Zipper-driven Parsing for LFG Grammars

Ronald M. Kaplan
Stanford University

Jürgen Wedekind
University of Copenhagen

Proceedings of the LFG'20 Conference

On-Line

Miriam Butt, Ida Toivonen (Editors)

2020

CSLI Publications

pages 169–189

`http://csli-publications.stanford.edu/LFG/2020`

Keywords: parsing, mildly context-sensitive dependencies, linear context-free rewriting system, LCFRS, finite copying grammar, finitely bounded grammar, zipper unification, tractability, nonconstructivity, XLE

Kaplan, Ronald M., & Wedekind, Jürgen. 2020. Zipper-driven Parsing for LFG Grammars. In Butt, Miriam, & Toivonen, Ida (Eds.), *Proceedings of the LFG'20 Conference, On-Line*, 169–189. Stanford, CA: CSLI Publications.

Abstract

We describe an approach to LFG parsing that is optimized for c-structure discontinuities that are established through "zipper" unification. These are characterized by parallel c-structure paths that carry the same function assignments. Wedekind and Kaplan (2020) demonstrated that LFG grammars giving rise to discontinuities with finitely bounded zipper paths can express only mildly context-sensitive dependencies and thus can be converted to equivalent linear context-free rewriting systems (LCFRSs). In principle, parsing with LCFRS grammars can be accomplished in polynomial time, but that may not be the most effective way of parsing with mildly context-sensitive dependencies. In this paper we propose a hybrid strategy for LFG parsing that is tuned to the common case of bounded zippers but still allows for putatively rare constructions that do not conform to the formal restrictions that guarantee finite boundedness. This strategy automatically takes advantage of mildly context-sensitive dependencies in addition to the context-free dependencies that the XLE parsing system has focused on (Maxwell and Kaplan 1996).

1 Introduction

The prohibition against c-structures with nonbranching dominance (NBD) chains ensures the decidability of the recognition/parsing problem for LFG grammars (Kaplan and Bresnan 1982) but still that problem is known to be NP-complete (Berwick 1982, Trautwein 1995) and thus intractable in the worst case. However, grammars for actual languages seem not to exploit all the mathematical power that the LFG formalism makes available, as witnessed by the fact that parsing and generation systems, for example, the XLE system, have been constructed that are practical for broad coverage grammars and naturally occurring sentences (Crouch et al. 2008, Maxwell and Kaplan 1996).

The implementations of these systems must be taking advantage implicitly of certain patterns of dependencies that are characteristic of linguistic grammars even if those properties have not been clearly articulated and explicitly coded. XLE in particular is optimized for context-free structures in sentences for which disjunctions arising from words and phrases that are distant from each other in the string are not incompatible. This optimization is based on the disjunctive constraint and lazy contexted constraint satisfaction algorithms developed by Maxwell and Kaplan (1991, 1996) (henceforth the MK algorithms). The XLE experience has shown these algorithms to be effective for a large majority of sentences in many languages, even though performance may—and does—degrade for constructions with dependencies that are more sensitive to context.

Recent papers (Wedekind and Kaplan 2020, Kaplan and Wedekind 2019) have identified a class of dependencies that are more sensitive to context

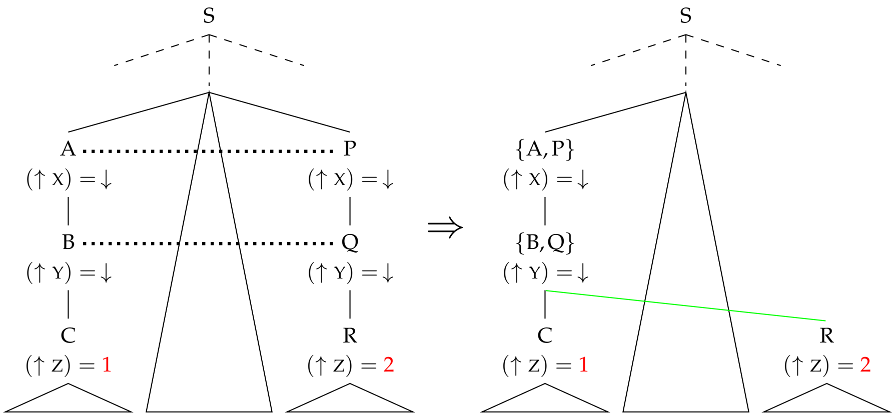


Figure 1: Zipping of discontinuous constituents.

but may still allow for efficient processing of a broader range of commonly occurring sentences. These dependencies allow information carried by distant c-structure nodes of discontinuous constituents to interact as long as those nodes map to f-structure through c-structure paths that are annotated with the same function assignments. This pattern has been described as "zipper" unification (Maxwell and Kaplan 1996) and is illustrated in Figure 1. The paths from the common mother of A and P to B and Q in the left tree are labeled in parallel by the $(\uparrow \text{X}) = \downarrow$ and $(\uparrow \text{Y}) = \downarrow$ function assignments and they can therefore be zipped into the structure on the right. This represents how information from the separate paths can be systematically combined. In particular it reveals the inconsistency between the $(\uparrow \text{Z}) = 1$ and $(\uparrow \text{Z}) = 2$ annotations even though the B and Q nodes dominate substrings that are not adjacent.

The key formal property of two nodes n and n' that zip together is that they map to the same f-structure (i.e., $\phi(n) = \phi(n')$) or, equivalently, that n and n' both belong to $\phi^{-1}(f)$ for some unit of f-structure f. If it can be established for an LFG grammar G that discontinuities are exclusively captured through zipper unification and the size of $\phi^{-1}(f)$ is bounded by a constant for all f-structure units for all sentences, then that grammar can be converted to a grammar G' in the formalism of linear context-free rewriting systems (LCFRSs) (Seki et al. 1991, Kallmeyer 2010), a grammatical formalism that can encode mildly context-sensitive dependencies. The LCFRS grammar G' is equivalent to G in that it produces the same f-structures for the same sentences (Wedekind and Kaplan 2020). Wedekind and Kaplan describe notational and derivational restrictions, here summarized in Section 2, that G must meet in order to determine whether G is convertible. They observe that grammars in this subclass, the finitely bounded LFG grammars, are still likely suitable for natural language description (see also Kaplan and Wedekind 2019).

The LCFRS conversion identifies and combines the information from all possible zippers, precomputing and eliminating from G' the effect of any combinations that would give rise to unsatisfiable f-descriptions. On its face, the advantage is that LCFRS parsing algorithms applied to G' will simulate the recognition of all and only the c-structures whose f-descriptions are guaranteed to be satisfiable. This crucially differs from the two-stage process of typical LFG parsing algorithms, including XLE, of context-free chart parsing that produces a representation of many candidate c-structures whose f-descriptions are then checked for satisfiability. The two-stage process is exponential and intractable in the worst case, because of the many candidate c-structure constituents that must be evaluated, while one-stage LCFRS parsing is known to take time that is polynomial in the length of the input string.

Realizing the advantages of direct LCFRS parsing for a given finitely-bounded G depends on the feasibility of carrying out the conversion and also on the size of the resulting G'. The conversion process for an arbitrary G may be too expensive and the equivalent LCFRS grammar too large for practical use. However, following Wedekind and Kaplan (2020) we point out in Section 3 that the grammar expansion is likely to be limited for LFG grammars describing natural languages and it may be feasible to apply direct LCFRS parsing to such languages. But that may not be the most effective way of exploiting the zipper configurations implicit in G's derivations.

Thus in Section 4 we consider a strategy that applies not to the given grammar G but to a specialization of G containing only the annotated rules that define the f-structures for a particular input string. The specialized grammar is likely to be finitely bounded even if the entire G is not, and the LCFRS for the specialized grammar is likely to be much smaller and to operate more efficiently than the LCFRS for the larger grammar. In Section 5 we sketch an alternative strategy for propagating zipper information that works even if the specialized grammar lies outside the finitely bounded class. This involves identifying and eliminating the zipper-entailed inconsistencies of the specialized grammar and then using MK bottom-up satisfiability algorithms to interpret any residual annotations. Performance for this zipper-driven strategy is proportionately as good as XLE in the context-free-equivalent case that XLE does particularly well at, is proportionately better than XLE if the particular sentence has only zipper dependencies, and is proportionately no worse than XLE if the sentence involves more complex annotations that interact in more intricate ways.

2 Finitely bounded LFG grammars

Seki et al. (1993) first established the connection between a restricted subclass of LFG grammars and formal systems that can describe only mildly context-sensitive dependencies. Their *finite copying grammars* permit rules with the very limited functional annotations in (1a) and that also satisfy the bounding condition (1b).

(1) a. Each category on the right-side of a rule can be annotated with at most one function assignment of the form $(\uparrow \text{F}) = \downarrow$ and any number of atom-value assignments only of the form $(\uparrow \text{A}) = \text{V}$.

b. There is a constant k such that no more than k nodes map to the same f-structure element f in any derivation. That is $|\phi^{-1}(f)| \leq k$.

It is decidable whether the bounding condition holds for such a notationally restricted grammar, and such a bounded grammar can be converted to an equivalent LCFRS. A grammar with these annotations is expressive enough to specify zipper paths as in Figure 1, but these restrictions are obviously too severe for linguistic description. This notation disallows, for example, the trivial $\uparrow = \downarrow$ annotations that mark the heads and coheads in the functional domain of a predicate, reentrancies such as $(\uparrow \text{XCOMP SUBJ}) = (\uparrow \text{OBJ})$ that represent functional control, multi-attribute value specifications, such as $(\uparrow \text{SUBJ NUM}) = \text{SG}$, that encode agreement requirements, and any direct specification of feature values on daughter nodes, as in $(\downarrow \text{CASE}) = \text{NOM}$.

The finitely bounded grammars of Wedekind and Kaplan (2020) allow the linguistically more suitable annotations in (2), but they must also satisfy other conditions whose effect is to limit their expressive power and endow them with the same mathematical and computational properties, including LCFRS equivalence, as Seki et. al's finite copying grammars.

(2) Basic annotations

$(\uparrow/\downarrow \text{ A B C} \cdots) = \text{V}$	general atom-value annotations
$(\uparrow \text{F}) = \downarrow$	function assignment
$\uparrow = \downarrow$	trivial (co)head identity

Reentrancies

$(\uparrow \text{F G}) = (\uparrow \text{H})$	functional control
$(\uparrow \text{F}) = (\uparrow \text{H})$	local-topic link
$(\downarrow \text{G}) = (\uparrow \text{H})$	daughter-mother control
$(\downarrow \text{G}) = (\downarrow \text{H})$	daughter sharing
$(\downarrow \text{G}) = \uparrow$	promotion
$(\uparrow \text{F}) = \uparrow$	mother cycle
$(\downarrow \text{G}) = \downarrow$	daughter cycle

The additional conditions that a finitely bounded grammar G must meet are listed in (3) (Wedekind and Kaplan 2020).

(3) a. Each right-side category is annotated with at most one function assignment $(\uparrow \text{F}) = \downarrow$ and trivial (co)head identities $\uparrow = \downarrow$ and function assignments always appear in complementary distribution (to keep separate the properties of a head and its complements).

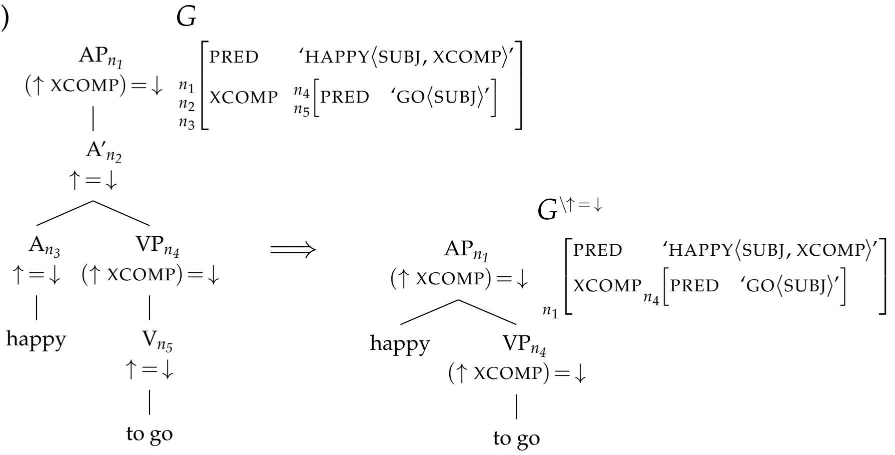
b. The *functional domains* of G (the collections of $\uparrow = \downarrow$-annotated nodes that map to the same f-structure) are height-bounded.

c. The *reentrancy-free kernel* of G (the grammar formed by removing all reentrancies from G) is bounded as in (1b).

d. Reentrancies are nonconstructive.

There is a simple transformation of a grammar G with height-bounded functional domains into a strongly equivalent LFG grammar $G^{\setminus\uparrow=\downarrow}$ that no longer contains $\uparrow = \downarrow$ annotations. The transformation is accomplished by recursively replacing a category annotated with $\uparrow = \downarrow$ in the right side of one rule by the right sides of all the rules expanding that category, and making the appropriate replacements of $\uparrow$ for $\downarrow$ to preserve the f-structure mappings. The effect of this transformation is illustrated in (4).

(4)



Although the grammar $G^{\setminus\uparrow=\downarrow}$ resulting from this simple transformation may be substantially larger than G, the transformation makes it unnecessary to give further consideration to $\uparrow = \downarrow$ annotations. And of relevance to present purposes, it exposes any zipper paths that trivial annotations may otherwise obscure, as pictured in Figure 2.

The nonconstructivity condition (3d) ensures that only function assignments (the zipper-forming annotations of finite copying grammars), can cause two nodes to map to the same f-structure.[1] The difference between constructive and nonconstructive reentrancies is illustrated in Figure 3. On the left side the reentrancies are constructive because they cause the nodes

[1] This condition has appeared implicitly in LFG grammars and has also been mentioned in the LFG literature (Crouch et al. 2008, Zaenen and Kaplan 1995).

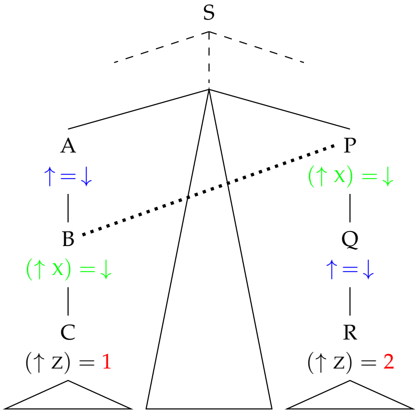


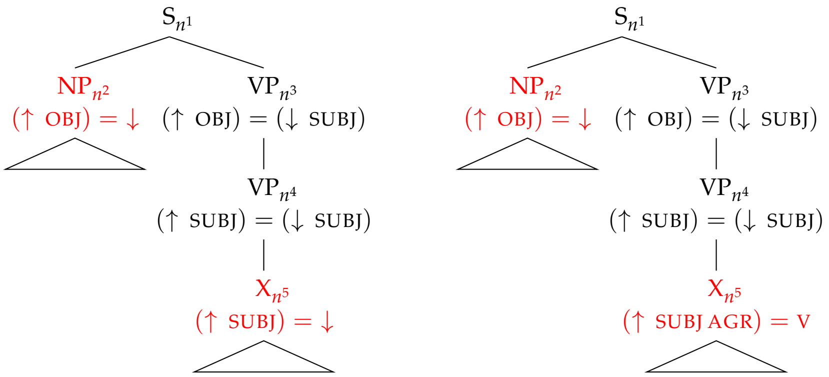
Figure 2: Head identities obscure zippers.

n^2 and n^5 to map to the same f-structure element. If reentrancies are nonconstructive, as in the derivation on the right side, they can only propagate atom-value information across the f-structure elements. Nonconstructive reentrancies do not introduce new node-to-f-structure mappings and thus do not affect the bounds on the ϕ^{-1} node classes.

The nonconstructivity of reentrancies is undecidable for grammars with functional control annotations (Wedekind and Kaplan 2020). However, in derivations that meet the requirements of the Coherence Condition, annotations such as $(\uparrow \textsc{xcomp subj}) = (\uparrow \textsc{obj})$ can always be reduced to daughter-mother control annotations. This is because the controllee (SUBJ) is a governable function in an open (XCOMP) complement and therefore must be licensed by the complement's semantic form. These licensing semantic forms are always introduced by simple PRED equations associated with individual lexical entries, such as $(\uparrow \textsc{pred}) = \text{‘}\textsc{walk}\langle\textsc{subj}\rangle\text{’}$. Therefore, $(\uparrow \textsc{pred}) = \text{‘}\textsc{walk}\langle\textsc{subj}\rangle\text{’}$ must instantiate to the equation $(\phi(n')\ \textsc{pred}) = \text{‘}\textsc{walk}\langle\textsc{subj}\rangle\text{’}$ at some node n', and the f-description must also entail an equation $(\phi(n)\ \textsc{xcomp}) = \phi(n')$ that links the complement to a higher clause and is also available to shorten the control equation. Wedekind and Kaplan (2020) provide a formal specification of nonconstructivity, this expected consequence of Coherence, and of other technical requirements that are sufficient to decide whether an arbitrary LFG grammar belongs to the finitely bounded subclass and therefore has an LCFRS equivalent.

3 Direct LCFRS parsing

For a k-bounded LFG G the equivalent LCFRS G' is constructed by precomputing the zipper interactions in G. Because trivial annotations obscure zippers, as depicted in Figure 2, the LCFRS is constructed from $G^{\backslash\uparrow=\downarrow}$ rules


S_{n^1}
NP_{n^2}
$(\uparrow \text{OBJ}) = \downarrow$
VP_{n^3}
$(\uparrow \text{OBJ}) = (\downarrow \text{SUBJ})$
VP_{n^4}
$(\uparrow \text{SUBJ}) = (\downarrow \text{SUBJ})$
X_{n^5}
$(\uparrow \text{SUBJ}) = \downarrow$
S_{n^1}
NP_{n^2}
$(\uparrow \text{OBJ}) = \downarrow$
VP_{n^3}
$(\uparrow \text{OBJ}) = (\downarrow \text{SUBJ})$
VP_{n^4}
$(\uparrow \text{SUBJ}) = (\downarrow \text{SUBJ})$
X_{n^5}
$(\uparrow \text{SUBJ AGR}) = \text{V}$

Constructive

$\phi(n^2) = \phi(n^5)$

Nonconstructive

$(\phi(n^2)\ \text{OBJ AGR}) = \text{V}$

Figure 3: Constructive and nonconstructive reentrancies.

rather than G rules. Thus the LCFRS for G is constructed in two stages. In the first stage a $\uparrow=\downarrow$-free LFG grammar $G^{\setminus\uparrow=\downarrow}$ is created by eliminating the $\uparrow=\downarrow$-annotated categories in favor of equivalent collections of flattened LFG rules. The second stage of the construction produces LCFRS rules for $G^{\setminus\uparrow=\downarrow}$. The LCFRS rule construction is based on locally disclosing structure sharing through zipper unification, as illustrated in Figure 1. The construction hypothesizes finite sequences of $G^{\setminus\uparrow=\downarrow}$ rules that might expand the categories realizing a k-bounded zipper, and it builds an LCFRS rule if the sequence gives rise to satisfiable f-descriptions. The LCFRS rule categories are refined by atom-value decorations containing the atomic-valued information that could be associated with the corresponding f-structure element in any valid LFG derivation (see Wedekind and Kaplan (2020) for more details on the construction and the parsing complexity for the LCFRSs G' that result from linguistically motivated k-bounded LFG grammars.)

The LCFRS G' can be used to parse sentences from $L(G)$, provided the LFG grammar G is finitely bounded. (XLE or some other LFG parser must be used if G is not finitely bounded.) LCFRS parsing complexity is $\mathcal{O}(|G'| \cdot n^{k(r+1)})$ (Seki et al. 1991) where n is the length of the input string, $|G'|$ is the number of rules in G', k is the fan-out of G' (the degree of discontinuity of G), and r is the rank of G' (the maximum number of phrasal categories in any G' rule). Parsing complexity is polynomial in the length of the input string (n) but, without further restrictions, parsing with the equivalent grammar may be impractical because the LCFRS G' can be exponentially larger than G (Wedekind and Kaplan 2020).

For linguistically motivated grammars, however, the potential growth is limited by conventions and principles of LFG theory and the properties of natural languages. In LFG, the distribution of trivial annotations is regulated by the principles of X-bar theory and its structure-function mapping principles (Bresnan 2001, Dalrymple 2001). In this (epsilon-free) framework the height of a functional domain is effectively bounded by the maximum number of coheads that can associate to a single predicate plus 1 for the head (denoted by c), the maximum number k of discontinuous c-structure phrases that can realize a particular function, and the maximum number g of different grammatical functions that an individual predicate can govern. Thus parsing complexity for a linguistically motivated grammar G is proportional to $|G^{\setminus\uparrow=\downarrow}| \leq |G|^{kg+c+1}$, where k, g, and c are typically rather small.[2] (In the broad-coverage, commercial-grade ParGram grammars, for example, no word in either lexicon governs more than four functions, and very few words allow even that many (in English only the word *bet*)).

In the second phase of the LCFRS construction, sequences of $G^{\setminus\uparrow=\downarrow}$ rules are converted into LCFRS rules with decorated categories. From the observations above, we can assume that for a linguistically motivated LFG G the rank of G' is bounded by $g + c$, the LCFRS categories for the grammatical functions are at most k-ary, and the categories for the coheads are unary. Thus the size bound on $G^{\setminus\uparrow=\downarrow}$ accounts for rule sequences of length up to k and therefore the number of LCFRS rules before they are decorated with agreement features. Those skeletal rules are refined by the combinations of agreement features that are associated with particular syntactic categories and grammatical functions, and the number of these combinations is limited by morphosyntactic constraints (nouns carry PERS and NUM but not TNS). Thus, with a denoting the maximum number of attested agreement feature combinations, the size of G' is bounded by $a^{g+c+1}|G|^{kg+c+1}$. (For instance, for English NP f-structures the number of (fully-specified) agreement feature combinations would be $24 = 3(\text{PERS}) \cdot 2(\text{NUM}) \cdot 4(\text{CASE})$; as shown in Wedekind and Kaplan (2020), the predicate values (semantic forms) do not have to be distinguished.)

4 Grammar specialization

Even if it is feasible to construct the LCFRS for a linguistically motivated grammar in its entirety, that may not be the best way of taking advantage of the mildly context-sensitive dependencies of natural language. As alternatives that may be more effective, we consider parsing strategies that avoid constructing the LCFRS for the whole grammar and instead only operate on the typically much smaller subset of annotated c-structure rules that participate in the analysis of a given input string. Such a specialized grammar may be finitely bounded even if the entire grammar is not, and the corresponding LCFRS may be much more manageable. Grammar specialization is also the first stage of a zipper-driven parsing strategy that may

[2] The exponent is increased by 1 to account for trivial-free rules obtained from functional domains smaller than $kg + c$.

be helpful for broad-coverage grammars that do not meet all the conditions of finite bounding.

We first apply a context-free chart parser to a given input string *s*, as does XLE, but we do not then execute the bottom-up traversal of chart edges (Maxwell and Kaplan 1996) to check for f-description satisfiability. Instead, we extract from the resulting parse-chart an LFG grammar G_s that has all and only the rules and annotations that are specialized to the particular input *s*. Grammar specialization depends on the fact that context-free languages are closed under intersection with regular languages (Bar-Hillel et al. 1961). As Lang (1992) and others have pointed out, a context-free grammar specialized to a particular *s* can be extracted in cubic time by any number of context-free parsing algorithms, and such an algorithm can easily be modified to record the annotations associated with the categories even though those are not evaluated during the context-free parse. The size of the resulting grammar G_s is proportional to $|s|^3$ (and G_s is *k*-bounded if *G* is *k*-bounded).

We illustrate grammar specialization with an analysis of the Dutch double infinitive construction in (5).

(5) ... (dat) hij het boek heeft kunnen lezen
... (that) he the book has able read
... (that) he has been able to read the book

This sentence is assigned the annotated c-structure and f-structure depicted in Figure 4.[3] For sentence (5) and the grammar *G* of Bresnan et al. (1982) we obtain the specialized grammar G_s that includes the rules in (6).

[3] Johnson (1986) used this example to demonstrate that the natural extension for these sentences of the Dutch grammar of Bresnan et al. (1982) violates the Nonbranching Dominance Constraint, and thus calls into question the linguistic suitability of the Kaplan and Bresnan (1982) formulation. In fact, this particular sentence does not violate the later refinement of the NBD constraint described by Kaplan and Maxwell (1996) and Dalrymple (2001) wherein functional annotations are also taken into account in determining whether a category has repeated. The recursive VPs in this sentence have different annotations, but sentences with more intransitive verbs and deeper XCOMP embeddings would still be disallowed. We return to this point below.

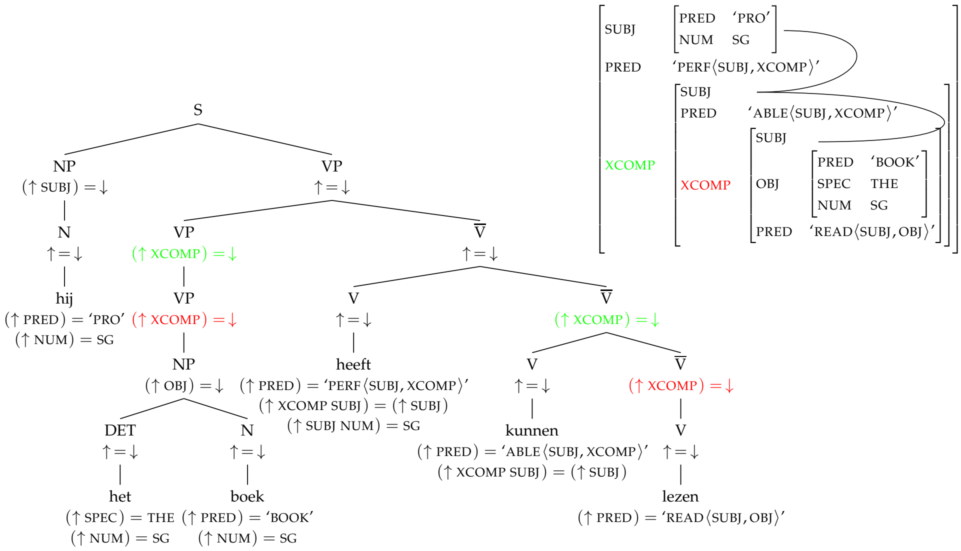


Figure 4: The Bresnan et al. (1982) analysis of sentence (5). The zipped functions are indicated in green and red.

(6)

$${}_0\text{S}_6 \rightarrow \underset{(\uparrow \text{SUBJ}) = \downarrow}{{}_0\text{NP}_1} \quad \underset{\uparrow = \downarrow}{{}_1\text{VP}_6} \qquad\qquad {}_1\text{VP}_6 \rightarrow \underset{(\uparrow \text{XCOMP}) = \downarrow}{{}_1\text{VP}_3} \quad \underset{\uparrow = \downarrow}{{}_3\overline{\text{V}}_6}$$

$${}_1\text{VP}_3 \rightarrow \underset{(\uparrow \text{XCOMP}) = \downarrow}{{}_1\text{VP}_3} \qquad\qquad {}_1\text{VP}_3 \rightarrow \underset{(\uparrow \text{OBJ}) = \downarrow}{{}_1\text{NP}_3}$$

$${}_3\overline{\text{V}}_6 \rightarrow \underset{\uparrow = \downarrow}{{}_3\text{V}_4} \quad \underset{(\uparrow \text{XCOMP}) = \downarrow}{{}_4\overline{\text{V}}_6} \qquad\qquad {}_4\overline{\text{V}}_6 \rightarrow \underset{\uparrow = \downarrow}{{}_4\text{V}_5} \quad \underset{(\uparrow \text{XCOMP}) = \downarrow}{{}_5\overline{\text{V}}_6}$$

$${}_0\text{NP}_1 \rightarrow \underset{\uparrow = \downarrow}{{}_0\text{N}_1} \qquad\qquad {}_1\text{NP}_3 \rightarrow \underset{\uparrow = \downarrow}{{}_1\text{DET}_2} \quad \underset{\uparrow = \downarrow}{{}_2\text{N}_3}$$

$${}_5\overline{\text{V}}_6 \rightarrow \underset{\uparrow = \downarrow}{{}_5\text{V}_6} \qquad\qquad {}_0\text{N}_1 \rightarrow \begin{array}{c} {}_0\text{hij}_1 \\ (\uparrow \text{PRED}) = \text{'PRO'} \\ (\uparrow \text{NUM}) = \text{SG} \end{array}$$

$${}_1\text{DET}_2 \rightarrow \begin{array}{c} {}_1\text{het}_2 \\ (\uparrow \text{SPEC}) = \text{THE} \\ (\uparrow \text{NUM}) = \text{SG} \end{array} \qquad\qquad {}_2\text{N}_3 \rightarrow \begin{array}{c} {}_2\text{boek}_3 \\ (\uparrow \text{PRED}) = \text{'BOOK'} \\ (\uparrow \text{NUM}) = \text{SG} \end{array}$$

$${}_3\text{V}_4 \rightarrow \begin{array}{c} {}_3\text{heeft}_4 \\ (\uparrow \text{PRED}) = \text{'PERF}\langle\text{SUBJ}, \text{XCOMP}\rangle\text{'} \\ (\uparrow \text{XCOMP SUBJ}) = (\uparrow \text{SUBJ}) \\ (\uparrow \text{SUBJ NUM}) = \text{SG} \end{array} \qquad\qquad {}_4\text{V}_5 \rightarrow \begin{array}{c} {}_4\text{kunnen}_5 \\ (\uparrow \text{PRED}) = \text{'ABLE}\langle\text{SUBJ}, \text{XCOMP}\rangle\text{'} \\ (\uparrow \text{XCOMP SUBJ}) = (\uparrow \text{SUBJ}) \end{array}$$

$${}_5\text{V}_6 \rightarrow \begin{array}{c} {}_5\text{lezen}_6 \\ (\uparrow \text{PRED}) = \text{'READ}\langle\text{SUBJ}, \text{OBJ}\rangle\text{'} \end{array}$$

The specialized grammar G_s contains refinements of all and only the G rules that describe the c-structures of (5). The categories of those rules are elaborated with indexes that record the beginning and ending positions of the substrings of s that they dominate. Thus the category ${}_0\text{S}_6$ is the refinement of S that covers the entire sentence and the category ${}_1\text{NP}_3$ covers the words of the second NP. The terminals are also refined with their particular string positions, so grammar G_s derives the specialized string in (7)

(7) ${}_0\text{hij}_1$ ${}_1\text{het}_2$ ${}_2\text{boek}_3$ ${}_3\text{heeft}_4$ ${}_4\text{kunnen}_5$ ${}_5\text{lezen}_6$

if and only if G derives the original input (5) and both are assigned the same f-structures. Note that there are infinitely many annotated c-structures for the specialized input sentence because the VP rule

$$ {}_1\mathrm{VP}_3 \;\rightarrow\; \begin{array}{c} {}_1\mathrm{VP}_3 \\ (\uparrow \textsc{xcomp}) = \downarrow \end{array} $$

is recursive and thus allows for derivations that violate the Nonbranching Dominance Condition.

If the emptiness algorithm for context-free languages establishes that $L(G_s) = \emptyset$, we know that s is ungrammatical with respect to G. Otherwise, s has at least one annotated c-structure and further analysis is necessary to determine whether any G_s derivations also meet the functional well-formedness requirements of LFG theory. As Wedekind and Kaplan (2020) have demonstrated, it is decidable whether G_s is finitely bounded and thus whether an equivalent LCFRS G'_s can be constructed to resolve the functional annotations for s. As noted, the complexity of LCFRS parsing for the entire grammar G (if in fact it is finitely bounded) is $\mathcal{O}(|G'| \cdot n^{k(r+1)})$ where G' is bounded by $a^{g+c+1}|G|^{kg+c+1}$. This formula applies to the specialized LCFRS G'_s but with parameters a_s, k_s, g_s, c_s, r_s that are typically much smaller and more likely to be practical than a, k, g, c, r (for our Dutch example sentence, for example, k_s is 2, g_s is 2, and r_s is 2).

The construction of G'_s begins with a top-down traversal of G_s that evaluates the annotations for c-structure paths with parallel function assignments to determine whether the zippers are bounded. If the zippers are bounded, this is followed by a bottom-up pass to detect constructive reentrancies and to test compatibility of any atom-valued features that might be promoted upwards by (nonconstructive) reentrancies. All this effort would be wasted for the (putatively rare) sentences for which G_s fails to meet the bounding conditions and parsing reverts to an MK bottom-up execution sequence. An alternative is to perform only the top-down zipper traversal in every case and use the information it uncovers to guide the operation of the bottom-up algorithms. The overall process will approach LCFRS efficiency automatically for specialized grammars that happen to be finitely bounded. This is the zipper-driven strategy that we illustrate below.

5 Zipper-driven parsing

We start as above by extracting the specialized grammar G_s from the context-free parser-chart for s. We then transform the rules of that grammar to produce a zipper-free grammar G^z_s whose annotations are free of zipper-entailed inconsistencies but continue to define all the f-structures of G_s and thus also of G. The rules of G^z_s are subsequently interpreted as a parse-chart that MK algorithms can operate on to check for inconsistencies that escaped the top-down zipper identification (for example, those arising from constructive reentrancies), if any. If there are no such inconsistencies, the bottom-up satisfiability check will quickly verify that the language of the annotated context-free grammar G_s^z is not empty. Otherwise, zipper-driven parsing may degrade to the performance of XLE for constructions that can only be described by more intricate annotations. In the following we describe these steps in more detail.

The construction of G_s^z from G_s involves several operations the first of which is to eliminate trivial $\uparrow = \downarrow$ annotations by promoting the daughter category strings of the rules that expand a trivially-annotated category. The trivial-free grammar $G_s^{\setminus \uparrow = \downarrow}$ that we obtain from G_s through trivial elimination is shown in (8).

(8) a. $${}_0\mathrm{S}_6 \rightarrow \begin{array}{c} {}_0\mathrm{NP}_1 \\ (\uparrow \mathrm{SUBJ}) = \downarrow \end{array} \quad \begin{array}{c} {}_1\mathrm{VP}_3 \\ (\uparrow \mathrm{XCOMP}) = \downarrow \end{array} \quad \begin{array}{c} {}_3\mathrm{heeft}_4 \\ (\uparrow \mathrm{PRED}) = \text{'PERF}\langle \mathrm{SUBJ}, \mathrm{XCOMP} \rangle\text{'} \\ (\uparrow \mathrm{XCOMP\ SUBJ}) = (\uparrow \mathrm{SUBJ}) \\ (\uparrow \mathrm{SUBJ\ NUM}) = \mathrm{SG} \end{array} \quad \begin{array}{c} {}_4\overline{\mathrm{V}}_6 \\ (\uparrow \mathrm{XCOMP}) = \downarrow \end{array}$$

b. $${}_1\mathrm{VP}_3 \rightarrow \begin{array}{c} {}_1\mathrm{VP}_3 \\ (\uparrow \mathrm{XCOMP}) = \downarrow \end{array}$$

c. $${}_1\mathrm{VP}_3 \rightarrow \begin{array}{c} {}_1\mathrm{NP}_3 \\ (\uparrow \mathrm{OBJ}) = \downarrow \end{array}$$

d. $${}_4\overline{\mathrm{V}}_6 \rightarrow \begin{array}{c} {}_4\mathrm{kunnen}_5 \\ (\uparrow \mathrm{PRED}) = \text{'ABLE}\langle \mathrm{SUBJ}, \mathrm{XCOMP} \rangle\text{'} \\ (\uparrow \mathrm{XCOMP\ SUBJ}) = (\uparrow \mathrm{SUBJ}) \end{array} \quad \begin{array}{c} {}_5\overline{\mathrm{V}}_6 \\ (\uparrow \mathrm{XCOMP}) = \downarrow \end{array}$$

e. $${}_5\overline{\mathrm{V}}_6 \rightarrow \begin{array}{c} {}_5\mathrm{lezen}_6 \\ (\uparrow \mathrm{PRED}) = \text{'READ}\langle \mathrm{SUBJ}, \mathrm{OBJ} \rangle\text{'} \end{array}$$

f. $${}_0\mathrm{NP}_1 \rightarrow \begin{array}{c} {}_0\mathrm{hij}_1 \\ (\uparrow \mathrm{PRED}) = \text{'PRO'} \\ (\uparrow \mathrm{NUM}) = \mathrm{SG} \end{array}$$

g. $${}_1\mathrm{NP}_3 \rightarrow \begin{array}{c} {}_1\mathrm{het}_2 \\ (\uparrow \mathrm{SPEC}) = \mathrm{THE} \\ (\uparrow \mathrm{NUM}) = \mathrm{SG} \end{array} \quad \begin{array}{c} {}_2\mathrm{boek}_3 \\ (\uparrow \mathrm{PRED}) = \text{'BOOK'} \\ (\uparrow \mathrm{NUM}) = \mathrm{SG} \end{array}$$

We then execute a simple top-down strategy for identifying zippers and solving their functional constraints. The zippers in this process are represented as those annotated subsets of specialized terminals and nonterminals that result from expanding categories top-down from the zipper-root $\{{}_0\mathrm{S}_6\}$ and grouping together daughter categories that are annotated with the same function assignment. Thus the root is expanded with rule (8a), and the instantiated description of the derived annotated categories is tested for well-formedness. This test eliminates as ill-formed zipper rules with inconsistent descriptions and rules that cannot be rendered complete and coherent through bottom-up propagation. In our example, the description is consistent and the subcategorization requirements of the local predicate are satisfied. Moreover, the XCOMP function assignment common to ${}_1\mathrm{VP}_3$ and ${}_4\overline{\mathrm{V}}_6$, depicted in green, gives rise to a two-element set for the discontinuous zipper constituent $\{{}_1\mathrm{VP}_3, {}_4\overline{\mathrm{V}}_6\}$. This reflects the fact that the discontinuity bound for this construction (and for Dutch as a whole) is two. The zipper conversion of (8a) is illustrated in (9).

(9)

$$\begin{array}{lcccc}
{}_0\mathrm{S}_6 \to & {}_0\mathrm{NP}_1 & {}_1\mathrm{VP}_3 & {}_3\mathrm{heeft}_4 & {}_4\overline{\mathrm{V}}_6 \\
 & (\uparrow \textsc{subj}) = \downarrow & (\uparrow \textsc{xcomp}) = \downarrow & (\uparrow \textsc{pred}) = \text{`}\textsc{perf}\langle \textsc{subj}, \textsc{xcomp}\rangle\text{'} & (\uparrow \textsc{xcomp}) = \downarrow \\
 & & & (\uparrow \textsc{xcomp subj}) = (\uparrow \textsc{subj}) & \\
 & & & (\uparrow \textsc{subj num}) = \textsc{sg} &
\end{array}$$

$$\Downarrow$$

$$\begin{array}{lccc}
\{{}_0\mathrm{S}_6\} \to & \{{}_0\mathrm{NP}_1\} & \{{}_1\mathrm{VP}_3, {}_4\overline{\mathrm{V}}_6\} & \{{}_3\mathrm{heeft}_4\} \\
 & (\downarrow \textsc{num}) = \textsc{sg} & (\downarrow \textsc{subj num}) = \textsc{sg} & (\uparrow \textsc{pred}) = \text{`}\textsc{perf}\langle \textsc{subj}, \textsc{xcomp}\rangle\text{'} \\
 & (\uparrow \textsc{subj}) = \downarrow & (\uparrow \textsc{xcomp}) = \downarrow & \\
 & & (\downarrow \textsc{subj}) = (\uparrow \textsc{subj}) &
\end{array}$$

The shared assignment also allows the functional control equation to be shortened to a mother-daughter control that specifies the identity of the matrix and embedded clause subjects. Because of the SUBJ assignment to the initial NP, the agreement requirement of the verb migrates to the $\{{}_0\mathrm{NP}_1\}$ so that it can propagate top-down and eventually come into contact with the singular pronoun. The number agreement feature also transfers to $\{{}_1\mathrm{VP}_3, {}_4\overline{\mathrm{V}}_6\}$ against the possibility left open by $(\downarrow \textsc{subj}) = (\uparrow \textsc{subj})$ that the subject is realized in the embedded clause.

Now consider the decorated zipped ${}_1\mathrm{VP}_3, {}_4\overline{\mathrm{V}}_6$ daughters of the zipper rule in (9) depicted in (10).

(10)

$$\begin{array}{c}
\{{}_1\mathrm{VP}_3, {}_4\overline{\mathrm{V}}_6\} \\
(\downarrow \textsc{subj num}) = \textsc{sg}
\end{array}$$

$G_s^{\setminus \uparrow = \downarrow}$ has the two alternative expansions (11a,b) for ${}_1\mathrm{VP}_3$ but only a single rule for ${}_4\overline{\mathrm{V}}_6$ (11c).

(11) a.

$$\begin{array}{lc}
{}_1\mathrm{VP}_3 \to & {}_1\mathrm{NP}_3 \\
 & (\uparrow \textsc{obj}) = \downarrow
\end{array}$$

b.

$$\begin{array}{lc}
{}_1\mathrm{VP}_3 \to & {}_1\mathrm{VP}_3 \\
 & (\uparrow \textsc{xcomp}) = \downarrow
\end{array}$$

c.

$$\begin{array}{lcc}
{}_4\overline{\mathrm{V}}_6 \to & {}_4\mathrm{kunnen}_5 & {}_5\overline{\mathrm{V}}_6 \\
 & (\uparrow \textsc{pred}) = \text{`}\textsc{able}\langle \textsc{subj}, \textsc{xcomp}\rangle\text{'} & (\uparrow \textsc{xcomp}) = \downarrow \\
 & (\uparrow \textsc{xcomp subj}) = (\uparrow \textsc{subj}) &
\end{array}$$

Here, the expansion with (11a,c) yields the daughter combinations in (12a) and the one with (11b,c) the combinations in (12b). The atom-value information that is inherited from the decoration of the mother (10) is shown in blue. (By convention, we assume that this information is inherited to the leftmost daughter.)

(12) a.

$$\begin{array}{ccc}
{}_1\mathrm{NP}_3 & {}_4\mathrm{kunnen}_5 & {}_5\overline{\mathrm{V}}_6 \\
(\uparrow \textsc{obj}) = \downarrow & (\downarrow \textsc{subj num}) = \textsc{sg} & (\uparrow \textsc{xcomp}) = \downarrow \\
(\uparrow \textsc{subj num}) = \textsc{sg} & (\uparrow \textsc{pred}) = \text{`}\textsc{able}\langle \textsc{subj}, \textsc{xcomp}\rangle\text{'} & \\
 & (\uparrow \textsc{xcomp subj}) = (\uparrow \textsc{subj}) &
\end{array}$$

b.

$$\begin{array}{ccc}
{}_1\mathrm{VP}_3 & {}_4\mathrm{kunnen}_5 & {}_5\overline{\mathrm{V}}_6 \\
(\uparrow \textsc{xcomp}) = \downarrow & (\downarrow \textsc{subj num}) = \textsc{sg} & (\uparrow \textsc{xcomp}) = \downarrow \\
(\uparrow \textsc{subj num}) = \textsc{sg} & (\uparrow \textsc{pred}) = \text{`}\textsc{able}\langle \textsc{subj}, \textsc{xcomp}\rangle\text{'} & \\
 & (\uparrow \textsc{xcomp subj}) = (\uparrow \textsc{subj}) &
\end{array}$$

Because the daughter combination in (12a) cannot be rendered coherent through bottom-up propagation (the object is not subcategorized by the predicate ABLE) the expansion with (11a,c) does not result in a well-formed zipper rule. For the expansion with (11b,c) on the other hand we obtain the zipper rule in (13).

(13)
$$\begin{array}{cccc}
\{{}_1\mathrm{VP}_3,{}_4\overline{\mathrm{V}}_6\} & \rightarrow & \{{}_1\mathrm{VP}_3,{}_5\overline{\mathrm{V}}_6\} & \{{}_4\mathrm{kunnen}_5\} \\
(\downarrow \textsc{subj num}) = \textsc{sg} & & (\downarrow \textsc{subj num}) = \textsc{sg} & (\uparrow \textsc{pred}) = \text{‘}\textsc{able}\langle \textsc{subj}, \textsc{xcomp}\rangle\text{’} \\
 & & (\uparrow \textsc{xcomp}) = \downarrow & \\
 & & (\downarrow \textsc{subj}) = (\uparrow \textsc{subj}) &
\end{array}$$

The entire zipper grammar G_s^z is shown in (14).

(14)
$$\begin{array}{ccccc}
\{{}_0\mathrm{S}_6\} & \rightarrow & \{{}_0\mathrm{NP}_1\} & \{{}_1\mathrm{VP}_3,{}_4\overline{\mathrm{V}}_6\} & \{{}_3\mathrm{heeft}_4\} \\
 & & (\downarrow \textsc{num}) = \textsc{sg} & (\downarrow \textsc{subj num}) = \textsc{sg} & (\uparrow \textsc{pred}) = \text{‘}\textsc{perf}\langle \textsc{subj}, \textsc{xcomp}\rangle\text{’} \\
 & & (\uparrow \textsc{subj}) = \downarrow & (\uparrow \textsc{xcomp}) = \downarrow & \\
 & & & (\downarrow \textsc{subj}) = (\uparrow \textsc{subj}) &
\end{array}$$

$$\begin{array}{cccc}
\{{}_1\mathrm{VP}_3,{}_4\overline{\mathrm{V}}_6\} & \rightarrow & \{{}_1\mathrm{VP}_3,{}_5\overline{\mathrm{V}}_6\} & \{{}_4\mathrm{kunnen}_5\} \\
(\downarrow \textsc{subj num}) = \textsc{sg} & & (\downarrow \textsc{subj num}) = \textsc{sg} & (\uparrow \textsc{pred}) = \text{‘}\textsc{able}\langle \textsc{subj}, \textsc{xcomp}\rangle\text{’} \\
 & & (\uparrow \textsc{xcomp}) = \downarrow & \\
 & & (\downarrow \textsc{subj}) = (\uparrow \textsc{subj}) &
\end{array}$$

$$\begin{array}{cccc}
\{{}_1\mathrm{VP}_3,{}_5\overline{\mathrm{V}}_6\} & \rightarrow & \{{}_1\mathrm{NP}_3\} & \{{}_5\mathrm{lezen}_6\} \\
(\downarrow \textsc{subj num}) = \textsc{sg} & & (\uparrow \textsc{obj}) = \downarrow & (\uparrow \textsc{pred}) = \text{‘}\textsc{read}\langle \textsc{subj}, \textsc{obj}\rangle\text{’}
\end{array}$$

$$\begin{array}{ccc}
\{{}_0\mathrm{NP}_1\} & \rightarrow & \{{}_0\mathrm{hij}_1\} \\
(\downarrow \textsc{num}) = \textsc{sg} & & (\uparrow \textsc{pred}) = \text{‘}\textsc{pro}\text{’} \\
 & & (\uparrow \textsc{num}) = \textsc{sg}
\end{array}$$

$$\begin{array}{cccc}
\{{}_1\mathrm{NP}_3\} & \rightarrow & \{{}_1\mathrm{het}_2\} & \{{}_2\mathrm{boek}_3\} \\
 & & (\uparrow \textsc{spec}) = \textsc{the} & (\uparrow \textsc{pred}) = \text{‘}\textsc{book}\text{’} \\
 & & (\uparrow \textsc{num}) = \textsc{sg} & (\uparrow \textsc{num}) = \textsc{sg}
\end{array}$$

The analysis of sentence (5) provided by the zipper grammar (14) appears in Figure 5.

For our Dutch grammar the top-down strategy for G_s^z is certainly sufficient to ensure that G_s^z will be a finite encoding of all and only the derivations of s in G. But that is not always the case even for finitely bounded LFGs. The top-down pass is insufficient if completeness and coherence depend on predicates or governable functions that propagate bottom-up. Moreover, the top-down construction may also fail to detect all inconsistencies. The top-down process creates a single descending branch for all daughter categories that share the same function assignment $(\uparrow \textsc{f}) = \downarrow$ and separate zipper branches for daughters with other assignments $(\uparrow \textsc{g}) = \downarrow$. Those branches are typically independent with respect to agreement features, but that is not necessarily the case if the separate branches have annotations that lift agreement features from daughter nodes. For example, if the F and G branches have promotions $(\downarrow \textsc{x}) = \uparrow$ and $(\downarrow \textsc{y}) = \uparrow$, then the X and Y values of the separate branches come into contact at the common mother and therefore must be consistent. Similarly, if the separate

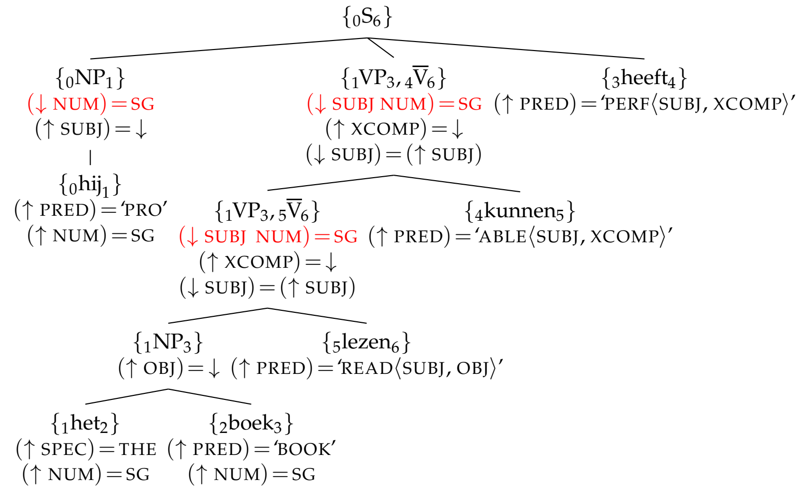


Figure 5: The zipper grammar analysis of sentence (5).

branches are annotated with control equations of the form $(\downarrow \text{X}) = (\uparrow \text{Z})$ and $(\downarrow \text{Y}) = (\uparrow \text{Z})$, then the lower X and Y values must be consistent as values of the common $(\uparrow \text{Z})$. Annotation combinations such as these may not be typical of linguistically motivated grammars, but additional tests through bottom-up propagation are necessary if they are encountered as the top-down process unfolds (as described in Wedekind and Kaplan (2020)).

Thus suppose that G_s happens to be *top-down complete* in the sense that the top-down traversal is sufficient to guarantee that G_s^z encodes all and only the valid derivations of s in G. Then the bottom-up MK algorithms will quickly check that there is at least one derivation of s in G_s^z and arrange it so that all of its f-structures can be read out each in linear time.

In sum, the incremental zipper-driven parsing algorithm performs the following steps:

1. Specialize G to an LFG grammar G_s characterizing all/only annnotated c-structures for s in G. If $L(G_s) \neq \emptyset$, move to step 2. Otherwise, stop and report that there is no parse for s.

2. Construct LFG rules for zipper grammar G_s^z of G_s:

   Step 2a: Eliminate identity annotations to produce $G_s^{\setminus\uparrow=\downarrow}$.

   Step 2b: Create G_s^z from candidate subsets of $G_s^{\setminus\uparrow=\downarrow}$ rules.[4]

3. Use MK algorithms to test $L(G_s^z) \neq \emptyset$ and prepare for the enumeration of the f-structures assigned to s.

[4] It is also possible to interleave $\uparrow = \downarrow$-elimination (Step 2a) with zipper identification (Step 2b) for epsilon-free grammars, but the process would have been more difficult to illustrate. The interleaved elimination process will terminate even without an explicit bound on the height of functional domains if rules that would obviously generate nonbranching dominance chains in G_s^z are discarded.

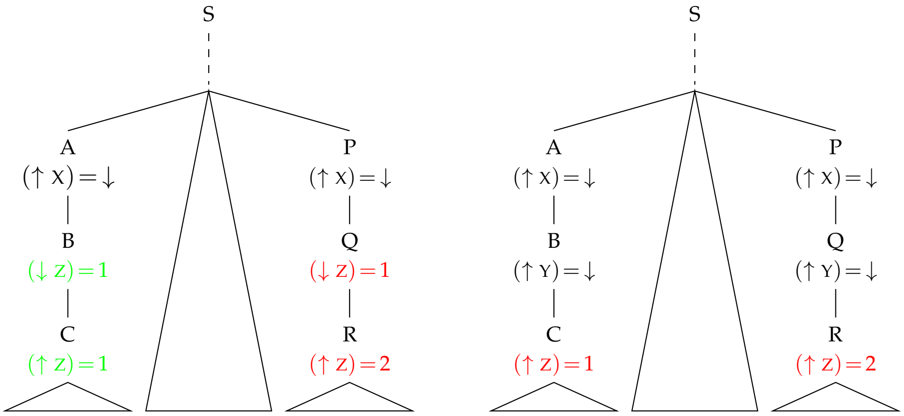


Figure 6: Illustration of the dependencies locally captured by XLE (left) and zipper parsing (right).

In Step 3 we apply to G_s^z the MK algorithms that XLE in essence applies to G_s to determine whether there are any derivations with satisfiable functional descriptions. These algorithms are particularly efficient for disjunctive systems with inconsistencies that are relatively few in number and arise from combinations of nearby constituents. Thus the optimal situation for XLE is illustrated by the schematic derivation on the left side of Figure 6. Here the derivation will fail quickly when the mother-daughter inconsistencies are encountered, and there is no need to evaluate the constituents that make up the large triangle (unless they also belong to an alternative derivation).

The situation illustrated on the right is much less optimal because the inconsistency is not discovered until bottom-up processing reaches the common mother of A and P. Significantly, the entire intermediate subderivation will also be processed before the failure is detected. In contrast, the top-down zipper traversal identifies the A and P subtrees as two branches of the discontinuous X–Y functional unit, as shown earlier in Figure 1. The inconsistency of the C and R annotations becomes apparent when those nodes are brought together by the expansion of the zipped category $\{B, Q\}$ in G_s^z. The failure is discovered immediately, before any computation is wasted in the evaluation of the intermediate subtree.

The improved performance for discontinuous constituents at Step 3 is purchased with the additional expense of the top-down traversal and zipper grammar construction of Step 2. This depends on the degree of discontinuity of G_s, the number of different rules for each specialized category, and the way the annotations of those rules interact when they are combined to form candidate expansions for a zipper set-category. It can be shown that the overall Step 2 effort is bounded by a polynomial in the length of the input.

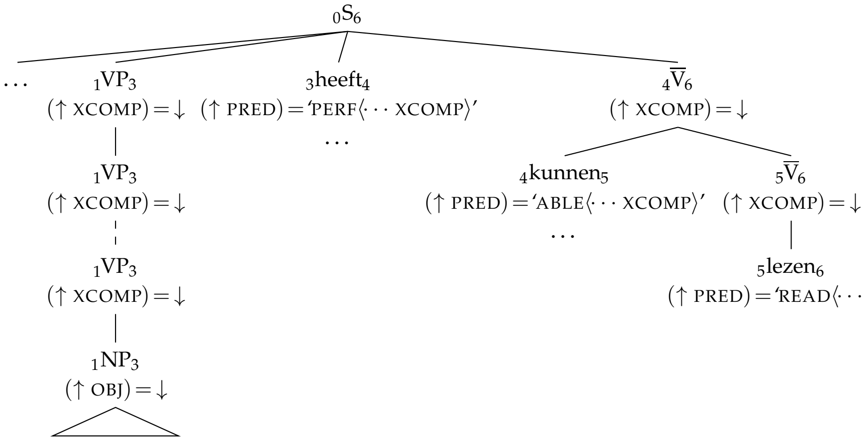


Figure 7: Recursive expansion of the ${}_1\mathrm{VP}_3$ rule.

We close this section with a remark on the nature of the NBD constraint in LFG theory. Formally, this constraint guarantees (for epsilon-free grammars) the decidability of the recognition problem because it bounds as a function of the length of an input string s the number of annotated c-structures and thus the number of f-structures assigned by the specialized grammar G_s. The refined NBD constraint (Kaplan and Maxwell 1996) and Dalrymple (2001) that takes category annotations into account does not eliminate the intended analysis of sentence (5). This is because only one expansion by the recursive rule (11b) is required to match the depth of the governable function OBJ on the left branch with its governing predicate READ on the right. But this rule can apply without limit to produce arbitrarily many nonbranching G_s derivations for this sentence, as indicated by the dashed line in Figure 7. The NBD condition removes those additional derivations from further consideration so that they do not have to be evaluated one by one to discover that the OBJ is ungoverned in each of them.

For this example the NBD condition only suppresses derivations with incoherent and incomplete f-structures, but that would not be the case for longer sentences with more subject-controlled intransitives appearing in the verb sequence on the right. As an example, the sentence (15) is admitted by the Bresnan et al. (1982) grammar.

(15) ... (dat) hij het boek moet hebben kunnen lezen
... (that) he the book must have able read
... (that) he must have been able to read the book

Nonbranching chains would be required to match the level of the OBJ in this and longer sentences with the level of its governing predicate. Unfortunately those derivations would be marked as inadmissible by the NBD condition on G_s, and s would not be accepted as a sentence of G. The problem is that NBD decisions are made separately on each branch of an annotated c-structure with no awareness of relevant information, for example subcategorization requirements, carried by parallel branches. The NBD condition for G (effectively narrowed to a condition on G_s derivations) does not correctly differentiate between all intended and unintended analyses.

We note, however, that the zipper grammar G_s^z in (14) does not contain a nonbranching rule corresponding to the recursive (11b) in G_s. The top-down traversal of the XCOMP zipper components matches each of the specialized VPs on the left with the corresponding predicate on the right, resulting in an internally well-formed zipper rule that necessarily branches. Thus the zipper derivation in Figure 5 has no nonbranching dominance chains. Importantly, by the same reasoning neither would the zipper derivations for sentences with more intransitive verbs: they would be admitted to the language even though all their derivations in G_s violate the NBD condition. These observations lead us to propose a revision to the LFG formalism whereby the original (annotation-insensitive) restriction against nonbranching dominance chains is displaced from derivations of G/G_s to derivations of the zipper grammar G_s^z. This makes a larger set of derivations available for bottom-up validation, but it still guarantees a bounded number of derivations for a given input string and thus the decidability of the recognition problem for arbitrary LFG grammars.

6 Conclusion

In this paper we have explored parsing strategies that follow from the strong equivalence between mildly context-sensitive grammatical formalisms and restricted subclasses of Lexical-Functional Grammar. That connection was first recognized by Seki et al. (1993) and characterized in the definition of finite copying grammars (1), but this key result went largely unnoticed because its notational constraints were so severe. Here we build on the recent work of Wedekind and Kaplan (2020) that demonstrates the same formal equivalence for the subclass of finitely bounded LFG grammars. These are defined with functional annotations and derivational conventions that are much more appropriate for linguistic description.

An LFG grammar that meets all the conditions of finite boundedness can be converted to a linear context-free rewriting system that provides exactly the same f-structures for exactly the same sentences. This enables what we have called the direct LCFRS parsing strategy wherein an LCFRS parser applies the converted grammar to individual input sentences. The LCFRS computation is polynomial in the length of the input and thus tractable in a technical sense. But the computation is likely dominated by another factor that enters into the complexity formula, the size of the converted grammar. While it may be feasible to construct an LCFRS for a finitely-bounded broad-coverage LFG grammars, given natural limits on the parameters of expansion, this may not be the most effective way of parsing with mildly context-sensitive dependencies.

We therefore consider alternative strategies that avoid constructing the LCFRS for all rules and features of an entire grammar and instead only operate on the LFG rules that participate in the context-free analyses for a given input. The specialized grammar is more likely to meet all the bounding conditions even if the entire grammar does not, and the most straightforward approach in that case is to build the LCFRS at parse-time only for the specialized grammar. This approach will typically increase performance because parsing complexity still conforms to the general polynomial formula for LCFRS parsing but with parameters that pertain only to individual inputs and not to the language as a whole. We must revert to conventional parsing algorithms, however, for the (putatively rare) sentences for which the specialized grammar is not bounded.

As another alternative, we propose a more heuristic zipper-driven strategy that incrementally resolves only those mildly context-sensitive zippers that can be identified through a top-down traversal of the rules of that specialized grammar. Zipper-driven parsing is particularly efficient for the majority of inputs with only mildly context-sensitive dependencies because it limits the complexity of subsequent f-structure consistency checking after the polynomial top-down phase is complete. Performance will likely degrade for inputs with more intricate dependencies, but zipper driving offers the benefit of eliminating many candidate f-descriptions before they are subjected to full-scale LFG equation solving.

References

Bar-Hillel, Yehoshua, Perles, Micha and Shamir, Eliahu. 1961. On Formal Properties of Simple Phrase Structure Grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung* 14, 143–172.

Berwick, Robert C. 1982. Computational Complexity and Lexical-Functional Grammar. *American Journal of Computational Linguistics* 8(3–4), 97–109.

Bresnan, Joan. 2001. *Lexical-Functional Syntax*. Malden, MA: Blackwell.

Bresnan, Joan, Kaplan, Ronald M., Peters, Stanley and Zaenen, Annie. 1982. Cross-serial Dependencies in Dutch. *Linguistic Inquiry* 13(4), 613–635.

Crouch, Dick, Dalrymple, Mary, Kaplan, Ronald M., King, Tracy Holloway, Maxwell, III, John T. and Newman, Paula S. 2008. XLE Documentation. http://ling.uni-konstanz.de/pages/xle/doc/xle_toc.html.

Dalrymple, Mary. 2001. *Lexical-Functional Grammar*, volume 34 of *Syntax and Semantics*. New York: Academic Press.

Dalrymple, Mary, Kaplan, Ronald M., Maxwell, III, John T. and Zaenen, Annie (eds.). 1995. *Formal Issues in Lexical-Functional Grammar*. Stanford, CA: CSLI Publications.

Johnson, Mark. 1986. The LFG Treatment of Discontinuity and the Double Infinitive Construction in Dutch. In *Proceedings of the Fifth West Coast Conference on Formal Linguistics*, pages 102–118, Stanford, CA: CSLI Publications.

Kallmeyer, Laura. 2010. *Parsing Beyond Context-Free Grammars*. Berlin: Springer Publishing Company.

Kaplan, Ronald M. and Bresnan, Joan. 1982. Lexical-Functional Grammar: A Formal System for Grammatical Representation. In Joan Bresnan (ed.), *The Mental Representation of Grammatical Relations*, pages 173–281, Cambridge, MA: MIT Press, reprinted in Dalrymple et al. (1995, 98–111).

Kaplan, Ronald M. and Maxwell, III, John T. 1996. *LFG Grammar Writer's Workbench*. Xerox Palo Alto Research Center, Palo Alto, CA.

Kaplan, Ronald M. and Wedekind, Jürgen. 2019. Tractability and Discontinuity. In *Proceedings of the LFG'19 Conference*, pages 130–148, Stanford, CA: CSLI Publications.

Lang, Bernard. 1992. Recognition can be Harder than Parsing. *Computational Intelligence* 10, 486–494.

Maxwell, III, John T. and Kaplan, Ronald M. 1991. A Method for Disjunctive Constraint Satisfaction. In Masaru Tomita (ed.), *Current Issues in Parsing Technology*, pages 173–190, Boston, MA: Kluwer Academic Publishers, reprinted in Dalrymple et al. (1995, 381–401).

Maxwell, III, John T. and Kaplan, Ronald M. 1996. Unification Parsers that Automatically Take Advantage of Context Freeness. In *Proceedings of the LFG'96 Conference*, Stanford, CA: CSLI Publications.

Seki, Hiroyuki, Matsumura, Takashi, Fujii, Mamoru and Kasami, Tadao. 1991. On Multiple Context-free Grammars. *Theoretical Computer Science* 88(2), 191–229.

Seki, Hiroyuki, Nakanishi, Ryuichi, Kaji, Yuichi, Ando, Sachiko and Kasami, Tadao. 1993. Parallel Multiple Context-free Grammars, Finite-state Translation Systems, and Polynomial-time Recognizable Subclasses of Lexical-Functional Grammars. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 130–139, Association for Computational Linguistics, Columbus, OH.

Trautwein, Marten. 1995. The Complexity of Structure Sharing in Unification-Based Grammars. In Walter Daelemans, Gert Durieux and Steven Gillis (eds.), *Computational Linguistics in The Netherlands 1995*, pages 165–179, Antwerp.

Wedekind, Jürgen and Kaplan, Ronald M. 2020. Tractable Lexical-Functional Grammar. *Computational Linguistics* 46(3), 515–569.

Zaenen, Annie and Kaplan, Ronald M. 1995. Formal Devices for Linguistic Generalizations: West Germanic Word Order in LFG. In Jennifer S. Cole, Georigia M. Green and Jerry L. Morgan (eds.), *Linguistics and Computation*, pages 3–27, Stanford, CA: CSLI Publications.