function of MOD and SPEC is now incorporated into XSEL, and everything that they have accomplished can be achieved by the new attribute. The value of XSEL is *synsem*. This results in the analysis illustrated in (22), the details of which will be examined below.

(22) Structure of the PP *auf Anhieb*:



The changes to be made affect only two points of our grammar. First, as already noted, we define an attribute XSEL for the sort *head* and remove MOD and SPEC from the signature. As we have introduced XSEL for the sort *head*, the value percolates via the Head Feature Principle (HFP). Second, we redefine the SPEC-Principle as Principle of External Selection, stated in (23).

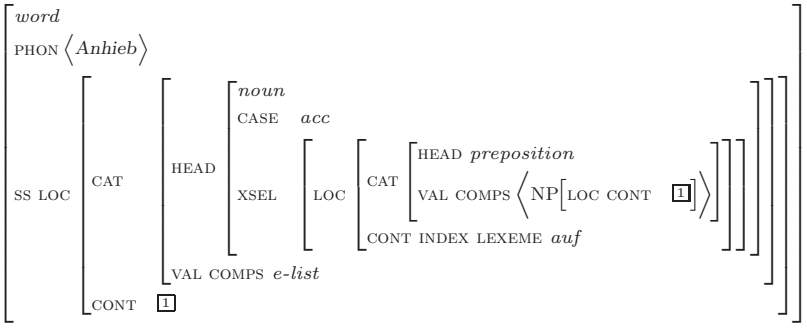
(23) Principle of External Selection (PXS):
The SYNSEM value of a word is identical to the XSEL value of each element on its ARG-ST list.
In a head-adjunct phrase and in a head-marker phrase the SYNSEM value of the HEAD-DTR is identical to the XSEL value of the nonhead daughter.

Ideally we could have formulated the PXS in a uniform way on all headed structures with a selectional relation between the daughters. Then the PXS would enforce identity between the nonhead's XSEL and the head's SYNSEM value. However, in argument-raising constellations

a *synsem* object may occur on several COMPS lists, while still only occuring on one ARG-ST list.[2] Therefore we refer to the ARG-ST list for complements, instead of giving a structural account of the XSEL requirements of complements. Head-filler structures are exempt from the PXS, as there is no selectional relation between head and filler.

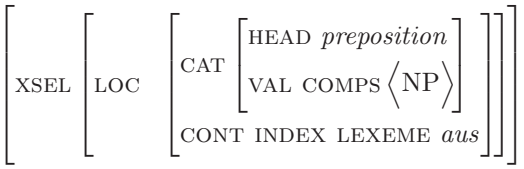
Using the attribute XSEL, a UNC can select a particular preposition by the LEXEME feature. The introduction of a mechanism to select particular lexemes will be deferred to section 9.5. What we get by these adaptations has already been illustrated in (22), where *Anhieb* (*first go*) selects the preposition *auf* (*at*) through the XSEL feature. Additionally, we give the lexical entry of *Anhieb* in (24). Note that there is a restriction on the content: the CONTENT of the complement of the preposition must be identical to the CONTENT of *Anhieb* itself. This excludes the occurrence of modifiers with *Anhieb*.

(24) Outline of the lexical entry of *Anhieb*:



In (25) we have sketched the XSEL value of *Daffke* for comparison. *Daffke* simply selects the preposition *aus*. As there are no further restrictions on the complement NP of this preposition, it follows from the normal selection mechanism that modification is possible, like in *aus reinem Daffke* (*just for pure fun*).

(25) The XSEL value of *Daffke*:



[2]Following Meurers (2000), we assume that argument raising takes place only via the COMPS list.

In principle the content of complement NPs are available for marking semantic restrictions. This allows further specification of the character of modification in the lexical entry. Ultimately it is possible for the UNC to select its preposition.

9.5 Selection of Lexemes

In the previous section we have examined how PPs with unique nominal complements can be accounted for. Some of these expressions go together with one particular verb, as in *zu Potte kommen* (*get to the point*) or *in Mitleidenschaft ziehen* (*to affect*). With our PXS we can account for this data too. We simply specify a *synsem* object in the XSEL value, in this case of *Potte*, that triggers the preposition *zu*. This preposition has another XSEL value, requiring the verb *kommen* (see 26).[3] The verb *kommen* cannot yet be selected as we are missing an important tool: the possibility to select a particular lexeme.

(26) The XSEL value of *Potte*:

Krenn and Erbach (1994) have proposed a method of lexeme selection, introducing an attribute LEXEME below the CONTENT INDEX feature. This has three advantages: the LEXEME value is below SYNSEM and thus available for subcategorization; it percolates along the head line (i.e., a projection has the same LEXEME value as its lexical head); and a pronoun (always sharing the INDEX with its antecedent according to the HPSG Binding Theory) also gets the same LEXEME value as its antecedent.[4] Nevertheless, their approach has one major disadvantage: Only nouns have an INDEX value of their own, which leaves open the question of what to do with verbs (lacking an INDEX) or adjectives (sharing the INDEX with the modified noun).

The solution we propose is a new definition of the INDEX feature for all parts of speech. We separate LEXEME from the traditional INDEX

[3] The PXS makes correct predictions for cases of argument raising, as assumed for German verbal complexes (Meurers, 2000). In (i) the PP *zu Potte* is realized as a complement of the modal verb *sollte*. Still, it only occurs on the ARG-ST list of the verb *kommen*, whose LEXEME value is as required in the XSEL value in the lexical entry of *Potte* in (26).

(i) dass Peter zu Potte kommen sollte
that Peter to pot come should 'that Peter should get to the point'

[4] See (30) for illustration.

features PERSON, NUMBER and GENDER, pooled together below a new feature PHI (see 27). As subsorts of *lexeme*, we introduce a new atomic sort for each lexeme.

(27) The new INDEX value:

$$\begin{bmatrix} \text{INDEX} & \begin{bmatrix} \text{LEXEME} & \textit{lexeme} \\ \text{PHI} & \begin{bmatrix} \text{PERSON} & \textit{person} \\ \text{NUMBER} & \textit{number} \\ \text{GENDER} & \textit{gender} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

These changes have no consequences for nouns. For verbs and prepositions we leave the PHI values underspecified, nevertheless LEXEME receives a non-trivial value. In the lexical entries of adjectives we define the PHI values to be identical with those of the modified noun, but give each adjective its individual LEXEME value. We also adapt the SEMANTICS-PRINCIPLE in a way in which INDEX values are inherited along the syntactic head (whereas all other values like NUCLEUS percolate along the semantic head). This facilitates a selection of individual lexemes and extends our analysis of UNC to combinations with fixed verbs as in (26).

9.6 Some Empirical Predictions

In the preceeding sections we presented a selectional analysis of UNCs. A combination of three factors made this possible: (i) the generalized use of selection by nonheads (by XSEL), (ii) the fact that the CONTENT value of a sign is part of its SYNSEM value (see 24), and (iii) the possibility of lexeme selection (by LEXEME). In this section we will continue to explore the potential of this approach. We will show specifically that the interaction of these three factors accounts for the phenomena that Sailer and Richter provide as evidence for their unconstrained collocational mechanism: cranberry words (Richter and Sailer, 2003) and polarity items (Sailer and Richter, 2002). Furthermore, a set of non-trivial constraints on potential collocational restrictions follow from our approach which have been empirically confirmed by the above-mentioned corpus of approximately 300 distributionally restricted lexical elements.[5]

9.6.1 Restrictions on Lexeme Selection

Richter and Sailer (2003) discuss so called *cranberry words* (CW), i.e., lexical elements that can only occur together with a particular other

[5]See Webelhuth and Ackermann (1994) and Everaert and Kuiper (1996) for other applications of this method.

lexeme. This cleary subsumes the UNCs considered so far. They distinguish three possible cases, depending on the selectional relation between the CW and its collocate: either the CW selects its collocate (28a), or the collocate selects the CW (b), or there is no direct selectional relation (c).

(28) a. jemandem **Angst**/ einen **Schrecken**/ *Entsetzen einjagen
someone fear/ a fright/ horror in.chase
'to frighten sb.'

b. Tacheles **reden**/ *sprechen
goal talk/ speak 'talk straight'

c. **schimpfen**/ *meckern wie ein Rohrspatz
grumble/ grumble like a reed.sparrow
'inveigh vehemently '

Richter and Sailer (2003) claim that the (a)-case is problematic because HPSG does not provide the means for selecting individual lexemes. With the new treatment of lexeme selection illustrated in (27), this argument loses ground. The case in (28b) is resolved by the XSEL mechanism because we have allowed all nonheads to impose selectional restrictions on the heads with which they combine. Together with the possibility of lexeme selection, the data conform to our proposal as well.

Finally in (28c), there is no direct selection between the CW *Rohrspatz* and its collocate, the lexeme *schimpfen* (*grumble*). Nonetheless, the NP headed by the CW is selected by *wie* (*like*), whose projection acts as an adjunct to the verb *schimpfen*. Therefore the preposition *wie* occurs on the XSEL value of the CW and the verb *schimpfen* on the XSEL value of the preposition. Thus we are in the same situation as with *zu Potte kommen* in (26).

Our approach can accurately handle the data discussed above. Moreover, we predict that there are certain restrictions on lexeme selection. In particular a head cannot select a certain adjunct lexeme. This follows from the fact that we have generalized the potential of nonheads to select heads by XSEL, but we have not enlarged the selectional potential of heads. Our collection of 300 German expressions with CWs does not contain a single counter-example to this restriction, and there are abundant examples of lexeme selection of and by complements, and of adjuncts that select particular heads.

We also incorporate a locality restriction. A selector has access to the LEXEME value of a directly selected element, and, as shown for *zu Potte kommen*, to the LEXEME value of signs that are externally selected by these. We predict that there are no CWs which ask for a

specific **complement** of a verb (instead of a specific verb). The hypothetical constellation is sketched in (29), where the bold face NP is a complement to the verb V, and the collocate of the CW. In this case the information about the lexeme of the complement NP is not available at the VP level, i.e. part of the SYNSEM value of the VP. The prediction is fully confirmed in our corpus.

(29) $[_{VP}$ V **NP**$]$ $[_{PP}$ P CW$]$

On the other hand, the apparent non-locality of lexeme selection is not a problem. The data show that lexeme information must be present at the phrasal projection, (30a), and must be shared between a pronoun and its antecedent, (30b,c).

(30) a. Das hat mir [einen riesigen **Schrecken**] eingejagt
this has me [a huge fright] in.chased
'This gave me a big fright.'

b. Der **Schrecken**$_i$, **den**$_i$ mir das eingejagt hat, war riesig.
the fright which me this in.chased has was huge
'The fright which he gave me was huge.'

c. Das war ein riesiger **Schrecken**$_i$ für mich. Und ausgerechnet du
this was a huge fright for me and just you
hast **ihn**$_i$ mir eingejagt.
have it me in.chased
'This was a huge shock for me and you were the one who caused it.'

These non-local effects, which can cross root sentences (see 30c), result from the inclusion of the lexeme information as part of the INDEX value. In contrast, Richter and Sailer (2003) would be required to assume that collocations are possible across root sentences in general.

We have shown two restrictions on potential lexeme-specific co-occurrence requirements. They are a direct consequence of the XSEL mechanism and appear to be empirically robust. In the analysis of Richter and Sailer (2003) it is not possible to express these constraints, let alone have them be the independent outcome of the way their collocational mechanism is built.

9.6.2 CONTENT Restrictions

The collocational requirements of the German verb *fackeln* (*dither*) are discussed in Sailer and Richter (2002). The authors base their study on a corpus investigation, according to which the verb occurs together with *nicht lange* (*not for long*) in 86% of the cases (i.e. 241 out of 280). See (31):

(31) Der neue Geschäftsführer fackelte nicht lange.
the new manager dithered not long
'The new manager didn't dither for very long.'

They demonstrate that the adverb *lange* must be considered a modifier, i.e., as not selected by the verb *fackeln*. This seems, therefore, to be a counter-example to the restrictions on XSEL because heads cannot impose lexeme restrictions on their modifiers. However Sailer and Richter (2002) give a more refined characterization of the occurrence restrictions of *fackeln*: In 95% of the corpus data and in all of the introspectively grammatical data, a negation and a durational modifier that have scope over the verb *fackeln* are present, but not necessarily *nicht* and *lange*, see (32). This variation is expected in the XSEL approach: again, there is no lexeme restriction imposed by the head verb on its modifier.

(32) Keiner hat einen Moment gefackelt, bevor er ins Wasser
nobody has a moment dithered before he into the water
gesprungen ist.
jumped has
'Nobody dithered for a single moment before jumping into the water.'

Before we turn to the analysis of *fackeln*, we should address a remaining issue in connection with the XSEL value. The PXS in (23) and the HFP determine the XSEL values of all signs that are part of a phrase that will eventually occur as a nonhead daughter in a sentence. Nothing is said, however, about the XSEL value of signs which occur along the head projection line of a root sign. To avoid spurious ambiguity, we can simply assume that the SYNSEM value of the root sign is the XSEL value in these cases. The notion of a "root sign" is needed independently in grammar (Richter, 1997).[6] In accordance with Riehemann (2001, p. 189f.) we assume a subsort of *sign* for this purpose, which we will call *root-sign*. The XSEL value of root signs is then determined by the constraint in (33).

(33) *root-sign* → $\left[\text{SYNSEM} \quad \boxed{1}\left[\text{LOC CAT HEAD} \quad \left[\text{XSEL} \quad \boxed{1}\right]\right]\right]$

As an effect of the constraint in (33) the CONTENT value of an utterance is in principle accessible to the XSEL specification of every word in the utterance. This can be done by recursively accessing the XSEL value of the *synsem* object in the XSEL value. We can use this property

[6] Root properties include the presence of illocutionary force and the absence of unfilled gaps.

to account for the *fackeln* data of Sailer and Richter, in which the distributional restrictions of the verb *fackeln* (*dither*) are characterized in terms of the particular CONTENT values of the utterances in which it may occur.

Ignoring details of the semantic representation, Sailer and Richter give a description of a CONTENT value which contains a negation and a durational modifier, both having scope over the semantic contribution of the verb *fackeln*. Let δ be this description. The occurrence restrictions on *fackeln* can then be captured in the following lexical entry.

(34) Outline of the lexical entry of *fackeln*:

$$\left[\begin{array}{l}\textsc{phon}\ \langle \textit{fackeln} \rangle \\ \textsc{syns loc}\ \left[\begin{array}{l}\textsc{cat}\ \left[\begin{array}{l}\textsc{head}\ \left[\begin{array}{l}\textit{verb} \\ \textsc{xsel}\ \left[(\textsc{loc cat head xsel})^*\ \textsc{loc cont}\ \ \delta\right]\end{array}\right] \\ \textsc{arg-st}\ \left\langle \textsc{np} \right\rangle\end{array}\right]\end{array}\right]\end{array}\right]$$

where δ describes a *content* object iff it contains a negation, a durative operator and the lexical content of *fackeln*, where the negation and the durative operator both have scope over the latter.

In (34) we use a Kleene-star operator with paths in the AVM. The notation is to be interpreted as describing the existence of a path of the given form. Even though this is not standard in HPSG, a formalization of HPSG such as RSRL (*Relational Speciate Re-entrant Language*, see Richter, 2000) will provide a way to formalize the idea expressed in this rather intuitive notation.

It can be seen that the XSEL requirements of *fackeln* are met in sentence (31). The XSEL value of the sentence in this case is identical to its own SYNSEM value. Since *fackeln* is the syntactic head of the sentence, this very same *synsem* object occurs in its XSEL value. Thus the XSEL LOC CONT value of *fackeln* is identical to the CONT of the root sentence. This content was described by δ in (31). The absence of a negation or a durational modifier would lead to a violation of the lexical specification of the XSEL value of *fackeln*.

If we consider the occurrence of *fackeln* in embedded sentences, the non-trivial effects of the Kleene-star operator in (34) can be observed.

(35) Niemand behauptet, dass Peter lange fackelt.
nobody claims that Peter long dithers
'Nobody claims that Peter dithers for long.'

In (35) it is only the CONT value of the root clause that contains the negation, the durational modifier and the verb *fackeln*. This CONT value appears as the XSEL LOC CONT value of the highest verb, *be-*

hauptet (*claims*). *Fackeln* is the lexical head of the complement clause to the verb *behauptet*. Thus the XSEL value of *fackeln* is identical to the SYNSEM value of *behauptet*. Consequently, the XSEL LOC CAT HEAD XSEL LOC CONT value of *fackeln* is identical to the CONT value of the root clause. As this path is part of the specification in the lexical entry of *fackeln*, the occurrence restriction of the verb is captured correctly.

This example shows that the non-local effect of the occurrence restriction on *fackeln* is fully accounted for in our approach, while we correctly exclude the possibility of constraints specific to the particular lexemes that occur, to the phonology of the required context elements or to details of the syntactic structure. On the other hand, the CONTENT of the utterance is available for imposing constraints.

9.7 Conclusion

In this paper we have offered a more consize generalization of the selectional mechanism of HPSG. We have shown that this allows us to handle the phenomenon of PPs with unique nominal complements. The same mechanism extends naturally to other phenomena of distributional restrictions. Our proposal relies above all on the overall architectural design of HPSG. This design restricts selection to *synsem* objects, where lexeme information and the content are part of *synsem*. The discussion in section 9.6 revealed that the interaction of these components leads to interesting and empirically valid predictions concerning what kinds of collocational restrictions are possible. Nonetheless, the question of whether it is adequate to include lexeme information and the entire content within *synsem* derserves further discussion.[7]

References

Dobrovol'skij, D. 1988. *Phraseologie als Objekt der Universallinguistik*. Verlag Enzyklopädie, Leipzig.

Dobrovol'skij, D. and E. Piirainen. 1994. Sprachliche Unikalia im Deutschen: Zum Phänomen phraseologisch gebundener Formative. *Folia Linguistica* 27(3–4):449–473.

Everaert, M. and K. Kuiper. 1996. Theory and data in idiom research. In L. McNair et al., ed., *CLS32: the Parasession on Theory and Data in Linguistics*, pages 43–57. Chicago Linguistic Society.

Fillmore, C., P. Kay, and M. O'Connor. 1988. Regularity and idiomaticity in grammatical constructions: The case of *let alone*. *Language* 64:501–538.

Fleischer, W. 1989. Deutsche Phraseologismen mit unikaler Komponente — Struktur und Funktion. In G. Gréciano, ed., *Europhras 88*, pages 117–126.

[7] See Sailer (to appear) for some thoughts regarding this question.

Fleischer, W. 1997. *Phraseologie der deutschen Gegenwartssprache*. Niemeyer, Tübingen, 2nd ed.

Jackendoff, R. 1995. The boundaries of the lexicon. In M. Everaert, E.-J. v. d. Linden, A. Schenk, and R. Schreuder, eds., *Idioms. Structural and Psychological Perspectives*, pages 133–165. Lawrence Erlbaum Associates.

Krenn, B. and G. 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.

Meurers, W. D. 2000. *Lexical Generalizations in the Syntax of German Non-Finite Constructions*. Phil. dissertation, Eberhard-Karls-Universität Tübingen. Published as *Arbeitspapiere des SFB 340*, Nr. 145.

Pollard, C. and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press.

Richter, F. 1997. Die Satzstruktur des Deutschen und die Behandlung langer Abhängigkeiten in einer Linearisierungsgrammatik. Formale Grundlagen und Implementierung in einem HPSG-Fragment. In E. Hinrichs, W. D. Meurers, F. Richter, M. Sailer, and H. Winhart, eds., *Ein HPSG-Fragment des Deutschen. Teil 1: Theorie*, Arbeitspapiere des SFB 340, Nr. 95, pages 13–187. Universität Stuttgart and Universität Tübingen.

Richter, F. 2000. *A Mathematical Formalism for Linguistic Theories with an Application in Head-Driven Phrase Structure Grammar*. Phil. dissertation, Eberhard-Karls-Universität Tübingen.

Richter, F. and M. Sailer. 2003. Cranberry words in formal grammar. In C. Beyssade, O. Bonami, P. Cabredo Hofherr, and F. Corblin, eds., *Empirical Issues in Formal Syntax and Semantics*, vol. 4, pages 155–171. Paris, Presses Universitaires de Paris-Sorbonne.

Riehemann, S. Z. 2001. *A Constructional Approach to Idioms and Word Formation*. Ph.D. thesis, Stanford University.

Sag, I. A. 1997. English relative clause constructions. *Journal of Linguistics* 33:431–483.

Sailer, M. 2003. Combinatorial semantics and idiomatic expressions in Head-Driven Phrase Structure Grammar. Phil. Dissertation, Eberhard-Karls-Universität Tübingen. Published as *Arbeitspapiere des SFB 340*, Nr. 161.

Sailer, M. to appear. Distributionsidiosynkrasien: Korpuslinguistische Erfassung und grammatiktheoretische Deutung. In *Tagungsband der 39. Jahrestagung des IDS*. Mannheim.

Sailer, M. and F. Richter. 2002. Not for love or money: Collocations! In G. Jäger, P. Monachesi, G. Penn, and S. Wintner, eds., *Proceedings of the Conference on Formal Grammar*, pages 149–160.

Soehn, J.-P. 2003. *Von Geisterhand zu Potte gekommen. Eine HPSG-Analyse von PPs mit unikaler Komponente*. Magisterarbeit, Universität Tübingen, Seminar für Sprachwissenschaft.

Tseng, J. L. 2000. *The Representation and Selection of Prepositions*. Ph.D. thesis, University of Edinburgh.

Webelhuth, G. and F. Ackermann. 1994. German idioms: An empirical approach. *Studies in the Linguistic Sciences* 24:455–471.

10

A New Application for Raising in HPSG: Complex Prepositions

Beata Trawiński

One of the most popular techniques used in HPSG-based studies to describe linguistic phenomena is the raising mechanism. Besides ordinary raising verbs or adjectives, this tool has been applied for handling verbal complexes and discontinuous constituents, among other phenomena. In this paper, a new application for raising within the HPSG paradigm will be discussed, thereby investigating data from the prepositional domain. We will analyze linguistic properties of word combinations in German consisting of a preposition, a noun, and another preposition (such as *auf Grund von* ('by virtue of')), thus arguing that raising is the most appropriate method for satisfactorily describing the crucial syntactic features which are typical for those expressions. The objective of this paper is thus to demonstrate the efficiency of the raising mechanism as used in HPSG, and therefore, to emphasize the importance of designing a satisfactory uniform theory of raising within this grammar framework.

10.1 Introduction

In describing linguistic phenomena, grammar frameworks apply various techniques that are generally accepted and well established, although they might be differently instantiated in various linguistic theories, depending on the formal tools provided by a given theory. Besides concepts such as movement, binding, or case assignment, the idea of raising plays an important role in many formal approaches to natural language.

Proceedings of FGVienna: The 8th Conference on Formal Grammar.
Gerald Penn (ed.).
Copyright © 2008, CSLI Publications.

Particularly frequently used is the raising technique in HPSG-based studies, especially in studies of German.

In this paper, we will discuss a new application for raising within the HPSG grammar framework in the tradition of Pollard and Sag (1994), thereby investigating data from the prepositional domain. We will analyze the syntax of word sequences commonly labeled "complex prepositions" (CPs) consisting of a preposition, a noun, and another preposition ($P_1N_1P_2$). Although CPs can certainly be considered to be a cross-linguistic phenomenon, we will focus exclusively on German data, because they provide very explicit and convincing linguistic evidence which motivates and supports our approach.[1] However, we assert that the analysis proposed here for German can also be applied to other languages such as Polish or English.

Presenting our analysis for German CPs and depicting parallels between this approach and the analysis of German verbal complexes, our objective is to indicate the efficiency of the raising technique as used in the lexicalist constraint-based grammar systems such as HPSG, thus demonstrating the need of designing a satisfactory uniform theory of raising within this grammar framework.

10.2 Raising in HPSG

Besides ordinary raising verbs and adjectives, the raising mechanism is used in HPSG-based studies for handling several linguistic issues such as verbal complexes (cf. Hinrichs and Nakazawa 1989, Meurers 2000), or discontinuous constituents (cf. De Kuthy 2000).

To illustrate how argument raising in terms of the HPSG functions, we will look at the essential aspect of the German verbal complex analysis in style of Hinrichs and Nakazawa (1989). According to this analysis, the lexical entries of German auxiliaries are specified to subcategorize for verbal complements, as well as to raise the arguments of their complements. Thus, the German auxiliary *will* ('wants') in the structure below selects the verb *lesen* ('read') first, and then the arguments of *lesen*, the NP *das Buch* ('the book') and the NP *Peter* ('Peter'). This idea underlies most current HPSG approaches to verbal complexes in Germanic and Romance languages.

[1] On "complex prepositions" in various languages see e.g. Beneš (1974), Buscha (1984), Lindqvist (1994), Meibauer (1995), Quirk and Mulholland (1964), Schröder (1986).

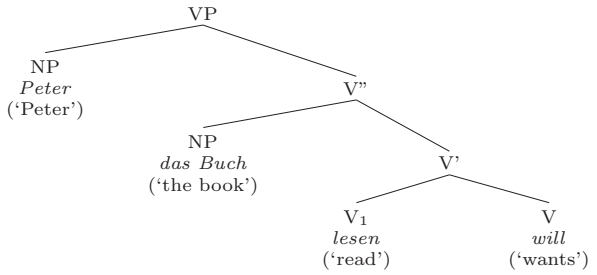


FIGURE 1 The structure of the VP *Peter das Buch lesen will* ('Peter wants to read the book')

10.3 Complex Prepositions: Empirical Characteristic and Possible Analyses

In this section, we will discuss the issue of "complex prepositions" in German, and we will show that their specific syntactic properties can convincingly be explained by dint of the same method.

We have taken into account word combinations such as those in (30).

(30) an Hand von ('by means of'), in Hinblick auf ('in terms of'), in Verbindung mit ('in connection with'), mit Hilfe von ('by dint of') ...

Expressions in (30), when combined with NPs, result in PPs, acting as modifiers within the entire sentence (cf. (31)).

(31) *In Bezug auf Privatsphäre* gibt es im WWW immer noch
in regard to private_sphere is there in_the WWW still yet
keine einheitlichen Richtlinien.
no uniform rules
'With regard to privacy, there are still no uniform rules in the WWW.'

However, the interdependence between the particular elements of those expressions seems to defy standard constraints on the PP structure of German. To illustrate this, we will consider a typical PP in (32).

(32) in einer engen Verbindung mit den Beratern
in a close connection with the advisers
'in close connection with the advisers'

The standard analysis for such PPs assumes that the preposition *in* ('in') acts as the head of the entire phrase taking the NP as its complement. The selected NP is headed by the noun *Verbindung* ('connection') and contains the adjective *engen* ('close') and the determiner

einer ('a'). Furthermore, we have the PP *mit den Beratern* ('with the advisers'), which is selected by the noun *Verbindung* as its complement and can be omitted without causing ungrammaticality (cf. Figure 2).

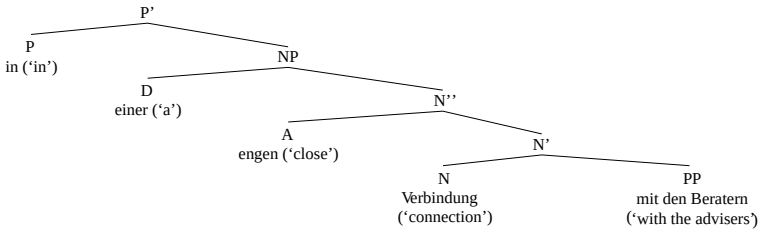


FIGURE 2 The structure of the PP *in einer engen Verbindung mit den Beratern* ('in a close connection with the advisers')

Trying to apply the above approach to an analysis of PPs containing CPs presents several problems. To show this, we will consider one of the CPs combined with an NP, which looks very similar to the PP in (32) (cf. (33)).

(33) in Verbindung mit diesem Problem
in connection with this problem
'in connection with this problem'

Using PPs such as those in (33) in contexts exemplified in (34), we can observe many contrasts with the traditional PPs such as those in (32).

(34) In Verbindung mit diesem Problem möchte ich darauf
in connection with this problem would_like I DA_on
hinweisen, dass ...
point_out that
'In connection with this problem, I would like to point out that ...'

First of all, the noun *Verbindung* cannot syntactically select for a determiner or a quantifier, nor it can be combined with possessive pronouns or prenominal genitives (cf. (35a)).[2] Secondly, it cannot be mod-

[2]However, the definiteness information can be provided directly by P_1s, since $P_1N_1P_2NP$ sequences as well as other PPs allow for expressions referred to as preposition-determiner contraction (e.g. *in dem* → *im*). Such expressions can be considered as a special kind of prepositions, that additionally carry the definiteness specification. For an analysis proposal for preposition-determiner contraction within the HPSG paradigm see Winhart (1997).

ified (cf. (35b) and (35c)).[3] Finally, the PP *mit den Beratern* ('with the advisers') cannot be deleted (cf. (35d)).

(35) a. in *einer/ *der/ *seiner/ *Peters Verbindung mit diesem
in a/ the/ his/ Peter's connection with this
Problem ...
problem

b. in *enger/ *unerwarteter Verbindung mit diesem Problem ...
in close/ unexpected connection with this problem

c. in [Verbindung mit diesem Problem], *die uns betrifft ...
in connection with this problem which us concerns

d. * in Verbindung ...
in connection

Based on these observations, the following assumption can be made: The string *in Verbindung mit* ('in connection with') in the PP exemplified in (33) is a lexical category evincing prepositional character. Thus, Fries (1988) assumes for these PPs that the preposition heading the entire phrase is a projection of three lexical categories which form together a complex lexical category, in this case, a preposition *in Verbindung mit*. This complex preposition then selects an NP forming a prepositional phrase (cf. Figure 3).

The main problem with the Fries's analysis consists in the assumption that the preposition *mit* ('with') belongs to the complex preposition and cannot form a constituent with the NP *diesem Problem*. However, there are several data demonstrating the opposite.

[3]However, there are a couple of cases in German where the nouns allow modification (cf. (i) quoted after Gisbert Fanselow, p.c.):

i. *In deutlichem Gegensatz zu / in großem Unterschied zu seinen*
in clear opposition to / in big difference to his
Behauptungen haben wir Tom niemals mit Maria sprechen sehen.
claims have we Tom never with Maria talk seen

Nevertheless, the number of nouns appearing within discussed PPs which allow for such modification is marginal in German and the set of adjectives approved within such expressions is limited to a very small semantical class. Moreover, no other types of adjuncts are possible within the PPs such as those in (i). Because of their irregular collocation-like character, I do not account for data such as those in (i) as arbitrative for my analysis. Instead, I presume another part of grammar to be responsible for licensing of such expressions. For considerations in handling collocational phenomena within the HPSG framework see e.g. Richter and Sailer (2002).

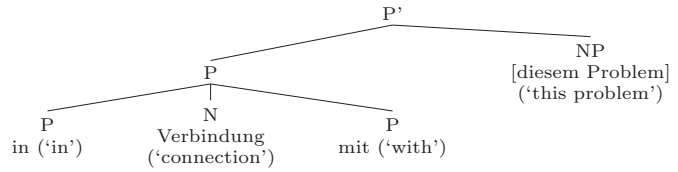


FIGURE 3 The structure of the PP *in Verbindung mit diesem Problem* ('in connection with this problem') proposed in Fries (1988)

Firstly, the P_2NP combinations where the preposition is realized by *von* ('of') can be replaced by the genitive; this replacement of *von* adheres to the restrictions on distribution for postnominal genitives and *von*-PPs in German (cf. (36a)). Secondly, the discussed sequences can be substituted by *wo/da* expressions as in (36b), which are usually handled as proforms for PPs.

(36) a. mit Hilfe ??von dem Buch/ des Buches
with help of the book/ the book$_{GEN}$
'by dint of the book'

b. in Verbindung womit/damit
in connection WO_with/DA_with
'in connection with what/with it'

These observations imply that the discussed sequences form a constituent. Thus, another analysis seems to arise, that assume P_1N_1 combinations to constitute complex lexical categories, requiring prepositional complements (cf. Figure 4).

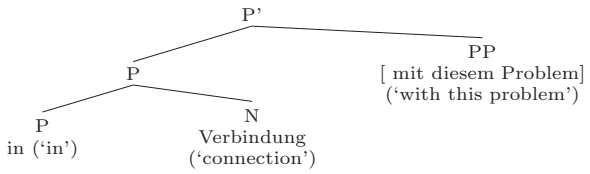


FIGURE 4 The structure of the PP *in Verbindung mit diesem Problem* ('in connection with this problem') assuming the *in Verbindung* string to be a complex lexical category

However, the following fact argues against the analysis in Figure 4: There is a type of nouns in German that allows for two options in realizing the dative case. While the first eventuality relates to suffixless forms, the second one relates to forms ending in *-e*. The choice of a given

form is usually determined by stylistic effects. Examples in (37) show that dative nouns of the discussed declension class can occur within P_1N_1 sequences in both forms.

(37) a. im Verlauf/Verlaufe von Jahrhunderten
in course/course_e of centuries
'in the course of the centuries'

b. im Fall/Falle von Mängeln
in case/case_e of deficit
'in case of deficit'

These examples illustrate that the declension form of N_1s is determined not by P_1N_1 combinations, but by the same factors that otherwise determine the form of inflection realization. Therefore, the data above clearly eliminate the analysis in Figure 4.

Further on, we will consider one more possible analysis, assuming prepositions heading $P_1N_1P_2$NPs as selecting for two arguments: a noun and a PP, which would result in structures such as those in Figure 5.

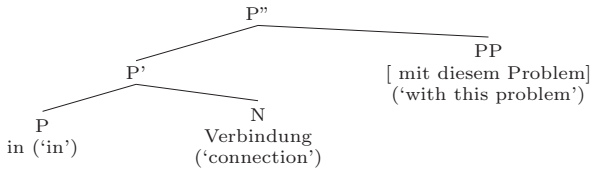


FIGURE 5 The structure of the PP *in Verbindung mit diesem Problem* ('in connection with this problem') assuming *in* to select for two complements

However, this assumption seems unmaintainable for the following reason: It cannot enforce that whenever a noun x appears, a PP headed by a preposition y is required. In consequence, ungrammatical PPs such as those in (38) cannot be ruled out.

(38) a. * in Verbindung zu diesem Problem
in connection to this problem

b. * in Bezug mit diesem Problem
in regard with this problem

Rather, the assumption seems plausible that syntactic properties of P_2NP sequences are determined by N_1s since these properties are identical with the properties of PPs selected by the corresponding nouns in their free occurrences.

All these observations seem to indicate the following: P_2NP sequences such as *mit diesem Problem* in (33) act as arguments of N_1s such as *Verbindung* in (33) in terms of being determined by these nouns with regard to their syntactic properties such as the form of the preposition heading these PPs. However, the discussion on constituency of $P_1N_1P_2NP$ expressions above indicates that P_2NP sequences are realized syntactically by P_1s.

This idea can easily be formalized within the HPSG paradigm by use of the raising mechanism.

10.4 Using Raising Mechanism

Based on the above observations, we assume two uses of prepositions: the raising and the non-raising use. The preposition *in* in (39a) occurs in a non-raising context, while the preposition *in* in (39b) occurs in a raising context.

(39) a. in einer engen Verbindung mit den Beratern
in a close connection with the advisors
'in close connection with the advisors'

b. in Verbindung mit diesem Problem
in connection with this problem
'in connection with this problem'

Our assumption is that both strings *mit den Beratern* in (39a) and *mit diesem Problem* in (39b) act as arguments of *Verbindung* in terms of being determined by the noun *Verbindung* with regard to their syntactic properties. We expect both *mit diesem Problem* and *mit den Beratern* to be selected by *Verbindung* syntactically. Thus, in both cases, we proceed according to the standard methods of handling relational nouns selecting prepositional arguments. This explains why the PP *mit diesem Problem* shares grammatical properties with the PP *mit den Beratern* and other ordinary PPs.

Furthermore, we assume that the preposition *in* in (39b) in opposition to *in* in (39a), which subcategorizes the saturated NP, selects first the noun *Verbindung* (which does not realize its complement) and then *in* selects the complement of *Verbindung*, the PP *mit diesem Problem*. That is, by virtue of an appropriate lexical principle of grammar specifying the valence of prepositions (cf. Figure 8), the complement of the noun *Verbindung* is raised by *in* to become the complement of *in*, and be realized by *in* syntactically.

Thus, avoiding redundancies in the lexicon, we specify only one lexical entry for *in*, thereby underspecifying the information about its

argument. In Figure 6 we can see the relevant part of the lexical entry of the preposition *in* in AVM notation.[4]

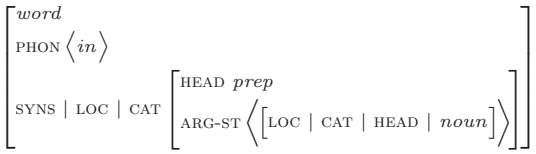


FIGURE 6 The relevant part of the lexical entry of the preposition *in* ('in')

The only information about potential arguments of *in* which this lexical entry provides is that *in* can take only one argument, and this argument has to be a noun. Here, information about the selection requirements of that noun will not be specified; nor will information about the selection requirements of the preposition *in* be specified.

The syntactic selection properties of *in* are licensed by a lexical constraint on the mapping of the elements of the ARG-ST list to the valence lists. For prepositions, the principle on mapping of the elements of the ARG-ST list to the valence lists is traditionally assumed to have the form as in Figure 7.

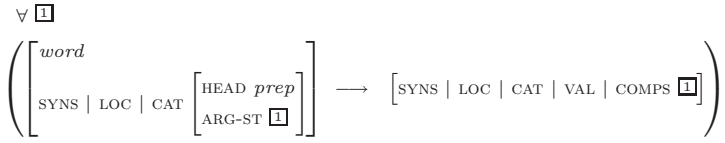


FIGURE 7 ARG-ST Mapping Lexical Principle for Prepositions (preliminary version)

That is, the ARG-ST value is assumed to be identical with the COMPS value. In order to facilitate prepositions to subcategorize nouns which are complement-unsaturated, and then select the complements of those nouns, the above principle has to be reformulated in the way shown in Figure 8. Here, the list of complements syntacticly selected by a preposition is a concatenation of its own ARG-ST list and the list of complements of its argument.[5]

[4]For the formalization of the language used in Pollard and Sag (1994) and for the formal definition of AVM syntax see Richter (2000).

[5]We assume, as Meurers (1997) does, that argument raising takes place only with respect to the valence attributes, and not with respect to the ARG-ST list.

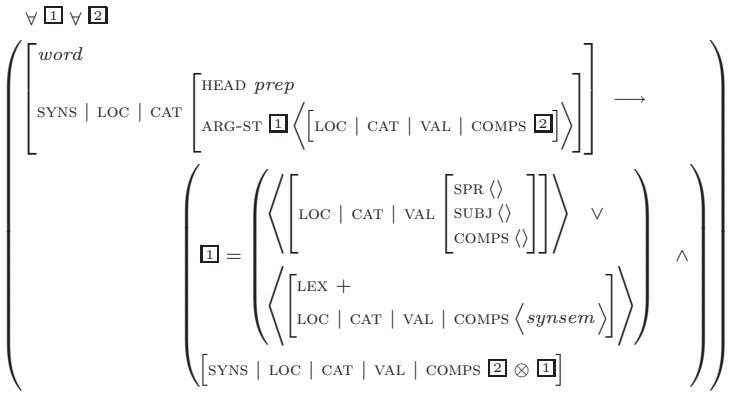


FIGURE 8 ARG-ST Mapping Lexical Principle for Prepositions

It has to be mentioned that the raising of more than one nominal complement result in ungrammatical constructions like those in (40).

(40) a. *in [Verbindung] [der Regierung] [mit diesem Problem] ...
in connection the government$_{GEN}$ with this problem

To avoid this problem, we have restricted the ARG-ST value of prepositions to the lists containing either one saturated element, or to the lists containing one element with a singleton COMPS list. Additionally, we have specified the LEX value of the second disjunct to be + with the idea of marking objects that have realized none of their complements. This restriction rules out the selection of relational nouns that have already realized one of their complements (cf. 41).

(41) a. *in [Verbindung der Regierung] [mit diesem Problem] ...
in connection the government$_{GEN}$ with this problem

The structure in Figure 9 exemplifies the interaction of the above assumptions in the licensing of a PP headed by a raising preposition.

Due to the ARG-ST Mapping Lexical Principle for Prepositions in Figure 8, the preposition *in*, which takes one nominal argument with one unrealized complement can be licensed. Thus, the syntactic and semantic properties of that complement are determined not by the preposition, but by the noun. Thereby, ungrammatical PPs such as those in (38) can be blocked. Both the noun and its unrealized complement are mapped to the COMPS list of *in* and, according to the constraints on the head-complement-structures for prepositions, they are syntactically selected by *in*.

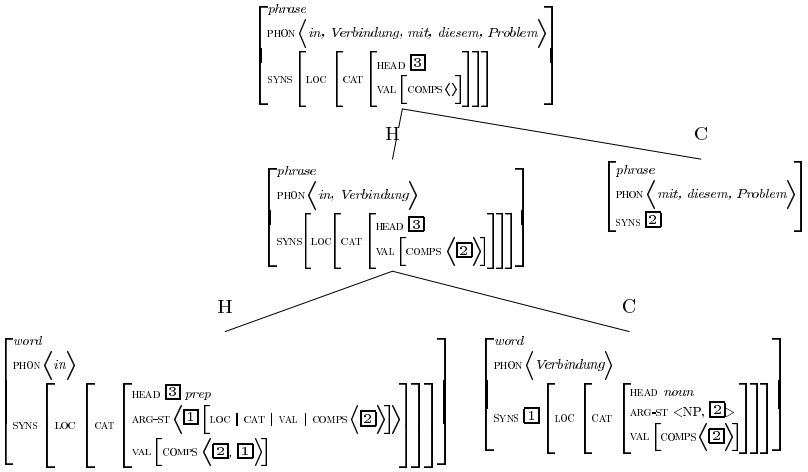


FIGURE 9 The structure of the PP *in Verbindung mit diesem Problem* ('in connection with this problem')

The first complement that *in* selects is the noun. By virtue of selectional requirements of restrictive adjectives as well as prepositions modifying nouns, that are specified as combining with complement-saturated nouns only, the modifying of complement-unsaturated nouns is blocked. The same restriction holds for determiners and quantifiers in German. These constraints, existing in the grammar independently of the principles of the CPs syntax, explain the apparent lexical fixedness of the P_1N_1 sequences (cf. (35a) and (35b)) without additional stipulations.

In the next and the last step the preposition *in* selects the complement of the noun as its own complement, forming a PP.

Exactly the same lexical entry for preposition *in* and the same set of principles license PPs headed by non-raising prepositions such as the PP *in einer engen Verbindung mit den Beratern* ('in close connection with the advisers').

10.5 A Cross-Linguistic Excursus

There is a strong evidence suggesting that the same technique can be applied for analyzing corresponding data in other languages. Quirk and Mulholland (1964) provide for instance a detailed description of sequences of the form $P_1N_2P_2NP$ in English, and isolate a class of expressions whose syntactic behavior corresponds to our observations of German data. According to Quirk and Mulholland (1964), N_1s within

English expressions such as *in spite of* or *by way of* cannot combine with determiners, cannot be premodified by adjectives, do not allow the P_2NP deletion, etc. Thus, they can probably be described in the same way as the corresponding German expressions.

Examples in (42) used in contexts such as those exemplified in (34) for German provide some evidence from Polish, that seems to substantiate our analysis as well. Here again, neither the selection of determiners by N_1s (cf. (42b)), nor the premodification (cf. (42c)) or the P_2NP deletion (cf. (42d)) are possible.

(42) a. z uwagi na ten problem
with regard to this problem
'with regard to this problem'

b. z *tej/*jego uwagi na ten problem
with the/ his regard to this problem

c. z *wielkiej uwagi na ten problem
with great regard to this problem

d. * z uwagi
with regard

Word combinations of the discussed type occurs in many other languages, thereby showing nearly uniform properties (cf. examples below quoted from Lindqvist 1994).

French: *en face de, en dépit de, au milieu de*
Spanish: *al lado de, en casa de*
Swedish: *i början av, med hjälp av, i stället för*

These parallels in the data strongly suggest that they can be described by the method presented in the previous section.

10.6 Summary

Here, the syntax of CPs in German have been examined. We have thereby seen that the previous approaches to this problem are highly problematic. We then proposed an analysis based on the raising mechanism assuming prepositions to be able to raise complements of their arguments. Underspecifying valence information within lexical entries of prepositions and applying appropriate lexical constraints, the presented theory offers a non-redundant description of linguistic facts about both the raising and non-raising prepositions.

The proposed analysis applies a technique which is already well es-

tablished in HPSG-based studies. Due to this technique, a treatment of different linguistic phenomena is possible that does not require any extensions of the existing description apparatus. We have shown, for instance, that there are parallels between the raising analysis proposed here for CPs and the raising analysis of German verbal complexes as proposed in Hinrichs and Nakazawa (1989). Possibly, a more precise investigation of these two empirical domains could result in generalizations that would contribute to formulating a consistent theory of raising within the HSPG grammar framework.

Acknowledgments

I would like to thank Gisbert Fanselow, Frank Richter, Manfred Sailer, and the anonymous reviewers of *FGVienna: The 8th Conference on Formal Grammar* in Vienna for their interesting comments on the issue presented in this paper and Carmella Payne for help with English.

References

Beneš, E. 1974. Präpositionswertige Präpositionalfügungen. In U. Engel and P. Grebe, eds., *Sprachsystem und Sprachgebrauch. Festschrift für Hugo Moser zum 65. Geburtstag. Teil I.*, Sprache der Gegenwart 33, pages 33–52. Düsseldorf: Schwann.

Buscha, J. 1984. Zur Syntax der Präpositionen. Deutsch als Fremdsprache 21.

De Kuthy, K. 2000. *Discontinuous NPs in German — A Case Study of the Interaction of Syntax, Semantics and Pragmatics*. Ph.D. thesis, Universität des Saarlandes, Saarbrücken.

Fries, N. 1988. *Präpositionen und Präpositionalphrasen im Deutschen und im Neugriechischen*. Linguistische Arbeiten 208. Tübingen: Max Niemeyer Verlag.

Hinrichs, E. and T. Nakazawa. 1989. Flipped Out: Aux in German. In *Papers from the 25th Regional Meeting of the Chicago Linguistics Society*, pages 193–202. Chicago.

Lindqvist, Ch. 1994. *Zur Entstehung der Präpositionen im Deutschen und Schwedischen*. Tübingen: Max Niemeyer Verlag.

Meibauer, J. 1995. Komplexe Präpositionen — Grammatikalisierung, Metapher, Implikatur und *Division of Pragmatic Labour*. In F. Liedtke, ed., *Implikaturen. Grammatische und pragmatische Analysen*, Linguistische Arbeiten 343, pages 67–74. Tübingen: Max Niemeyer Verlag.

Meurers, W. D. 1997. Using Lexical Principles in HPSG to Generalize over Valence Properties. In *Proceedings of the Conference on Formal Grammar*. Aix-en-Provence.

Meurers, W. D. 2000. *Lexical Generalizations in the Syntax of German Non-Finite Constructions*. Ph.D. thesis, Eberhard-Karls-Universität Tübingen. Published as *Arbeitspapiere des SFB 340*, Nr. 145.

Pollard, C. J. and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Quirk, R. and J. Mulholland. 1964. Complex Prepositions and Related Sequences. In *English studies presented to R. W. Zandvoord on the occasion of the 70th birthday, Supplement to Vol. 45*, pages 64–73. Amsterdam.

Richter, F. 2000. A Mathematical Formalism for Linguistic Theories with an Application in Head-Driven Phrase Structure Grammar. PhD thesis, Eberhard-Karls-Universität Tübingen, Version of April 28th, 2000.

Richter, F. and M. Sailer. 2002. Cranberry Words in Formal Grammar. In C. Beyssade, O. Bonami, P. Cabredo Hofherr, and F. Corblin, eds., *Empirical Issues in Formal Syntax and Semantics*, vol. 4. Presses Universitaires de Paris-Sorbonne, Paris.

Schröder, J. 1986. *Lexikon deutscher Präpositionen*. Leipzig: Verlag Enzyklopädie.

Winhart, H. 1997. Die Nominalphrase in einem HPSG-Fragment des Deutschen. In E. Hinrichs, D. Meurers, F. Richter, M. Sailer, and H. Winhart, eds., *Ein HPSG-Fragment des Deutschen, Teil 1: Theorie*, vol. 95 of *Arbeitspapiere des SFB 340*, pages 319–384. Universität Tübingen.

11

Regular Approximations through Labeled Bracketing

Anssi Yli-Jyrä

11.1 Introduction

Regular approximations for context-free languages have a wide use as language models in natural language and speech processing. Since the famous work by Pereira and Wright (1991), several approximation methods have become available (cf. Nederhof 2000). However, there are further approximation methods for *structured languages* whose strings indicate the constituent structure by means of brackets. Such methods may have a wide application area because of the recently renewed interest in structured languages in formal language theory (e.g. Kappes 1998), in XML document processing (e.g. Berstel and Boasson 2000), in finite-state methods in NLP (Roche 1996, Yli-Jyrä 2003a), and in dependency syntax (Yli-Jyrä 2003b).

A good regular approximation for context-free grammars should have at least the following properties: (i) It is well understood. (ii) It admits a compact representation that is so small that it can be stored into realistic computers (a compact representation consists of sub-automata that are combined lazily). (iii) It assigns bracketings with an accuracy that is practically sufficient for replacing the original parsing grammar. (iv) Its representation is easy to inspect and to modify (Nederhof 1997). (v) Parsing with the approximation does not lead to a combinatorial explosion from the compact representation.

The exactness up to any predefined center-embedding depth is an important property when we measure accuracy of approximations that

Proceedings of FGVienna: The 8th Conference on Formal Grammar.
Gerald Penn (ed.).
Copyright © 2008, CSLI Publications.

assign bracketings. Schematically, in a phrase $[\alpha\beta\gamma]$, the substring β is a *center-embedding*, if $\alpha \neq \lambda$ and $\gamma \neq \lambda$. The *center-embedding depth* measures how many phrases are nested as center-embeddings into each other. A *subset (superset) approximation* (Nederhof 2000) is said to be *exact up to any given center-embedding depth d* if it rejects (accepts) a string w only if w is rejected (accepted) by the original grammar or if the center-embedding depth of w is beyond the bound d.

A paradox in the earlier approximation methods is that they fail to combine a small compact representation with a sufficient exactness. For example, the approximation by Mohri and Nederhof (2001) admits a very compact representation, but it is a very coarse superset approximation for parsing purposes. The approximation by Johnson (1998) is exact up to any given depth of center-embedding, but the size of its compact representation grows very fast compared to the number of rules in the source grammar (Nederhof 2000: Figure 12: the methods LC2,LC3,LC4). Nevertheless, it would be interesting to find regular approximations that combine a small compact representation with a sufficient bound for exactness. Such grammars would be sufficient for context-free parsing under a reasonable performance restriction for the center-embedding.

This paper presents an approximation method that is based on a new *representation theorem* for context-free languages. According to it, any context-free language can be represented as a homomorphic image of an intersection of a set of *constraint languages* defining properties of valid labeled bracketings. The intersected languages of the new theorem differ from the ones used in the famous theorem by Chomsky and Schützenberger (1963). If these constraint languages are restricted to make them regular, we obtain a new kind of compact representation for regular approximations. The resulting approximation can be chosen to be either a subset or a superset of the original context-free language.

The subset and superset approximations obtained in this paper solve the paradox between the practical size of the representation and the sufficient bound for exactness. The approximations have properties (i) – (iv). As to property (v), we do not know yet how the current approximation behaves. The compact representations for the approximations constitute a subclass of *finite-state intersection grammars (FSIG)*. In general, FSIGs have a danger of combinatorial explosion during parsing (Tapanainen 1997), but it is tantalizing to see whether the currently presented subclass admits parsing algorithms that eliminate the problem.

The paper is structured as follows. In Section 11.2, we will introduce *context-free bracketing grammars (CFBGs)*. Section 11.3 discusses

a constraint-based approach to language specification. This approach gives rise to *flat CFBGs*, which are introduced in Section 11.4. Section 11.5 contains a representation theorem that connects flat CFBGs with CFBGs. In section 11.6, we will explain how various regular approximations can be obtained from a flat CFBG. We conclude in Section 11.7.

11.2 Generating Labeled Bracketings

11.2.1 Extended Context-Free Grammars

We have chosen extended context-free grammars (ECFGs) as the starting point, because this leads to wider applicability of the results. ECFGs were introduced by Thatcher (1967). It is well known that they generate exactly the context-free languages. Furthermore, it could be shown that for any standard context-free grammar there exists a strongly equivalent ECFG. Formally, ECFGs are defined as follows:

Definition 21 If E is a *regular expression* (Salomaa 1973), its *language* is denoted with $L(E)$. The empty string is λ. When L is a language, expression $L?$ denotes the language $L \cup \{\lambda\}$.

Definition 22 *Extended context-free grammar* is a quadruple $G = (V_N, V_T, P, S)$, where V_N is called *nonterminal alphabet* and V_T *terminal alphabet*, $S \in V_N$ is called the *start symbol* and P is a finite set of *production schemas* — one for each nonterminal; the alphabets V_N and V_T are finite and disjoint and their union $V = V_N \cup V_T$ is called the *total alphabet*; each production schema is of the form $X \rightarrow E_X$, where X is in V_N and E_X is a regular expression over the alphabet V.

Let w and w' be strings over V. We say that $w \overset{\Rightarrow}{G} w'$ or w' is *directly derivable from* w in grammar G if there are strings $u, v, x \in V^*$, a nonterminal $X \in V_N$ and a production schema $X \rightarrow E_X \in P$ such that $w = uXv$, $w' = uxv$ and $x \in L(E_X)$. The *language* $L(G)$ *generated by* G is $\{w \mid w \in V_T^*, S \overset{\Rightarrow}{G}{}^+ w\}$, where $\overset{\Rightarrow}{G}{}^+$ denotes the transitive closure of the relation $\overset{\Rightarrow}{G}$.

11.2.2 Context-Free Bracketing Grammars (CFBGs)

We want to encode constituent structures by means of labeled bracketing. For standard CFGs, this could be accomplished with CFGs that generate the *structured context-free languages* of Ritchie and Springsteel (1972). However, we define *context-free bracketing grammars* because this new class of grammars gives us more flexibility.

Definition 23 A *context-free bracketing grammar (CFBG)* is an ECFG $G = (V_N, V_T, P, S)$, where

- V_N contains three disjoint subsets:
  - N is the set of *basic nonterminals*,
  - $\overleftarrow{N} = \{\overleftarrow{X} | X \in N\}$ is the set of *left-branching nonterminals*,
  - $\overrightarrow{N} = \{\overrightarrow{X} | X \in N\}$ is the set of *right-branching nonterminals*,
- V_T is the set of terminal symbols containing five disjoint subsets:
  - $B_L = \{\, [_X \mid X \in N\}$ and $B_R = \{\,]_X \mid X \in N\}$ are, respectively, the set of *left square brackets* and the set of *right square brackets*,
  - $B_l = \{\, \langle_X \mid X \in N\}$ and $B_r = \{\, \rangle_X \mid X \in N\}$ are, respectively, the set of *left angle brackets* and the set of *right angle brackets*,
  - $\Sigma = V_T - (B_L \cup B_R \cup B_l \cup B_r)$
- P contains three disjoint sets of production schemata:

$$\{\overleftarrow{X} \to \quad E_{\overleftarrow{X}} \rangle_X \mid \overleftarrow{X} \in \overleftarrow{N} \text{ and } L(E_{\overleftarrow{X}}) \subseteq (\overleftarrow{N}?)(\Sigma \cup N)^* \quad \} \cup$$
$$\{X \to [_X\, E_X\,]_X \mid X \in N \text{ and } L(E_X) \subseteq (\overleftarrow{N}?)(\Sigma \cup N)^*(\overrightarrow{N}?)\} \cup$$
$$\{\overrightarrow{X} \to \langle_X E_{\overrightarrow{X}} \mid \overrightarrow{X} \in \overrightarrow{N} \text{ and } L(E_{\overrightarrow{X}}) \subseteq \quad (\Sigma \cup N)^*(\overrightarrow{N}?)\}.$$

CFBGs generate *context-free bracketing languages (CFBLs)*, which are properly included into the context-free languages.

Definition 24 Let $G_0 = (N, \Sigma, P_0, S)$ be an ECFG. Without loss of genericity we can assume that for every $X \in N$ there are new symbols $\overleftarrow{X}, \overrightarrow{X}, [_X,]_X, \langle_X$, and $\rangle_X$ such that they do not belong to $N \cup \Sigma$.

The *canonically obtained CFBG* $G = (V_N, V_T, P, S)$ is constructed as follows: The set of basic nonterminals N fixes, by Definition 23, $\overleftarrow{N}$, $\overrightarrow{N}$, B_L, B_R, B_l and B_r. Now $V_N = N \cup \overleftarrow{N} \cup \overrightarrow{N}$ and $V_T = \Sigma \cup B_L \cup B_R \cup B_l \cup B_r$. Let the set of production schemas P be $\{X \to [_X E_X]_X \mid X \to E_X \in P_0\} \cup \{X \to \langle_X \emptyset \mid X \in \overleftarrow{N}\} \cup \{X \to \langle_X \emptyset \mid X \in \overrightarrow{N}\}$.

Let h_1 be the homomorphism $h_1 : V_T^* \to \Sigma^*$ such that $h_1(a) = a$ for every $a \in \Sigma$, and $h_1(V_T - \Sigma) = \lambda$.

Theorem 11 *Every context-free language L' is a homomorphic image of a CFBL generated by a canonically obtained CFBG.*

Proof. Let G_0 be an ECFG generating L', and G the CFBG canonically obtained from grammar G_0. Obviously, $L' = h_1(L(G))$. □

Recognition of bracketed strings with a CFBG G is easy: the process reduces to validation of the constituent structure indicated by the bracketing. However, in many practical settings, the strings are not bracketed in advance. All the labeled bracketings for an unbracketed string w are obtained by computing $L(G) \cap h_1^{-1}(w)$. The resulting language is context-free, because $h_1^{-1}(w)$ is a regular language and the

context-free languages are closed under intersection with regular languages (Harrison 1978).

Historical note 1 Other kinds of grammars that generate structured languages have been defined in the literature. So-called *bracketed context-free grammars* (Ginsburg and Harrison 1967) differ from CFBGs in two ways: in them, (i) the rules are productions rather than production schemata (i.e. the right-hand side ω of each grammar rule $X \rightarrow \omega$ is a string rather than a regular expression), (ii) there may be several productions for each nonterminal; they are identified with different brackets. Other structured grammars include the *Chomsky-Schützenberger grammars* (cf. Berstel and Boasson 2000), *very simple grammars* (Korenjak and Hopcroft 1966), *parenthesis or parenthesized grammars* (McNaughton 1967, Salomaa 1973), *multi-parenthesis grammars* (Ginsburg et al. 1975), *generalized parenthesis grammars* (Takahashi 1975), *bracketing transduction grammars* (Wu 1995), *bracketed contextual grammars* (Kappes 1998) and *XML grammars* (Berstel and Boasson 2000).

11.2.3 CFBGs that Generate Reduced Bracketing

In canonically obtained CFBGs, the brackets occurring on both sides of each phrase force all the embedded phrases to constitute a center-embedding. These extra center-embeddings involve a disadvantage from the practical point of view, especially when the language is approximated through pushdown automata whose storage size is restricted.

Definition 25 In *full bracketing*, no bracket is shared by nested constituents. In *reduced bracketing*, a bracket symbol is never repeated, and two kinds of brackets are used; square brackets are balanced while the angle brackets aren't. The following two lines illustrate the two formats for bracketed strings:

$$\begin{array}{l} [[[[[a]b]c]d][e][[f]] \;\; g[h[i[[j[k[l[m]]]]n]]]] \\ [\quad a\rangle b\rangle c\rangle d\rangle [e][\langle f\;] \;\; g\langle h\langle i\langle [j\langle k\langle l\langle m\quad]n\quad] \end{array}$$

A repeated square bracket on the upper line corresponds to only a single square bracket on the lower line. To maintain the balanced square bracketing on the lower line, some of the remaining brackets have been changed to angle brackets. The important thing is that the format of the upper line is easily *restorable* from the lower line (using a deterministic linear-time algorithm). So, we can say that the two bracketings denote the same constituent structure.

Definition 26 Define rational transductions $s, t : V^* \to V^*$ as follows:

$$s(w) = \begin{cases} \overleftarrow{X}v, & \text{if } w = Xv \text{ for some } X \in N, v \in V^*; \\ w, & \text{otherwise.} \end{cases}$$

$$t(w) = \begin{cases} v\overrightarrow{X}, & \text{if } w = vX \text{ for some } X \in N, v \in V^*; \\ w, & \text{otherwise.} \end{cases}$$

Theorem 12 *For every canonically obtained CFBG $G = (V_N, V_T, P, S)$ there exists a CFBG $G' = (V_N, V_T, P', S)$ that uses reduced bracketing so that $L(G)$ is restorable from $L(G')$.*

Proof. For every production schema $X \to [_X E_X]_X \in P$, let $E'_{\overleftarrow{X}}$, E'_X and $E'_{\overrightarrow{X}}$ be regular expressions denoting, respectively, the languages $s(L(E_{\overleftarrow{X}}))$, $s(t(L(E_X)))$ and $t(L(E_{\overrightarrow{X}}))$. The set of production schemata P' is constructed as follows:

$$\begin{aligned} P' = \{\ \overleftarrow{X} \to \quad & E'_{\overleftarrow{X}}\rangle_X \mid X \in N\} \cup \\ \{\ X \to & [XE'_X]_X \mid X \in N\} \cup \\ \{\ \overrightarrow{X} \to & \langle_X E'_{\overrightarrow{X}} \quad \mid X \in N\} \end{aligned}$$

It is easy to see that the phrases that immediately contain square brackets disallow repeating square brackets in those immediate constituents that are in the outermost positions inside the local brackets; in the outermost positions the immediate constituents switch to "widow" angle brackets. Thus, the resulting bracketing is reduced.

Finally, it remains to be shown that the full bracketing is easily restorable in each string. For each angle bracket, the respective pair should be inserted next to the square bracket that embraces the angle-bracketed constituent. It is easy to see that this can be done for each bracketed string by means of a deterministic linear-time algorithm. □

Historical note 2 The historical roots of the reduced bracketing goes back to the ideas of Krauwer and des Tombe (1981) or even further (cf. Chomsky 1963: Section 4). According to Nederhof (2000), similar ideas have been put forward by Langendoen and Langsam (1987), Pullman (1986), Black (1989), Johnson (1996). The method by Johnson (1996) applies only to binary context-free productions, while our method is more general and applies to production schemas.

11.3 A Language Representation via Intersection

Our goal is to eliminate rewriting that is based on rewriting rules and to define a new representation for CFBLs using the intersection operation. One technique for representing context-free languages by means

of intersection is given by the Chomsky-Schützenberger Theorem.

Definition 27 The *Dyck language* (i.e. *semi-Dyck language* according to Harrison (1978)) D_n over the alphabet $L_n \cup R_n$, where $L_n = \{[_1, [_2, \ldots, [_n\}$ and $R_n = \{]_1,]_2, \ldots,]_n\}$, is the language generated by the context-free grammar $S \rightarrow \lambda \mid SS \mid [_1 S]_1 \mid [_2 S]_2 \mid \ldots \mid [_n S]_n$.

Theorem 13 (Chomsky and Schützenberger 1963) *Every context-free language L is a homomorphic image of the intersection of a Dyck language D and a regular language R.*

If a grammar G is in the Chomsky normal form, it is easy to construct an equivalent representation $h_2(D \cap R)$. The homomorphism h_2 substitutes brackets either with λ or with the letters of L. The strings of $D \cap R$ are bracketed so that each bracket indicates either a terminal symbol or a production. Intuitively, the regular language R takes care of local properties. The Dyck language D, in turn, takes care of non-local properties. A detailed proof can be found e.g. in Salomaa (1973).

The Chomsky-Schützenberger representation eliminates the need for language specific rewriting rules. It describes all the properties of the context-free language using constraints on strings. However, in order to implement parsing of CFBG in this way, we have to use two morphisms, h_1 and h_2, which results in an unnecessarily complex formula: $h_2(D \cap R \cap h_2^{-1}(h_1^{-1}(w)))$

Definition 28 Given an alphabet V and languages $C, Lc, Rc \subseteq V^*$, the *context restriction constraint* $C \Rightarrow Lc \_ Rc$ (Koskenniemi 1983) denotes the language $V^* - ((V^* - V^*Lc)CV^* \cup V^*C(V^* - RcV^*))$.

Let the homomorphism $h_3 : (L_n \cup R_n)^* \rightarrow \{[_1,]_1)^*$ be defined in such a way that $h_3(L_n) = \{[_1\}$ and $h_3(R_n) = \{]_1\}$. Denote the inverse homomorphic image $h_3^{-1}(D_1)$ with $\widehat{D}$.

Theorem 14 (Wrathall 1977) *Any Dyck language D_n equals to an intersection of $\widehat{D}$ and n context restriction constraints $]_i \Rightarrow [_i \widehat{D} \_ \lambda$, $1 \leq i \leq n$.*

Wrathall's theorem demonstrates that bracket labels can be matched using distributed constraints instead of D_n. In the following section, we use other kinds of distributed constraints to achieve the same effect.

11.4 Flat CF Bracketing Grammars (FCFBGs)

The Chomsky-Schützenberger theorem involves a hidden structured language. There is, however, a direct way to represent CFBL by means of constraint languages.

Definition 29 Let the homomorphism $h_4 : V_T^* \to \{[_1,]_1\}^*$ be defined in such a way that $h_4(B_L) = \{[_1\}$, $h_4(B_R) = \{]_1\}$, and $h_4(\Sigma \cup B_l \cup B_r) = \{\lambda\}$. The language $h_4^{-1}(D_1)$ is denoted with D'. Substitution f_1 is a mapping from $(V \cup \{\Delta\})^*$ to subsets of V^* defined so that it replaces the special symbol Δ with the language D'.

Given an alphabet V and regular expressions Lc, Rc, C over $V \cup \{\Delta\}$, the *bracketing restriction constraint* $\#Lc \_ Rc\# \Rightarrow C$ denotes the language $L(\#Lc \_ Rc\# \Rightarrow C) = \{v \mid v \in D' \wedge \forall x(x \in D' \wedge v \in f_1(L(Lc)\, x\, L(Rc)) \implies x \in f_1(L(C)))\ \}$. When $L(Lc) = V^*Lb$ and $L(Rc) = RbV^*$, such that $Lb \subseteq (B_L \cup B_l)$ and $Rb \subseteq (B_R \cup B_r)$, the constraint $\#Lc \_ Rc\# \Rightarrow C$ can be written more conveniently as $Lb \_ Rb \Rightarrow C$.

A flat CFBG is a triple $G = (N, V_T, K)$, where N is a finite set of *bracket labels*, V_T is a finite alphabet, and K is a finite set of *bracketing restriction constraints*. Alphabet V_T contains five disjoint subsets: $B_L = \{[_X | X \in N\}$, $B_R = \{]_X | X \in N\}$, $B_l = \{\langle_X | X \in N\}$, $B_r = \{\rangle_X | X \in N\}$, and $\Sigma = V_T - (B_L \cup B_R \cup B_l \cup B_r)$. The language $L(G) \subseteq V_T^*$ described by G is $\cap_{K_i \in K} L(K_i)$.

In the following, we assume that alphabets V_N, V_T, N, $\overleftarrow{N}$, $\overrightarrow{N}$, B_L, B_R, B_l, B_r, Σ are defined by the context.

Definition 30 Substitution f_2 is a mapping from V to $(V \cup \{\Delta\})^*$ defined as follows:

$$f_2(X) = \begin{cases} \Delta\rangle_X & \text{if } X \in \overleftarrow{N}; \\ [_X\, \Delta]_X & \text{if } X \in N; \\ \langle_X \Delta & \text{if } X \in \overrightarrow{N}; \\ X & \text{otherwise.} \end{cases}$$

The mapping f_2 is extended to regular expressions as follows: $f_2(\lambda) = \lambda$; $f_2(\emptyset) = \emptyset$; $f_2(xy) = f_2(x)f_2(y)$; $f_2(x^*) = f_2(x)^*$; $f_2(x|y) = f_2(x)|f_2(y)$.

A *canonically obtained flat CFBG* $G' = (N, V_T, K)$ is associated with every CFBG $G = (V_N, V_T, P, S)$ as follows: N is the basic nonterminal alphabet of G. Let

$$\begin{aligned} K = \{ \ \# \_ \# \ &\Rightarrow f_2(S)\ \} && \cup \\ \{ \ [_X \_]_X \ &\Rightarrow f_2(E_X) \ \mid X \in N, X \to [_X E_X]_X \ \in \ P\ \} && \cup \\ \{ \ B_L \_ \rangle_X \ &\Rightarrow f_2(E_{\overleftarrow{X}}) \mid X \in N, \overleftarrow{X} \to \ E_{\overleftarrow{X}}\rangle_X \ \in \ P\ \} && \cup \\ \{ \ \langle_X \_ B_R \Rightarrow f_2(E_{\overrightarrow{X}}) \mid X \in N, \overrightarrow{X} &\to \langle_X E_{\overrightarrow{X}} \ \in \ P\ \} \end{aligned}$$

Example. Let G_1 be a CFBG with the start symbol S and the following set of production schemata:

$$\begin{array}{lcl} S & \rightarrow & [_S\ NP\ \ VP\ ]_S \\ VP & \rightarrow & [_{VP}\ V\ ]_{VP} \mid [_{VP}\ V\ \ NP\ ]_{VP} \\ NP & \rightarrow & [_{NP}\ \text{Jim}\ ]_{NP} \mid [_{NP}\ \text{Sue}\ ]_{NP} \\ V & \rightarrow & [_V\ \text{ran}\ ]_V \end{array}$$

These production schemata and the start symbol S give rise to the following set of bracketing restriction constraints:

$$\begin{array}{lcl} \#\ \_\ \# & \Rightarrow & [_S\Delta]_S \\ [_S\ \_\ ]_S & \Rightarrow & [_{NP}\Delta]_{NP}[_{VP}\Delta]_{VP} \\ [_{VP}\ \_\ ]_{VP} & \Rightarrow & [_V\Delta]_V \ \mid\ [_V\Delta]_V[_{NP}\Delta]_{NP} \\ [_{NP}\ \_\ ]_{NP} & \Rightarrow & \text{Jim} \mid \text{Sue} \\ [_V\ \_\ ]_V & \Rightarrow & \text{ran} \end{array}$$

11.5 Representing CFLs with FCFBGs

Lemma 15 *The languages D' and $L(\#\ \_\ \# \Rightarrow f_2(S))$ are generated by ECFGs. Furthermore, for each bracket label $X \in N$, the language $L([_X\ \_\ ]_X \Rightarrow f_2(E_X))$ is generated by an ECFG.*

Proof. Let $E_{\neq} = \cup_{X,Y \in N, X \neq Y}[_X\Delta\ ]_Y$. D' is generated by an ECFG $G_{D'} = (\{\Delta\}, V_T, P_{D'}, \Delta)$, where $P_{D'} = \{\Delta \rightarrow \lambda|\Delta\Delta|\Sigma|B_l|B_r|B_L\Delta B_R\}$. $L(\#\ \_\ \# \Rightarrow f_2(S))$ is generated by an ECFG $G_{\#} = (\{S, \Delta\}, V_T, P_{\#}, S)$, where $P_{\#} = \{\ S \rightarrow [_S\Delta]_S,\ \Delta \rightarrow \lambda|\Delta\Delta|\Sigma|B_l|B_r|B_L\Delta B_R\ \}$. $L([_X\ \_\ ]_X \Rightarrow f_2(E_X))$ is generated by an ECFG $G_X = (N \cup \{\Delta\}, V_T, P_X, \Delta)$, where $P_X = \{\Delta \rightarrow \lambda|\Delta\Delta|\Sigma|B_l|B_r|E_{\neq}|N\} \cup \{X \rightarrow [_X E_X]_X\} \cup \{Y \rightarrow [_Y\Delta]_Y \mid Y \in N, Y \neq X\}$. □

Lemma 16 *Let $G' = (N, V_T, K)$ a flat CFBG canonically obtained from a CFBG $G = (V_N, V_T, P, S)$ that is canonically obtained from an ECFG. $L(G') = L(G)$.*

Proof. We first show that the language $L(G')$ is generated by an ECFG. By Lemma 15, each constraint $L([_X\ \_\ ]_X \Rightarrow f_2(E_X))$ is generated by an ECFG G_X. The intersection $\cap_{X \in N} L(G_X)$ is generated by an ECFG $G_N = (N \cup \{\Delta\}, V_T, P_N, \Delta)$, where $P_N = \{\Delta \rightarrow \lambda|\Delta\Delta|\Sigma|B_l|B_r|E_{\neq}|N\} \cup \{X \rightarrow [_X E_X]_X \mid X \in N\}$. The intersection $L(G')$ of $L(G_N)$ and $L(G_{\#})$ is generated by a CFBG $G'' = (V_N, V_T, P', S)$, where $P' = \{X \rightarrow [_X E_X]_X \mid X \in N\} \cup \{X \rightarrow \emptyset\rangle_X \mid X \in \overleftarrow{N}\} \cup \{X \rightarrow \langle_X\emptyset \mid X \in \overrightarrow{N}\}$. Because $G'' = G$, it holds that $L(G') = L(G)$. □

Theorem 17 *Every context-free language L' is a homomorphic image of an intersection of bracketing restriction constraints.*

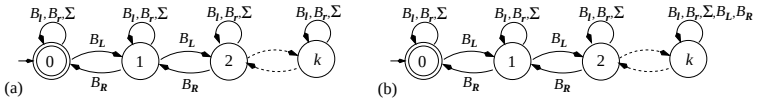


FIGURE 1 A schematic representation of automata for A_k and A'_k.

Proof. Let G_0 be an ECFG generating L' and G the CFBG canonically obtained from G_0. By Theorem 11, L' is a homomorphic image of a $L(G)$. A flat CFBG G' canonically obtained from G is equivalent to G by Theorem 16. So, the language $L(G)$ equals to an intersection of bracketing restriction constraints. □

11.6 Obtaining Approximations from FCFBGs

FCFBGs have the advantage that they allow us to define useful regular approximations simply by altering the language denoted by Δ: if a regular language is substituted for Δ, then the whole grammar becomes regular.

We define approximations for D' by restricting the number of nested brackets. The series of subset approximations $A_0, A_1, A_2, \ldots$ is defined inductively as follows: $A_0 = (\Sigma \cup B_l \cup B_r)^*$, $A_i = A_{i-1}(B_L A_{i-1} B_R A_{i-1})^*$, for $i = 1, 2, \ldots$. A schema of the finite automaton that accepts A_k is shown in Figure 1a. In NLP, a plausible setting for k is probably near to 5 (Johnson 1998). It is also possible to define a series of superset approximations $A'_0, A'_1, A'_2, \ldots$; the schema of the automaton that accepts A'_k is shown in Figure 1b.

It is possible to analyze the descriptive complexity of the resulting approximations. *Generalized regular expressions* are regular expressions with the complement operation. Languages like $A_0, A_1, A_2, \ldots$ are captured with *generalized regular expressions without Kleene's star* (Brzozowski and Knast 1978, Yli-Jyrä 2003a). In addition to this, the bracketing restriction constraints can be written with star-free generalized regular expressions. Thus, the whole approximation reduces to a star-free regular language. Our conjecture is that the *dot-depth* of this language is decidable, although, in general, it is an open problem to decide whether a regular language has the dot-depth n (Pin 2003).

The bracketing restriction constraints of a FCFBG correspond to generalized regular expressions in the approximation. Let $\widetilde{D}$ be the approximated language used for D'. Substitution f_3 is a mapping from $(V_T \cup \{\Delta\})^*$ to subsets of V_T^* defined so that it replaces the special symbol Δ with the language $\widetilde{D}$. Mapping f_3 is extended to regular expressions in the same manner as f_2 (Page 160). The subset approximation of the bracketing restriction constraint $\#Lc \_ Rc \Rightarrow \#f_2(E)$

can be compiled using the following formula:

$$\widetilde{D} - Lc(\widetilde{D} - f_3(f_2(E)))Rc$$

This compilation formula can be varied in many ways. When $\widetilde{D} = A_k$, the constraint $Lb \_ Rb \Rightarrow f_2(E)$, where $L(Lb) \subseteq B_L$, and $L(Rb) \subseteq B_R$, is equivalent to $A_k - V_T^* Lc(A_{k-1} - f_3(f_2(E)))RcV_T^*$. Furthermore, if we intersect the language of each bracketing restriction constraint with context restriction constraints $Lb \Rightarrow \lambda \_ A_k Rb$ and $Rb \Rightarrow LbA_k \_ \lambda$ over the alphabet V_T, we get smaller constraint languages and smaller finite automata. This does not affect the language of the whole grammar. In addition to this, equivalent but substantially more efficient ways to compile bracketing restriction constraints exist but they are not reported here.

For any regular language L', its *state complexity* is the size of the minimal deterministic finite automaton that recognizes L'. On the basis of our initial experiments, we have reached to the following generalization for the state complexity of bracketing restriction constraints: If a bracketing restriction constraint is of the form $[_X \_]_X \Rightarrow f_2(E)$, $L(E) \subseteq N^*$, m is the state complexity of $L(f_2(E))$ and t is the number of transitions in the minimal automaton accepting $L(E)$, then the state complexity $s(k)$ of the approximated constraint language

$$L(A_k - V_T^* \, [_X \, (A_k - f_3(f_2(E)) \,) \,]_X \, V_T^*) \cap$$
$$(\, [_X \Rightarrow \lambda \_ A_k]_X \,) \, \cap \, (\,]_X \Rightarrow [_X A_k \_ \lambda \,)$$

is estimated inductively as follows: $s(0) = 1$, $s(1) = 2$, and $s(k)$ in $O(s(k-1) + ts(k-2) + m)$. For example, if $E = \{YZ\}$ and $Y, Z \neq X$, we obtain $m = 7$, $t = 2$, $s(2) = 8$, $s(3) = 16$, $s(4) = 36$, $s(5) = 72$ etc. We can also obtain a state complexity that is quadratic in k, if we split each constraint into k sub-constraints so that each sub-constraint checks one nesting level only; each sub-constraint will have a linear state complexity according to k.

Certain computationally cheap changes in the formula improve the accuracy of the resulting constraint languages. If we use the formula $V_T^* - Lc(A_k - f_3(f_2(E)))Rc$, the constraint languages become larger, but this does not affect the language of the whole grammar. The accuracy of the whole grammar can be improved if the constraint languages are compiled so that they have different depth bounds for nested self-embeddings and nested center-embeddings. A *self-embedding* is a center-embedding, where a phrase is center-embedded into a phrase of the same category.

The standard compact representations are based on lazy substitution of transitions with sub-automata (Mohri and Pereira 1998, Neder-

hof 2000). Their primary purpose is to improve efficiency of off-line construction of finite automata. However, the compact representation may also be used on-line for processing of input (Mohri and Nederhof 2001). Our compact representation is used on-line, but it is different: the sub-automata accepting the individual constraint languages are *intersected* rather than substituted. Parsing in our representation is based on satisfiability of all the finite-state constraints at the same time. The approach is open to extensions where the domain of constraints is not necessarily restricted to local phrase structures.

When the approximation is derived from a FCFBG that uses reduced bracketing, we gain an essential advantage over regular approximations that assign full bracketing to the strings. For example, a small-scale regular approximation (available upon request) analyzed PP-attachment more efficiently and accurately using reduced bracketing.

11.7 Conclusion

We have suggested a new constraint-based representation for context-free sets of bracketed strings. The new representation lends itself for a direct construction of some regular approximations.

The obtained regular approximations have a number of favorable properties: (i) they admit a small compact representation, (ii) they handle tail-recursion appropriately and they are exact up to a predefined depth of center-embedding, and (iii) they can be represented by means of simple regular expressions.

The compact representation has been implemented and tested successfully in practice. Parsing with full-scale grammars *may* require special intersection algorithms to avoid combinatorial explosion.

Acknowledgments

The author is indebted to the anonymous referees and the FGVienna audience for comments and questions that helped to improve the quality of this paper. The research has been funded by the Nordic Academy of Advanced Study (NorFA).

References

Berstel, J. and L. Boasson. 2000. XML grammars. In *Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science, LNCS 1983*, pages 182–191. Springer.

Black, A. W. 1989. Finite state machines from feature grammars. In M. Tomita, ed., *Proceedings of the First International Workshop on Parsing Technologies*, pages 277–285. Pittsburgh: Carnegie Mellon University Press.

Brzozowski, J. A. and R. Knast. 1978. The dot-depth hierarchy of star-free languages is infinite. *Journal of Computer and System Sciences* 16:37–55.

Chomsky, N. 1963. Formal properties of grammars. In R. Luce, R. R. Bush, and E. Galanter, eds., *Handbook of Mathematical Psychology*, vol. II, pages 323–428. New York: John Wiley and Sons.

Chomsky, N. and M.-P. Schützenberger. 1963. The algebraic theory of context-free languages. In P. Brafford and D. Hirschberg, eds., *Computer Programming and Formal Systems*, pages 118–161. Amsterdam: North-Holland.

Ginsburg, S., J. Goldstine, and S. A. Greibach. 1975. Uniformly erasable AFL. *Journal of Computer and System Sciences* 10:237–247.

Ginsburg, S. and M. A. Harrison. 1967. Bracketed context-free languages. *Journal of Computer and System Sciences* 1(1):1–23.

Harrison, M. A. 1978. *Introduction to Formal Language Theory*. Reading, MA: Addison-Wesley.

Johnson, M. 1996. Left corner transforms and finite state approximation. DRAFT of 12th May, 1996. Rank Xerox Research Centre.

Johnson, M. 1998. Finite-state approximation of constraint-based grammars using left-corner grammar transforms. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, vol. 1, pages 619–623. Montréal.

Kappes, M. 1998. On the generative capacity of bracketed contextual grammars. *Grammars* 1(2):91–101.

Korenjak, A. J. and J. E. Hopcroft. 1966. Simple deterministic languages. In *Proceedings of the 7th Annual IEEE Symposium on Switching and Automata Theory*, pages 36–46.

Koskenniemi, K. 1983. Two-level morphology: a general computational model for word-form recognition and production. Publications of the Department of General Linguistics 11, University of Helsinki, Finland.

Krauwer, S. and L. des Tombe. 1981. Transducers and grammars as theories of language. *Theoretical Linguistics* 8:173–202.

Langendoen, D. T. and Y. Langsam. 1987. On the design of finite transducers for parsing phrase-structure languages. In A. Manaster-Ramer, ed., *Mathematics of Language*, pages 191–235. Amsterdam: John Benjamins.

McNaughton, R. 1967. Parenthesis grammars. *Journal of the Association for Computing Machinery* 14:490–500.

Mohri, M. and M.-J. Nederhof. 2001. Regular approximation of context-free grammars through transformation. In J.-C. Junqua and G. van Noord, eds., *Robustness in Language and Speech Technology*, chap. 9, pages 153–163. Kluwer Academic Publishers.

Mohri, M. and F. C. N. Pereira. 1998. Dynamic compilation of weighted context-free grammars. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, vol. 2, pages 891–897. Montréal.

Nederhof, M.-J. 1997. Regular approximation of CFLs: a grammatical view. In H. Bunt and A. Nijholt, eds., *Proceedings of the 5th International Workshop on Parsing Technologies*, pages 159–170. Massachusetts Institute of Technology.

Nederhof, M.-J. 2000. Practical experiments with regular approximation of context-free languages. *Computational Linguistics* 26(1):17–44.

Pereira, F. C. N. and R. N. Wright. 1991. Finite-state approximation of phrase-structure grammars. In *The Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 246–255. Berkeley.

Pin, J.-E. 2003. Algebraic tools for the concatenation product. *Theoretical Computer Science* 292(1):317–342.

Pullman, S. G. 1986. Grammars, parsers, and memory limitations. *Language and Cognitive Processes* 1(3):197–225.

Ritchie, R. W. and F. N. Springsteel. 1972. Language recognition by marking automata. *Information and Control* 20:313–330.

Roche, E. 1996. Parsing with finite-state transducers. Technical Report TR-96-30, MERL - A Mitsubishi Electronic Research Laboratory.

Salomaa, A. 1973. *Formal Languages*. Academic Press: New York.

Takahashi, M. 1975. Generalizations of regular sets and their application to a study of context-free languages. *Information and Control* 27:1–35.

Tapanainen, P. 1997. Applying a finite-state intersection grammar. In E. Roche and Y. Schabes, eds., *Finite-state language processing*, pages 311–327. Cambridge: MIT Press.

Thatcher, J. W. 1967. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences* 1(?):317–322.

Wrathall, C. 1977. Characterizations of the Dyck sets. *R.A.I.R.O. Informatique théorique/Theoretical Computer Science* 11(1):53–62.

Wu, D. 1995. An algorithm for simultaneously bracketing parallel texts by aligning words. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 244–251.

Yli-Jyrä, A. 2003a. Describing syntax with star-free regular expressions. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 379–386. Budapest.

Yli-Jyrä, A. 2003b. Multiplanarity – a model for dependency structures in treebanks. In *2nd Workshop on Treebanks and Linguistic Theories*.

Index

approximations
  regular, 153
  subset, 154, 163
  superset, 154, 163
argument raising, 128, 130

Becker, T., 100
binding
  Principle C, 96
bracketing
  full, 157
  labeled, 155
  reduced, 157, 164
bracketing grammars, 154, 155
  canonically obtained, 156
  canonically obtained flat, 160
  flat, 155, 159
brackets of a CFBG
  angle, 156, 157
  square, 156, 157

categorial grammar, 98
  classical, 83
center-embedding depth, 154
Chomsky, N., 95
COLL, *see* collocation
Collins, C., 103
collocation, 127, 131
combinatorial explosion, in
    parsing, 153, 154, 164
constraints
  bracketing restriction, 160
  constraint languages, 154
  context restriction, 159
Construction Grammar, 126
CONTENT, 130, 134
context-free grammars
  bracketed, 157
  bracketing grammars, 154–156,
      159, 160
  extended, 155
context-free languages,
    structured, 155
corpus data, 133
countercyclic(ity)
  adjunction, 96
  merge, 103
  move, 103
cranberry word, 131–133
CW, *see* cranberry word

Dobrovol'skij, D., 125
Dyck language, 159

Erbach, G., 130

feature forest, 15
Fillmore, C., 126
finite-state intersection
    grammars, 154
Fleischer, W., 125
Frey, W., 95

generative capacity
  of MGs, derivational, strong,
      weak, 96, 100

strong, 20
generative-enumerative theory, 112
German, 124
grammar extraction, 20
graph, 115
expressions, algebra of, 117
operations, 116
graphs
context-free set of, 117
recognisable, 118
grid, 119

Harkema, H., 101

idiom, 126

Janssen, T. M. V., 102
Joshi, A. K., 98

Keenan, E. L., 100
Krenn, B., 130

late adjunction, 96
Lebeaux, D., 96
LEXEME, 129–131
licensing theory, 112
linear context-free rewriting system, 17

maximum-entropy model, 13, 14
Meurers, W. D., 129, 130
MGs, *see* minimalist grammar
mildly context–sensitive grammars, 95
minimalist grammar, 95, 112
with generalized adjunction, 98
MOD, 128
model-theoretic syntax, 112
monadic second-order logic, 118

negation, 134

phrasal lexical entry, 125
polarity item, 131

Rackowski, A., 103
range-concatenation grammar, 24
regular expressions, 155
generalized, 162
star-free generalized, 162
regular languages
compact representations of, 153, 163
dot-depth of, 162
star-free, 162
state complexity of, 163
Relational Speciate Re-entrant Language, 135
relative clause(s)
Bach-Cooper, 102
extraposition of, 102
restrictive, 102
representation theorem
Chomsky-Schützenberger, 154, 159
for Dyck languages, 159
with bracketing restriction, 154, 161
Richter, F., 134
Riehemann, S. Z., 126, 127, 134
root sign, 134
RSRL, *see* Relational Speciate Re-entrant Language

satisfiability, 164
Schabes, Y., 98, 101
scrambling, rightward, 102
selection, 126
external, 128–130
of lexemes, 129–131
self-embedding, 163
Shieber, S. M., 101
SPEC, 128
Stabler, E. P., 95, 100
statistical parsing, 13
structural examples, 81
structured languages, 153
subcategorization, 126

Travis, L., 103
tree-adjoining grammar, 19, 21, 98, 112
treewidth, 117

types
  canonical, 84
  semantic, 82

UNC, *see* unique nominal complement
unique nominal complement, 123–126

verbal complex, *see* argument raising
Vijay-Shanker, K., 101

XSEL, 128, 134