Tractable Lexical-Functional Grammar

Jürgen Wedekind
University of Copenhagen
Department of Nordic Studies
and Linguistics
jwedekind@hum.ku.dk

Ronald M. Kaplan
Stanford University
Linguistics Department
rmkaplan@stanford.edu

The formalism for Lexical-Functional Grammar (LFG) was introduced in the 1980s as one of the first constraint-based grammatical formalisms for natural language. It has led to substantial contributions to the linguistic literature and to the construction of large-scale descriptions of particular languages. Investigations of its mathematical properties have shown that, without further restrictions, the recognition, emptiness, and generation problems are undecidable, and that they are intractable in the worst case even with commonly applied restrictions. However, grammars of real languages appear not to invoke the full expressive power of the formalism, as indicated by the fact that algorithms and implementations for recognition and generation have been developed that run—even for broad-coverage grammars—in typically polynomial time. This article formalizes some restrictions on the notation and its interpretation that are compatible with conventions and principles that have been implicit or informally stated in linguistic theory. We show that LFG grammars that respect these restrictions, while still suitable for the description of natural languages, are equivalent to linear context-free rewriting systems and allow for tractable computation.

1. Introduction

Grammar formalisms are attractive for modeling and processing natural language if they are expressive enough to describe the structure of natural languages and if they are computationally tractable. While the first requirement is certainly satisfied by the majority of unification-based grammar formalisms, the second is certainly not. Computational performance is minimally identified with the complexity of the universal recognition/parsing problem—that is, the problem of determining for any given grammar G and terminal string s whether G assigns a feature structure to s. Because a problem is computationally tractable if it has a polynomial-time solution, formalisms are thus considered tractable if their recognition/parsing problem can be solved in polynomial time.

Submission received: 21 December 2018; revised version received: 3 November 2019; accepted for publication: 27 June 2020.

https://doi.org/10.1162/COLI_a_00384

For almost all unrestricted unification-based grammar formalisms, the recognition problem has been known to be undecidable since the earliest days of unification grammar (see, e.g., Kaplan and Bresnan 1982; Johnson 1988; Blackburn and Spaan 1993). To sidestep this undecidability issue in the design of Lexical-Functional Grammar (LFG), Kaplan and Bresnan (1982) introduced a constraint, later called the Off-line Parsability Constraint, that guarantees decidability of the recognition problem. The Off-line Parsability Constraint proscribes empty productions and nonbranching dominance chains and thus bounds the number and size of the c-structures of a string by a function of the length of that string. This works for recognition/parsing because the size of the set of constraints on each f(eature)-structure (the f-description) of a string is always bounded by the size of its c-structures. The parsing problem is decidable with off-line parsability, but that constraint is not sufficient to ensure tractability: The parsing problem for off-line parsable LFG grammars is known to be NP-complete (Berwick 1982).

The universal generation problem is another problem of practical importance. This is the problem of determining for an arbitrary grammar G and an arbitrary f-structure F whether G derives any terminal string with F. This has been shown to be undecidable even for grammars that are off-line parsable (Wedekind 2014).[1] Note further that the emptiness problem (the problem of determining for an arbitrary grammar G whether $L(G) = \varnothing$) is also undecidable for off-line parsable grammars.[2] This problem is of formal interest but perhaps has less practical significance.

A number of variants of the Off-line Parsability Constraint have been proposed since Kaplan and Bresnan's (1982) original formulation (see, e.g., Jaeger et al. [2005] for a survey). However, none of them is particularly effective in improving the computational properties of LFG and other unification-based formalisms. We therefore here ignore Off-line Parsability constraints, even though some of them may help in increasing efficiency and may also be required for linguistic interpretation: They are not sufficient for attaining recognition tractability and decidability of the generation and emptiness problems and they are not necessary given the solution that this article identifies.

A separate line of research has examined whether there are restrictions on the form of LFG rule annotations that might reduce the complexity class of the formalism to a system with tractable recognition and generation algorithms. In particular, Seki et al. (1993) consider LFG grammars where each right-hand side category is annotated with exactly one function-assigning schema of the form $(\uparrow \text{F}) = \downarrow$ and finitely many atomic-valued annotations of the form $(\uparrow \text{A}) = v$. They show that the class of languages generated by these LFG grammars is equal to that generated by tree-to-string finite-state translation systems. Because the recognition problem for tree-to-string finite-state translation systems is still NP-complete, Seki et al. (1993) considered in addition the condition that the number of nodes that relate to the same f-structure element is finitely bounded. This restriction limits the number of dependent c-structure paths and finally ensures that the resulting "finite-copying" LFG grammars are mildly context-sensitive and hence tractable. Vijay-Shanker et al. (1987) and Weir (1988) identified several representatives of this class, among them linear context-free rewriting systems (LCFRSs). These have subsequently been shown to be equivalent to multiple context-free grammars (Seki et al.

1 LFG's generation problem is only undecidable for cyclic f-structures. If only acyclic f-structures are considered the problem is decidable even for grammars that are not off-line parsable (Wedekind and Kaplan 2012).

2 Wedekind (1999) provides a proof for off-line parsable grammars; see Roach (1983) and Nishino (1991) for earlier undecidability proofs of this problem.

1991) and simple range concatenation grammars (Boullier 2000) (see also Kallmeyer [2010b] for a useful survey of mildly context-sensitive grammar formalisms).

These tractability-motivated restrictions on the formalism are quite severe and seem to exclude the possibility of expressing some of the key principles and devices of LFG theory. For example, among the common annotations that these restrictions disallow are the trivial $\uparrow = \downarrow$ equations that mark the heads of constituents, multi-attribute value specifications, such as $(\uparrow \text{SUBJ NUM}) = \text{SG}$, that encode agreement requirements, and reentrancy equations, such as $(\uparrow \text{XCOMP SUBJ}) = (\uparrow \text{SUBJ})$, that represent functional control. However, these restrictions provide a useful starting point for identifying other formal properties that guarantee tractability but still allow linguistic generalizations to be stated perspicuously. Natural language grammars exploit more of the LFG formalism than the Seki et al. analysis would allow, but they do not make arbitrary use of its expressive power.

The computational advantage of these previous approaches depended on severely limiting the form of functional annotations and how they are distributed across the rules of a grammar. Here, we relax the restrictions on notation to allow a broader range of annotations as are more typically used in linguistic descriptions. We show that tractability and decidability can still be established with this linguistically more suitable notation if the propagation of atomic-value information across a derivation is regulated in other ways. We characterize a subclass of LFG grammars with properties that guarantee limitations on the flow of information and yet also seem compatible with the way that the LFG formalism is deployed in linguistic practice. In this article we focus on the descriptive devices as originally proposed by Kaplan and Bresnan (1982) and still in common use, leaving to later work the more sophisticated extensions to the formalism (e.g., functional uncertainty) that were later introduced and are now in common use. The Kaplan-Bresnan formalism is still the core of the theory and resolving its computational issues is a prerequisite for the future analysis of any additional mechanisms.

The organization of this article is as follows. In the next section we define a primitive subclass of LFG grammars, the basic LFGs, and characterize their derivations and languages. Basic LFGs are stripped down versions of grammars in the Kaplan and Bresnan (1982) formalism that retain the essential equational mechanism of this and other unification-based formalisms. They provide a simplified setting for establishing the properties crucial for controlling the flow of atomic-value information; the additional Kaplan-Bresnan formal devices are simple augmentations of this basic machinery. In Section 3 we introduce a subclass of basic LFG grammars that allow for trivial annotations and reentrancies of a particularly limited form. These are required for linguistic description but have unacceptable formal and computational properties. We therefore review the notational and derivational restrictions of the finite-copying subclass (Seki et al. 1993), which is known to be LCFRS equivalent and computationally tractable, and we argue in Section 4 that a Seki et al.-style boundedness condition must be paired with a nonconstructivity condition on the reentrancies to achieve LCFRS equivalence and thus to preserve the key advantages of Seki et al.'s strictly less expressive formalism. We claim that these restrictions are compatible with linguistic description and show their decidability. We then provide in Section 5 a constructive proof of the equivalence between grammars in this linguistically relevant subclass and LCFRSs, and we demonstrate that the LCFRS equivalence extends to LFG grammars that make use of the richer set of descriptive devices originally proposed by Kaplan and Bresnan (1982). Section 6 highlights the beneficial effects on computational performance of a broader set of LFG

principles and explores possible implementation difficulties and alternative parsing strategies. The last section summarizes our results and provides a brief discussion of open issues.

2. Preliminaries

We start with a formal characterization of LFG grammars with only equational statements and without any parsability constraints. These are called basic LFGs. We define the notation for this broad class of grammars and specify how that notation is interpreted in derivations that establish the correspondence between strings, functional descriptions, and functional structures. Let Σ^* denote the set of all finite strings over some finite set of symbols Σ and $\Sigma^+ = \Sigma^* \backslash \{\epsilon\}$, with ϵ denoting the empty string. Then a basic LFG grammar G is defined as follows:

Definition 1
A **basic LFG grammar** G over a finite set of attributes $\mathcal{A}$ and values $\mathcal{V}$ is a 4-tuple (N, T, S, R) where N is a finite set of nonterminal categories, T is a finite set of terminal symbols, $S \in N$ is the root category, and R is a finite set of rules that includes annotated productions of the form

$$\begin{array}{lllll} A \rightarrow & Y_1 & .. & Y_m & \qquad A \rightarrow \epsilon \\ & D_1 & & D_m & \qquad\quad D_1 \end{array}$$

with $A \in N$, $Y_j \in (N \cup T)$ $(j = 1, .., m)$, and $m \geq 1$. Each annotated description D_j is a (possibly empty) finite set of equalities between expressions of the form $(\uparrow\ \sigma)$, $(\downarrow\ \sigma)$ or v where v is a value of $\mathcal{V}$ and σ is a possibly empty sequence of attributes of $\mathcal{A}$. When σ is empty, $(\uparrow\ \sigma)$, $(\downarrow\ \sigma)$ are equivalent to $\uparrow$ and $\downarrow$, respectively.

A basic LFG assigns to each sentence in its language at least one constituent structure (c-structure) and at least one f-structure. The f-structure is the minimal solution for a set of instantiated annotations (f-description) that is obtained by instantiating the annotations of rules that license the local mother–daughter configurations of the c-structure. For each such rule, all occurrences of the $\uparrow$ symbol (called a **metavariable**) in the annotations of the daughters are replaced by the mother node, and for each of the daughter categories, all occurrences of the $\downarrow$ metavariable in its annotations are replaced by the corresponding daughter node. This instantiation procedure uses the nodes themselves instead of the related f-structure variables of Kaplan and Bresnan (1982) or the more explicit ϕ projection of the LFG correspondence architecture (Kaplan 1995). (Node instantiation is mathematically equivalent to the other representations but is better suited to present purposes.) The f-structure is the unique minimal solution (minimal model) of the f-description.

To be more explicit, instantiated descriptions are obtained from the rules by substituting for the $\uparrow$ and $\downarrow$ metavariables elements drawn from a collection of elements. C-structure nodes are included in the collection, but later on we also make use of other elements. We define a function *Inst* that assigns to each m-ary rule r, element b, and sequence of elements β of length m the instantiated description that is obtained from the annotations of r by substituting b for $\uparrow$ and β_j for $\downarrow$ in the annotations of all $j = 1, .., m$ daughters. In the definition below we use the (more compact) linear rule notation $A \rightarrow (X_1, D_1)..(X_m, D_m)$ that we prefer in more formal specifications. (Using this notation, epsilon rules are defined as unary rules ($m = 1$) with $X_1 = \epsilon$.) Moreover,

for a set of formulas D we let $D[c_1/b_1, .., c_n/b_n]$ denote the set of formulas obtained from D by simultaneously replacing all occurrences of c_i by the corresponding element b_i, for $i = 1, .., n$.

Definition 2
Let r be an m-ary LFG rule $A \rightarrow (X_1, D_1)..(X_m, D_m)$ and (b, β) be a pair of an element and a sequence of elements of length m. Then the **instantiated description** that results from r and (b, β) is given by

$$Inst(r, (b, \beta)) = \bigcup_{j=1}^{m} D_j[\uparrow/b, \downarrow/\beta_j].$$

Before we present the usual definition of LFG derivations, we first define derivations of strings and their instantiated functional descriptions without requiring the f-descriptions to be clash-free (consistent), that is, without requiring that there be an associated f-structure. These **unevaluated** derivations are useful for our purposes because we can exploit certain properties of the f-descriptions even for grammar classes whose recognition, emptiness, and generation problems are known to be undecidable. In the definitions, we do not require the label of the root node to be S nor the yield to be a terminal string. This generality is needed for conducting inductive proofs both top–down and bottom–up. We further assume that *root* is the root node of any c-structure c, and that *dts* is a function that assigns to each nonterminal node n of c the sequence of its immediate daughters.

Definition 3
A pair (c, ρ) consisting of a labeled tree c and a mapping ρ from the nonterminal nodes of c into R is a(n unevaluated) **derivation** of string $s \in (N \cup T)^*$ from B **with functional description** FD in G iff

(i) the label of *root* is B,

(ii) the yield is s,

(iii) for each nonterminal node n with label A and $dts(n) = n_1..n_m$ with labels $X_1, .., X_m$, respectively, $\rho_n = A \rightarrow (X_1, D_1)..(X_m, D_m)$,

(iv) $FD = \bigcup_{n \in Dom(\rho)} Inst(\rho_n, (n, dts(n)))$.

Definition 4
A functional description FD is **clash-free** iff

(i) $FD \not\vdash a = v$, where $v \in \mathcal{V}$ and a is any other constant (atomic feature value or node) occurring in FD,

(ii) $FD \not\vdash t = (v\ \sigma)$, where $v \in \mathcal{V}$, $\sigma \in \mathcal{A}^+$, and t is an arbitrary term.

These conditions are syntactic versions of the constant/constant and constant/complex clash conditions that together capture LFG's functional uniqueness condition (the denotations of an atomic feature value and any other distinct atomic feature value or node constant have to be distinct (i); atomic feature values have no attributes (ii)).

There are well-known algorithms for deciding whether a given description is clash-free (e.g., Nelson and Oppen 1980).

LFG derivations are then defined as follows.

Definition 5
A pair (c, ρ) consisting of a labeled tree c and a mapping ρ from the nonterminal nodes of c into R is a **derivation** of string $s \in (N \cup T)^*$ from B **with functional description** FD **and f-structure** F in G iff

(i) (c, ρ) is a derivation of s from B with f-description FD in G,

(ii) FD is clash-free,

(iii) F is isomorphic to $M|_{\mathcal{A}\cup\mathcal{V}}$, the restriction to $\mathcal{A} \cup \mathcal{V}$ of a minimal model M of FD.

The effect of condition (iii) is to abstract away from the nodes and their interpretation, and the universe of the given minimal model that have no linguistic significance. In contrast to the unevaluated derivations of Definition 3, we refer to derivations as defined in Definition 5 sometimes as **valid** derivations, although it will always be clear from the context which of the two notions is meant.

Finally, we define LFG's derivability relation Δ and the language of an LFG grammar as follows.

Definition 6
A terminal string s is **derivable** with f-structure F in G ($\Delta_G(s, F)$) iff there is a derivation of s from S with F (and some f-description FD) in G. The language of G is the set

$$L(G) = \{s \in T^* \mid \exists F(\Delta_G(s, F))\}.$$

As an example consider the LFG grammar G with the productions given in (1).

(1) a. $\mathrm{S} \rightarrow \underset{(\uparrow \mathrm{S}) = \downarrow}{\mathrm{NP}} \quad \underset{\uparrow = \downarrow}{\mathrm{VP}}$

b. $\mathrm{VP} \rightarrow \underset{(\uparrow \mathrm{O}) = \downarrow}{\mathrm{NP}} \quad \underset{(\uparrow \mathrm{X}) = \downarrow}{\overline{\mathrm{VP}}} \quad \underset{\uparrow = \downarrow}{\mathrm{V}}$

c. $\overline{\mathrm{VP}} \rightarrow \underset{(\uparrow \mathrm{O}) = \downarrow}{\mathrm{NP}} \quad \underset{(\uparrow \mathrm{X}) = \downarrow}{\overline{\mathrm{VP}}}$

d. $\overline{\mathrm{VP}} \rightarrow \begin{array}{c}\mathrm{NP} \\ (\uparrow \mathrm{O}) = \downarrow \\ (\uparrow \mathrm{X}\,\mathrm{X}) = \#\end{array}$

e. $\mathrm{V} \rightarrow \underset{\uparrow = \downarrow}{\mathrm{V}'} \quad \underset{(\uparrow \mathrm{X}) = \downarrow}{\overline{\mathrm{V}}}$

f. $\overline{\mathrm{V}} \rightarrow \underset{\uparrow = \downarrow}{\mathrm{V}''} \quad \underset{(\uparrow \mathrm{X}) = \downarrow}{\overline{\mathrm{V}}}$

g. $\mathrm{NP} \rightarrow \begin{array}{c}\mathrm{n} \\ (\uparrow \mathrm{P}) = \mathrm{N} \\ (\uparrow \mathrm{A}) = \mathrm{C}\end{array}$

h. $\mathrm{V}' \rightarrow \begin{array}{c}\mathrm{v} \\ (\uparrow \mathrm{P}) = \mathrm{V} \\ (\uparrow \mathrm{X}\,\mathrm{S}) = (\uparrow \mathrm{O}) \\ (\uparrow \mathrm{S}\,\mathrm{A}) = \mathrm{C}\end{array}$

i. $\mathrm{V}'' \rightarrow \begin{array}{c}\mathrm{v} \\ (\uparrow \mathrm{P}) = \mathrm{V} \\ (\uparrow \mathrm{X}\,\mathrm{S}) = (\uparrow \mathrm{O})\end{array}$

j. $\overline{\mathrm{V}} \rightarrow \begin{array}{c}\mathrm{v} \\ (\uparrow \mathrm{P}) = \mathrm{V} \\ (\uparrow \mathrm{X}) = \#\end{array}$

This grammar derives the language $\{\mathrm{n}^n\,\mathrm{v}^n \mid n > 2\}$. It mimics the LFG grammar for Dutch cross-serial dependencies presented in Bresnan et al. (1982) in that it produces for a string $\mathrm{n}^n\,\mathrm{v}^n$ an f-structure that is structurally equivalent to the f-structure assigned to a Dutch c-structure with preterminal string $\mathrm{NP}^n\,\mathrm{V}^n$. The attributes S, P, O, and X correspond to SUBJ, PRED, OBJ, and VCOMP, respectively, of the original grammar, and

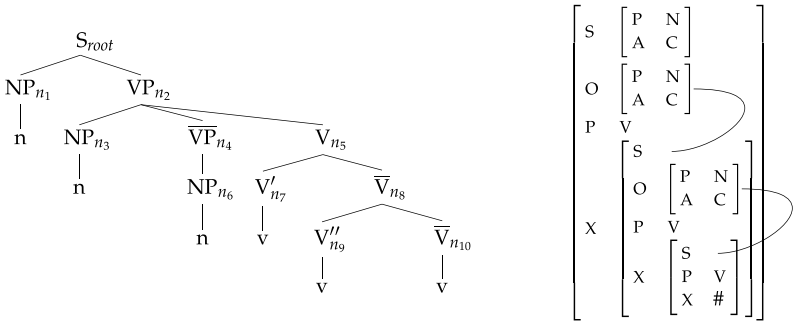


Figure 1
The c- and f-structure assigned to the terminal string nnnvvv by the LFG grammar with the rules in (1).

A is a generic subject–verb agreement feature. However, in contrast to Bresnan et al.'s grammar, we use a more fine-grained category system and a boundary marker # to ensure that the different v's are derived at the right positions and that the number of n's and v's properly match. This is just because the basic formalism does not include constraints and devices for enforcing the valency requirements of the individual predicates, the Completeness and Coherence Conditions, that usually account for that. Finally, for simplicity, we do not maintain the strict distinction between phrasal and lexical categories.

The terminal string nnnvvv, for example, is well-formed (nnnvvv $\in L(G)$), because it is derivable with the c- and f-structure depicted in Figure 1. The nonterminal nodes of the c-structure tree are explicitly specified since they are used for the instantiation of the rule annotations. The rule mapping ρ that justifies the c-structure is given in (2).

(2) $\rho_{root} = (1a)$, $\rho_{n_1} = \rho_{n_3} = \rho_{n_6} = (1g)$, $\rho_{n_2} = (1b)$, $\rho_{n_4} = (1d)$, $\rho_{n_5} = (1e)$, $\rho_{n_7} = (1h)$, $\rho_{n_8} = (1f)$, $\rho_{n_9} = (1i)$, $\rho_{n_{10}} = (1j)$

The f-structure of Figure 1 is associated with the given c-structure because it is obtained from the minimal model (3b) of the f-description (3a) by restricting the interpretation function to the attributes and atomic feature values in $\mathcal{A} \cup \mathcal{V}$, thus disregarding the nodes and their interpretation.

(3) a.
$$\left\{\begin{array}{c}
(root\ \textsc{s}) = n_1, root = n_2,\\
(n_1\ \textsc{p}) = \textsc{n}, (n_1\ \textsc{a}) = \textsc{c},\\
(n_2\ \textsc{o}) = n_3, (n_2\ \textsc{x}) = n_4, n_2 = n_5,\\
(n_3\ \textsc{p}) = \textsc{n}, (n_3\ \textsc{a}) = \textsc{c},\\
(n_4\ \textsc{o}) = n_6, (n_4\ \textsc{x}\ \textsc{x}) = \#,\\
n_5 = n_7, (n_5\ \textsc{x}) = n_8,\\
(n_6\ \textsc{p}) = \textsc{n}, (n_6\ \textsc{a}) = \textsc{c},\\
(n_7\ \textsc{p}) = \textsc{v}, (n_7\ \textsc{x}\ \textsc{s}) = (n_7\ \textsc{o}), (n_7\ \textsc{s}\ \textsc{a}) = \textsc{c},\\
n_8 = n_9, (n_8\ \textsc{x}) = n_{10},\\
(n_9\ \textsc{p}) = \textsc{v}, (n_9\ \textsc{x}\ \textsc{s}) = (n_9\ \textsc{o}),\\
(n_{10}\ \textsc{p}) = \textsc{v}, (n_{10}\ \textsc{x}) = \#
\end{array}\right\}$$

b.
$$\begin{array}{l}
root \\ n_2 \\ n_5 \\ n_7
\end{array}\left[\begin{array}{lll}
\textsc{s} & n_1\left[\begin{array}{ll}\textsc{p} & \textsc{n}\\ \textsc{a} & \textsc{c}\end{array}\right] & \\
\textsc{o} & n_3\left[\begin{array}{ll}\textsc{p} & \textsc{n}\\ \textsc{a} & \textsc{c}\end{array}\right] & \\
\textsc{p} & \textsc{v} & \\
 & \begin{array}{l} n_4 \\ n_8 \\ n_9 \end{array}\left[\begin{array}{ll}\textsc{s} & \\ \textsc{o} & n_6\left[\begin{array}{ll}\textsc{p} & \textsc{n}\\ \textsc{a} & \textsc{c}\end{array}\right]\\ \textsc{p} & \textsc{v}\end{array}\right] & \\
\textsc{x} & & \\
 & \left[\begin{array}{ll} & n_{10}\left[\begin{array}{lll}\textsc{s} & & \\ \textsc{p} & \textsc{v} & \\ \textsc{x} & \# & \end{array}\right]\\ \textsc{x} & \end{array}\right] &
\end{array}\right]$$

3. Linguistically Motivated Annotations and Finite-Copying LFGs

In this section we consider a proper subclass of basic LFGs that provides the primitive notation needed for linguistic analysis and we demonstrate that even for these notationally more restricted grammars the core computational problems are undecidable. We then introduce Seki et al.'s finite-copying grammars. These grammars are considerably more restricted in notation and their derivations are crucially required to satisfy a strong bounding condition. As indicated, finite-copying grammars admit of decidable and tractable solutions to key computational problems, but their limited expressivity makes them unsuited for theoretically motivated descriptions of natural languages.

3.1 Linguistically Motivated Annotations

The LFG grammars that we will consider in the following are characterized by short reentrancies with no more than two attributes. This respects the theoretical Principle of Functional Locality, the stipulation that designators in lexical and grammatical annotations can specify no more than two grammatical functions (Kaplan and Bresnan 1982, page 278).

Definition 7
An **annotated reentrancy** is an equation (or a symmetric variant) of one of the following forms:

$(\uparrow \text{F G}) = (\uparrow \text{H})$
$(\uparrow \text{F}) = (\uparrow \text{H})$
$(\downarrow \text{G}) = (\uparrow \text{H})$
$(\downarrow \text{F}) = (\downarrow \text{H})$
$(\downarrow \text{G}) = \uparrow$
$\uparrow = (\uparrow \text{H})$
$(\downarrow \text{G}) = \downarrow$

An **instantiated reentrancy** is the result of substituting nodes for the metavariables in an annotated reentrancy.

Long reentrancies (with a two-attribute designator) may appear on terminal and nonterminal categories and are typically used for functional control (e.g., $(\uparrow \text{XCOMP SUBJ}) = (\uparrow \text{SUBJ})$).[3] All other reentrancies contain at most one attribute on each side. Reentrancies of the form $(\uparrow \text{F}) = (\uparrow \text{H})$ might be used for clause-internal topicalization (e.g., $(\uparrow \text{TOPIC}) = (\uparrow \text{OBJ})$) and those of the form $(\downarrow \text{G}) = (\uparrow \text{H})$ for subject control of open adjuncts (XADJUNCTS) $((\downarrow \text{SUBJ}) = (\uparrow \text{SUBJ}))$. The remaining reentrancies just cover other possibilities for equating the $\uparrow/\downarrow$ metavariables with a one-attribute designator. The last two introduce immediate cycles and are not commonly attested in natural language grammars, but they introduce no additional expressive power or formal complexity since their effect can be achieved by combinations of other reentrancies

3 Long reentrancies can also be used for agreement of atomic values (e.g., $(\uparrow \text{SUBJ NUM}) = (\uparrow \text{NUM})$). However, because there is a finite bound on the number of agreement features, agreement can always be equivalently encoded through collections of atomic-valued annotations and thus requires no special consideration.

on the list. We also admit atomic-valued annotations of the form $(\uparrow\ \sigma) = v$ or $(\downarrow\ \sigma) = v$, with $|\sigma| > 0$, and we allow each nonterminal to be annotated either by $\uparrow = \downarrow$ or exactly one function-assigning annotation of the form $(\uparrow\ \textsc{f}) = \downarrow$ or none of them. The precise definition of the class of LFG grammars with linguistically motivated rule annotations is given below.

Definition 8
A basic LFG $G = (N, T, S, R)$ is **suitably annotated** if every annotated right-hand side rule category (X, D) satisfies the following conditions:

(a) if $X \in N$, then D may contain at most one annotation of the form $(\uparrow\ \textsc{f}) = \downarrow$ or $\uparrow = \downarrow$, any number of annotated reentrancies, and any number of atomic-value annotations of the form $(\uparrow\ \sigma) = v$ or $(\downarrow\ \sigma) = v$, with $|\sigma| > 0$,

(b) if $X \in T$ or $X = \epsilon$, then D may include annotated reentrancies not containing $\downarrow$ and atomic-value annotations of the form $(\uparrow\ \sigma) = v$, with $|\sigma| > 0$.

3.2 Undecidable Problems for Suitably Annotated LFGs

As previously noted, the emptiness, parsing, and generation problems are undecidable for unrestricted basic LFG grammars. Without further limitations these problems are also undecidable for suitably annotated LFGs. Wedekind (1999) and Wedekind (2014) provided simple proofs of these results for a slightly less restricted formalism using a reduction from the emptiness problem for the intersection of context-free languages. We adapt this proof strategy to the suitable notation in order to highlight the sources of computational complexity that reentrancies introduce and that must be controlled to ensure LCFRS equivalence.

Following Wedekind (1999) we demonstrate the undecidability of the emptiness problem by constructing for each context-free grammar $G = (N, T, S, R)$ in Chomsky normal form a suitably annotated LFG grammar $String(G)$ such that $L(String(G)) = L(G)$ and the f-structure for each derivable string s contains a simple encoding of s and a representation of a derivation in G that leads to that terminal string. For each terminal rule $A \to a$ in R the rule set of the string grammar contains a rule of the form (4a), and for each nonterminal rule $A \to C\,D$ it contains a rule of the form (4b).

(4) a. $A \to \quad a$
$\qquad\qquad (\uparrow\ \textsc{b}\ \textsc{f}) = a$
$\qquad\qquad (\uparrow\ \textsc{b}\ \textsc{t}) = (\uparrow\ \textsc{e})$

b. $A \to \quad C \qquad\qquad D$
$\qquad\qquad (\downarrow\ \textsc{b}) = (\uparrow\ \textsc{b}) \quad (\downarrow\ \textsc{e}) = (\uparrow\ \textsc{e})$
$\qquad\qquad (\downarrow\ \textsc{e}) = (\uparrow\ \textsc{m}) \quad (\downarrow\ \textsc{b}) = (\uparrow\ \textsc{m})$

The attributes B, E, M, F, and T are mnemonic for "begin", "end", "middle", "first", and "tail", respectively. Derivations are threaded through the B(egin), M(iddle), and E(nd) attributes and the F(irst) and T(ail) attributes encode linked-list representations of terminal strings. As illustrated by the c-structure/f-structure pair in Figure 2, an atomic F value corresponds to a terminal at a given string position and the T value sequence represents the suffix of the string from that position. The yield of each context-free constituent is encoded by the B(egin) and E(nd) attributes of the corresponding f-structure, and the path between the *root* B and E values represents the entire string. The M(iddle) attribute in (4b) is an auxiliary used to link the daughter yields of binary constituents.

Let $G_1 = (N_1, T_1, S_1, R_1)$ and $G_2 = (N_2, T_2, S_2, R_2)$ be two arbitrary context-free grammars in Chomsky normal form with $N_1 \cap N_2 = \varnothing$. Construct an LFG grammar

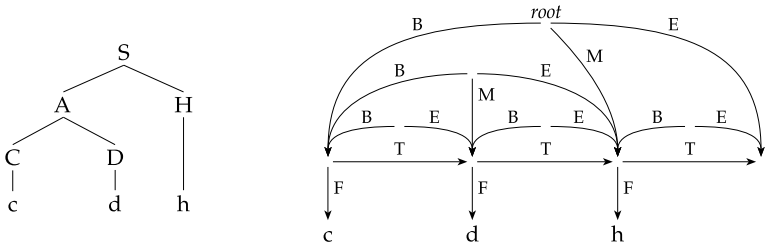


Figure 2
A sample c- and f-structure derived with string grammar rules of the form (4).

$G = (N, T, \mathrm{S}, R)$ with $N = N_1 \cup N_2 \cup \{\mathrm{S}\}$, $\mathrm{S} \notin N_1 \cup N_2$, and $T = T_1 \cup T_2$. R consists of the rules of $String(G_1)$ and $String(G_2)$ and the following start rule:

(5) $\mathrm{S} \rightarrow$ S_1 S_2
$(\downarrow \mathrm{B}) = (\uparrow \mathrm{B})$ $(\downarrow \mathrm{B}) = (\uparrow \mathrm{B})$
$(\downarrow \mathrm{E\,F}) = \#$ $(\downarrow \mathrm{E\,F}) = \#$

Then for any valid derivation of a terminal string s in G, s has the form $s_1 s_2$, with s_1 derived from S_1 and s_2 derived from S_2, and $s_1 = s_2$, because both string encodings are unified by the S rule and the instantiations of $(\uparrow \mathrm{E\,F}) = \#$ ensure that one string is not a proper prefix of the other. Thus $L(G) = \varnothing$ iff $L(G_1) \cap L(G_2) = \varnothing$.

The undecidability of the parsing/recognition problem follows from a simple modification to the string-grammar specification and, again, the emptiness problem for context-free intersection. If we construct for each terminal rule $A \rightarrow a$ two rules of the form

(6) $A \rightarrow$ A^a $A^a \rightarrow \epsilon$
$(\uparrow \mathrm{B\,F}) = a$
$(\uparrow \mathrm{B\,T}) = (\uparrow \mathrm{E})$

where A^a is a new nonterminal, for each terminal a, then $\epsilon \in L(G)$ if and only if $L(G_1) \cap L(G_2) \neq \varnothing$.[4]

String grammars defined with rules (4–6) contain short reentrancies on the nonterminals and long reentrancies on the terminals or preterminals but no annotated function assignments or trivial annotations. This demonstrates that complexity can arise from just the interaction of structure sharing constraints on f-structure features without the support of node equalities.

An alternative formulation of $String(G)$ shows, on the other hand, that combining just one kind of reentrancy, the short ones, with function assignments can lead to the same undecidability results. Long reentrancies only appear in rules of the form (4a).

4 Empty nodes are disallowed in some modern versions of LFG, particularly when long-distance dependencies are characterized by functional uncertainty rather than traces (Kaplan and Zaenen 1989; Dalrymple et al. 2015). Unless the theory retains some variant of the original Off-line Parsability Constraint, the undecidability of the recognition problem can be demonstrated even in the absence of empty nodes.

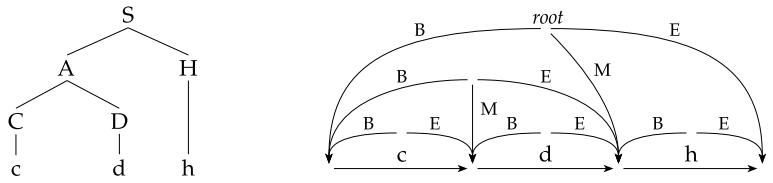


Figure 3
A sample c- and f-structure obtained from a string grammar that encodes the derived strings in the f-structure as descending chains of attributes.

They can be eliminated in favor of function assignments by using for each terminal rule $A \rightarrow a$ two rules of the form (7a,b) (emptiness problem) or (7a,c) (recognition problem)

(7) a. $A \rightarrow \quad A^a$
$(\uparrow \text{B}) = \downarrow$
$(\uparrow \text{B F}) = a$
$(\downarrow \text{T}) = (\uparrow \text{E})$

b. $A^a \rightarrow a$

c. $A^a \rightarrow \epsilon$

where A^a is a new nonterminal, for each terminal a.

The terminal strings can also be encoded as descending chains of attributes rather than as first-rest sequences of atomic values.[5] A sample c- and f-structure are shown in Figure 3. Because such string grammars do not contain atomic-valued annotations, they provide a powerful proof-theoretic tool for establishing that certain more specific questions about the interaction of function assignments and reentrancies remain undecidable even in the absence of the filtering effect of atomic-valued annotations.

We obtain such string grammars by combining nonterminal rules of the form (4b) with terminal rules of the form (8).

(8) $A \rightarrow \quad a$
$(\uparrow \text{B } a) = (\uparrow \text{E})$

If the revised string grammars are combined with the start rule (9), then any problem whose solution can be made dependent on the derivability of node equalities can be shown to be undecidable for suitably annotated LFGs without atomic-valued annotations.

(9) $\text{S} \rightarrow \quad \text{S}_1 \quad \text{S}_2 \quad \text{X} \quad \text{Y}$
$(\downarrow \text{B}) = (\uparrow \text{B}) \quad (\downarrow \text{B}) = (\uparrow \text{B}) \quad (\uparrow \text{E}_1) = \downarrow \quad (\uparrow \text{E}_2) = \downarrow$
$(\downarrow \text{E}) = (\uparrow \text{E}_1) \quad (\downarrow \text{E}) = (\uparrow \text{E}_2)$

By construction, the function assignments on X and Y will result in a node equality only if the instantiations of the two $(\downarrow \text{E})$ designators have the same value. This happens only if S_1 and S_2 derive the same string s ($s \in L(G_1) \cap L(G_2)$). Thus, if we add epsilon rules for X and Y then this construction shows that it is in general undecidable whether there are derivations with co-referring nodes in such grammars.

5 Such an encoding has been used in the undecidability proof of the generation problem (from cyclic f-structures) in Wedekind (2014). This proof depends on annotations that lie outside of the suitable notation, but the argument can easily be reformulated for suitably annotated LFGs.

If we construct for each terminal rule $A \rightarrow a$ string grammar rules with only short reentrancies as in (10),

(10) $A \rightarrow \begin{array}{c} A^a \\ (\uparrow \text{B}) = \downarrow \\ (\downarrow a) = (\uparrow \text{E}) \end{array} \qquad A^a \rightarrow a$

then the more elaborated X and Y expansions in (11) can be used to show that, even in absence of clashes, it is undecidable whether there are derivations in which all long reentrancies are equivalent to a combination of function assignments and short reentrancies.

(11) $\text{X} \rightarrow \begin{array}{c} \text{Z} \\ (\uparrow \text{F G}) = (\uparrow \text{H}) \end{array} \qquad \text{Y} \rightarrow \begin{array}{c} \text{Z} \\ (\uparrow \text{F}) = \downarrow \end{array}$

The undecidability of these different configurations of annotations is an important diagnostic in our attempt at arriving at LCFRS equivalence: The emptiness and recognition problems are decidable for LCFRS, so further restrictions must be applied to any configuration where these problems are undecidable. For our further considerations it is therefore important to note that the use of short reentrancies must be restricted if they are combined with function assignments, and that the use of long reentrancies must be severely restricted.[6] This is because it is even undecidable whether they reduce to short ones.

3.3 Finite-Copying LFGs

The finite-copying grammars of Seki et al. form a subclass of the suitably annotated LFGs that are known to have a tractable recognition and a decidable emptiness problem. These results are obtained by limiting the notation for rule annotations to include only direct function assignments of the form $(\uparrow \text{F}) = \downarrow$ and atomic-value annotations of the form $(\uparrow \text{A}) = v$. In addition the derivations of a finite-copying grammar must respect a global boundedness restriction. Finite-copying LFGs are effectively defined as follows.

Definition 9
A suitably annotated LFG $G = (N, T, \text{S}, R)$ is a **finite-copying LFG** iff

(i) every rule $A \rightarrow (X_1, D_1)..(X_m, D_m)$ in R satisfies the following conditions:

(a) if $X_j \in N$, then D_j may contain at most one annotation of the form $(\uparrow \text{F}) = \downarrow$ and any number of annotations of the form $(\uparrow \text{A}) = v$,

(b) if $X_j \in T$, then D_j may only contain annotations of the form $(\uparrow \text{A}) = v$,

6 Feinstein and Wintner (2008) show that unification grammars are equivalent to linear indexed grammars (LIGs) if, for each rule, the mother feature structure is required to share at most one structure with the daughter feature structures. As indicated by the proper inclusion of LIGs in LCFRSs, the use of reentrancies is overly restricted in this formalism.

(ii) for every valid derivation (c, ρ) of a terminal string from S with f-description FD, the number of nodes of c that relate to the same f-structure element is not greater than k, that is, $|\{n' \mid FD \vdash n = n'\}| \leq k$, for any node n of c.

In this definition, condition (i) defines the format of the rule annotations, indicating that finite-copying LFGs are a quite restricted subclass of suitably annotated LFGs. Specifically, they do not allow trivial annotations or annotations that give rise to f-structure reentrancies, the annotations that are required for linguistic description.

Given these notational restrictions, structure sharing can only be achieved through instantiated function-assigning annotations. This specific type of structure sharing is occasionally referred to as "zipper" unification. That is, if two distinct nodes n and n' co-refer (have the same f-structure) in a valid derivation ($FD \vdash n = n'$), then there must always be a node $\hat{n}$ dominating these nodes and the sequences of function-assigning annotations obtained from the annotations on the paths from $\hat{n}$ to n and n', respectively, must be identical, that is, form a "zipper". As a consequence, co-referring nodes must have the same c-structure depth.

The bounding condition (ii) limits the number of non-local dependencies or co-referring nodes that can arise through structure sharing and thus excludes c-structure recursions that give rise to zippers of size greater than any constant k. Indeed, Seki et al. have shown that the recognition problem is NP-complete for nondeterministic copying LFGs, which are similar to finite-copying LFGs except that they are not required to satisfy the bounding condition. Thus the bounding condition is crucial for tractable performance even with the severe notational restrictions of the finite-copying formalism. Consider for example the (suitably annotated) LFG grammar with the rules in (12).

(12) $\underset{}{\text{S}} \rightarrow \underset{(\uparrow \text{C}) = \downarrow}{\text{S}} \quad \underset{(\uparrow \text{C}) = \downarrow}{\text{S}} \qquad \underset{}{\text{S}} \rightarrow \underset{(\uparrow \text{C}) = \#}{\text{a}}$

The unbounded recursion of this grammar derives the language $\{\text{a}^{2^n} \mid n \geq 0\}$ with zippers that also grow exponentially. This language is not of constant growth and hence is not in the class of mildly context-sensitive languages (Joshi 1985). Seki et al. observe, however, that it is decidable whether a grammar notationally restricted according to condition (i) also satisfies the bounding condition (ii) and therefore has tractable computational properties.

4. k-Bounded LFG Grammars

In this section, we define a set of restrictions on LFG grammars with linguistically motivated annotations that are compatible with linguistic description and together ensure LCFRS equivalence. These restrictions thus preserve the key advantages of Seki et al.'s minimally expressive formalism.

Because finite boundedness is a necessary condition for LCFRS equivalence, clearly there must be a finite bound on the number of nodes in a functional domain.[7] These nodes are annotated with $\uparrow = \downarrow$ annotations and typically carry information about heads, coheads, and the local morphosyntactic features of a functional unit. The $\uparrow = \downarrow$ annotations map all the nodes in a functional domain to the same f-structure, and thus

7 The collection of trivial-annotated nodes that map to the same f-structure has been called a functional domain in linguistic theory (Bresnan 2001).

allow information to propagate up, down, and across the chain of nodes that relate to a single head.

Definition 10
An LFG grammar G has (**height-**)**bounded functional domains** if for any sequence of rules $A_1 \rightarrow \phi^1(A_2, D_2)\psi^1\ A_2 \rightarrow \phi^2(A_3, D_3)\psi^2\ ..\ A_n \rightarrow \phi^n(A_1, D_1)\psi^n$ of length $n \geq 1$, with $A_i \neq A_j$, for $1 \leq i < j \leq n$, $\uparrow = \downarrow \in D_i$ does not hold for all $i = 1, .., n$.

In the following, we refer to LFG grammars with linguistically suitable annotations and height-bounded functional domains as suitable LFGs.

Definition 11
A basic LFG G is called **suitable** if G is suitably annotated and G's functional domains are height-bounded.[8]

Whether a basic LFG G is suitable is easily decidable by inspecting G's rules and rule sequences of length less than or equal to $|N|$.

Because of the height bound, there is a simple transformation of a grammar G 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 detailed construction and the equivalence proof are provided in Appendix A.

Lemma 1
For any suitable LFG G we can construct a suitable LFG without $\uparrow = \downarrow$ annotations, denoted by $G^{\setminus\uparrow=\downarrow}$, such that $\Delta_G = \Delta_{G^{\setminus\uparrow=\downarrow}}$.

For the suitable LFG grammar in (1), for example, we then obtain a $\uparrow = \downarrow$-free grammar whose useful rules (that is, the ones whose left-hand category is reachable from the start symbol) are given in (13).[9]

(13) a. $\mathrm{S} \rightarrow$ NP $(\uparrow \mathrm{S}) = \downarrow$ NP $(\uparrow \mathrm{O}) = \downarrow$ $\overline{\mathrm{VP}}$ $(\uparrow \mathrm{X}) = \downarrow$ V $(\uparrow \mathrm{P}) = \mathrm{V}$ $(\uparrow \mathrm{X\,S}) = (\uparrow \mathrm{O})$ $(\uparrow \mathrm{S\,A}) = \mathrm{C}$ $\overline{\mathrm{V}}$ $(\uparrow \mathrm{X}) = \downarrow$

b. $\overline{\mathrm{VP}} \rightarrow$ NP $(\uparrow \mathrm{O}) = \downarrow$ $\overline{\mathrm{VP}}$ $(\uparrow \mathrm{X}) = \downarrow$

c. $\overline{\mathrm{VP}} \rightarrow$ NP $(\uparrow \mathrm{O}) = \downarrow$ $(\uparrow \mathrm{X\,X}) = \#$

8 Instead of an explicit boundary condition as in Definition 10, the bound can also be specified as an extragrammatical parameter. We return to that option in Section 6.

9 Even though our example grammar (with only four lexical rules) yields a $\uparrow = \downarrow$-free grammar with very few rules, the elimination of the trivial annotations can—without further restriction—easily result in a grammar that is substantially larger than the original. We return to this issue in Section 6.

d. $\overline{\text{V}} \rightarrow$ V ($\uparrow$ P) = V; $\overline{\text{V}}$ ($\uparrow$ X) = $\downarrow$; ($\uparrow$ X S) = ($\uparrow$ O)

e. NP $\rightarrow$ n ($\uparrow$ P) = N; ($\uparrow$ A) = C

f. $\overline{\text{V}} \rightarrow$ v ($\uparrow$ P) = V; ($\uparrow$ X) = #

It is straightforward to see that for suitable LFGs any bound k, $k \geq 1$, on derivations of the sort that Seki et al. used in the definition of finite-copying grammars must also be undecidable. Note first that the emptiness problem is already undecidable for 1-bounded suitable LFGs as the grammars with rules of the form (4, 5) are trivial-free and 1-bounded. With two suitable LFGs G^{ub} and G^{b} that are known to be unbounded and k-bounded, respectively, we can easily reduce the emptiness problem to the k- or finite-boundedness problem. Given an arbitrary 1-bounded suitable LFG G, we construct a suitable grammar G' with a rule set consisting of the rules of G^{ub} and G^{b}, and the two start rules $\text{S}' \rightarrow \text{S}\ \text{S}^{ub}$ and $\text{S}' \rightarrow\ \text{S}^{b}$ (with N, N^{ub}, N^{b} and $\{\text{S}'\}$ pairwise disjoint). Then by construction, G' is k-bounded iff $L(G) = \varnothing$.

We therefore rely on the observation that every LFG grammar G can be decomposed into two subgrammars, a **reentrancy-free kernel** and an **atom-free kernel**, both with a decidable emptiness problem. The reentrancy-free kernel of G is formed by removing all reentrancies from its rules, leaving just function assignments and atomic-value annotations. The atom-free kernel is formed by removing all atomic-value annotations from its rules, so that only function assignments and reentrancies remain. The decomposition then allows us to define some necessary and sufficient restrictions on the LFG grammars G by carefully regulating the interplay of the functional descriptions of these two grammars. Formally, we state all further restrictions on the reentrancy-free and atom-free kernels and their interaction in terms of the descriptions $FD_{\setminus\mathcal{R}}$ and $FD_{\setminus\mathcal{A}}$ obtained from FD by removing all instantiated reentrancies and all instantiated atomic-valued annotations, respectively.

Because k-boundedness is a necessary condition for LCFRS equivalence we first require the reentrancy-free kernel of $G^{\setminus\uparrow=\downarrow}$ to be k-bounded. We show in Section 4.1 that k-boundedness is decidable for the reentrancy-free kernel of any suitable LFG.

Definition 12
A suitable LFG G has a ***k*-bounded reentrancy-free kernel** iff for every derivation (c, ρ) of a terminal string from S with f-description FD and clash-free $FD_{\setminus\mathcal{R}}$ in $G^{\setminus\uparrow=\downarrow}$, $|\{n' \mid FD_{\setminus\mathcal{R}} \vdash n = n'\}| \leq k$, for any node n of c.

We now identify a minimally intrusive restriction that ensures that $G^{\setminus\uparrow=\downarrow}$ conserves the ϕ projection of its reentrancy-free kernel, so that in particular $G^{\setminus\uparrow=\downarrow}$ is k-bounded if its reentrancy-free kernel is k-bounded. This restriction relates to another notion in the LFG literature, the concept of "nonconstructivity" that has been discussed in the context of functional uncertainty and off-path constraints (Zaenen and Kaplan 1995, Section 3; Dalrymple et al. 1995a, page 133; Crouch et al. 2008, chapter on grammatical notations). We provide a technical formulation of this notion and propose it as a general condition on the operation of reentrancy annotations, a condition that is necessary to

ensure LCFRS equivalence. Nonconstructive reentrancies are defined in terms of the interplay between the reentrancy-free and the atom-free kernel in the following way:

Definition 13
A suitable LFG with k-bounded reentrancy-free kernel has **nonconstructive reentrancies** iff for every derivation (c, ρ) of a terminal string from S with f-description FD and clash-free $FD_{\setminus\mathcal{R}}$ in $G^{\setminus\uparrow=\downarrow}$

$$\text{if } FD_{\setminus\mathcal{A}} \vdash n = n', \text{ with } n \text{ distinct from } n', \text{ then } FD_{\setminus\mathcal{R}} \vdash n = n'.$$

This restriction ensures that reentrancies in $G^{\setminus\uparrow=\downarrow}$ cannot interact with the function-assigning annotations of the reentrancy-free kernel to produce node equalities beyond those that follow just from that kernel's more limited set of annotations. We show in Section 4.2 that nonconstructivity is decidable for suitable LFGs with k-bounded reentrancy-free kernel and only short reentrancies. Moreover, suitable LFGs with k-bounded reentrancy-free kernel and nonconstructive short reentrancies are equivalent to LCFRSs (see Section 5.2) and thus have a decidable emptiness problem. This, however, does not generalize to suitable grammars with k-bounded reentrancy-free kernel and long reentrancies. To see this, consider the grammars that we used in Section 3.2 to show that it is in general undecidable whether there are derivations with co-referring nodes. These grammars are trivial-free grammars with 1-bounded reentrancy-free kernel, but without atomic-valued annotations. Thus $FD = FD_{\setminus\mathcal{A}}$, for any derivation. Moreover, because of the 1-boundedness of the reentrancy-free kernel, $FD_{\setminus\mathcal{R}} \vdash n = n'$ does not hold for any pair of distinct nodes n and n' in any derivation. Thus the nonconstructivity condition of Definition 13 is undecidable because $FD_{\setminus\mathcal{A}} \vdash n = n'$ is undecidable.[10]

We observe, however, that long reentrancies of the form $(\uparrow \text{F G}) = (\uparrow \text{H})$, which are typically used to specify the structure sharing relationships in grammatical control constructions (e.g., $(\uparrow \text{XCOMP SUBJ}) = (\uparrow \text{OBJ})$), can always be shortened in derivations that meet the requirements of the Coherence Condition.[11] This is, specifically, because the controllee (SUBJ) of an instantiated control equation $(n \text{ XCOMP SUBJ}) = (n \text{ OBJ})$ 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, for example $(\uparrow \text{PRED}) = \text{'WALK}\langle\text{SUBJ}\rangle\text{'}$. Thus, $(\uparrow \text{PRED}) = \text{'WALK}\langle\text{SUBJ}\rangle\text{'}$ must instantiate to the equation $(n' \text{PRED}) = \text{'WALK}\langle\text{SUBJ}\rangle\text{'}$ at some node n', and the functional description must also entail an equation $(n \text{ XCOMP}) = n'$ that links the complement to the higher clause. This (derived) function assignment justifies the reduction of the long reentrancy to the equivalent shorter one $(n' \text{ SUBJ}) = (n \text{ OBJ})$.

Certainly, we cannot anticipate this effect of the coherence condition just by removing from consideration all derivations in which long reentrancies cannot be reduced to shorter ones: Such an additional filter on valid derivations would preserve the undecidability of the emptiness problem. And this would, as demonstrated in Section 3.2, even apply to the weaker filter where the shortening equations are required to follow from the atom-free subset of FD. Hence those grammars cannot be equivalent to LCFRSs

10 Note that based on the short-reentrancy grammars of the reduction to the emptiness problem a similar argument can be used to show that, even for grammars with only short reentrancies, the nonconstructivity of reentrancies would be undecidable if in the definition all node equalities $n = n'$ following from FD (and not $FD_{\setminus\mathcal{A}}$) were required to follow from $FD_{\setminus\mathcal{R}}$.

11 We provide a technical interpretation of Coherence (and Completeness) in Section 5.4.

(because LCFRSs have a decidable emptiness problem). Therefore, we require the derivations to meet the stronger stipulation that the shortening equations $(n\ \text{XCOMP}) = n'$ follow from the f-description of the reentrancy-free kernel, which excludes the use of other reentrancies in those supporting inferences. This is made explicit in the following extension to the notion of derivation.

Definition 14
A pair (c, ρ) consisting of a labeled tree c and a mapping ρ from the nonterminal nodes of c into R is a **derivation** of string $s \in (N \cup T)^*$ from B **with functional description** FD in G iff c and ρ satisfy the conditions (i–iv) of Definition 3 and

(v) if $(n\ \text{F G}) = (n\ \text{H}) \in FD$, then $FD_{\setminus\mathcal{R}} \vdash (n\ \text{F}) = n'$ for some node n'.

Certainly, this stronger condition excludes some analyses that otherwise appear to lie within scope of the normal derivational machinery of the LFG formalism. In all likelihood the derivations that this restriction eliminates would also fail to meet the coherence condition and thus no linguistically significant derivations will be lost.

LFG grammars that meet all the restrictions we have set out are called k-bounded LFGs.

Definition 15
A basic LFG G is ***k*-bounded** iff

(i) G is suitably annotated,

(ii) G has height-bounded functional domains,

(iii) G's reentrancy-free kernel is k-bounded,

(iv) G has nonconstructive reentrancies.

For k-bounded LFGs the recognition and emptiness problems are decidable because, as we show in Section 5.2, k-bounded LFGs are weakly equivalent to k-LCFRSs. Moreover, because of the nonconstructivity of the reentrancies and condition (v) of Definition 14, $G^{\setminus\uparrow=\downarrow}$ cannot produce node equalities beyond those of its reentrancy-free kernel.

Corollary 1
Let G be a k-bounded LFG grammar. Then for every derivation (c, ρ) of a terminal string from S *with clash-free f-description FD in $G^{\setminus\uparrow=\downarrow}$*

$$\textit{if } FD \vdash n = n', \textit{ with } n \textit{ distinct from } n', \textit{ then } FD_{\setminus\mathcal{R}} \vdash n = n'.$$

Hence, the k-boundedness of G's reentrancy-free kernel also extends to $G^{\setminus\uparrow=\downarrow}$, although, because of the filtering effect of the atomic-valued annotations, the bound of $G^{\setminus\uparrow=\downarrow}$ can be smaller than k.[12]

12 For k-bounded LFGs the universal generation problem is decidable as well. This follows directly from the decidability result in Wedekind and Kaplan (2012). That article describes an algorithm that produces for

Corollary 2
Let G be a k-bounded LFG grammar. Then for every derivation (c, ρ) of a terminal string from S *with clash-free f-description FD in $G^{\setminus\uparrow=\downarrow}$, $|\{n' \mid FD \vdash n = n'\}| \leq k$, for any node n of c.*

4.1 The k-Boundedness of the Reentrancy-Free Kernel

As for Seki et al.'s finite-copying grammars, structure sharing in reentrancy-free kernels of suitable LFGs can only be achieved through instantiated function-assigning annotations. Co-referring nodes therefore belong to identical (zipper) paths and lie at the same depth below a common ancestor node.

We exploit a shrinking argument to show that the k-boundedness of G's reentrancy-free kernel can be determined by inspecting a sufficiently large but finite number of "small" derivations in $G^{\setminus\uparrow=\downarrow}$. Shrinking of a derivation is accomplished by removing certain parts of that derivation, yielding a smaller derivation. To identify the conditions that allow shrinking of $G^{\setminus\uparrow=\downarrow}$'s derivations, let (c, ρ) be a derivation of a terminal string from S with f-description FD and clash-free $FD_{\setminus\mathcal{R}}$ in $G^{\setminus\uparrow=\downarrow}$. Consider the sets of nodes that co-refer in $FD_{\setminus\mathcal{R}}$, that is, the sets $[n] = \{n' \mid FD_{\setminus\mathcal{R}} \vdash n = n'\}$ where n is a nonterminal node. Set $[n] = \{n\}$ if n does not occur in FD.

As an illustration, consider in Figure 4 the annotated c-structure of the terminal string nnnvvv in the $\uparrow = \downarrow$-free grammar with the rules in (13). This more conventional way of depicting c-structures and their licensing rule mappings (Kaplan and Bresnan 1982) makes it easy to read the f-description from the licensed c-structure. The description $FD_{\setminus\mathcal{R}}$ obtained from the f-description FD of this derivation is given in (14).

(14)
$$\left\{\begin{array}{l}
(root\ \textsc{s}) = n_1, (n_1\ \textsc{p}) = \textsc{n}, \\
\qquad\qquad\quad (n_1\ \textsc{a}) = \textsc{c}, \\
(root\ \textsc{o}) = n_2, (n_2\ \textsc{p}) = \textsc{n}, \\
\qquad\qquad\quad (n_2\ \textsc{a}) = \textsc{c}, \\
(root\ \textsc{x}) = n_3, (n_3\ \textsc{x x}) = \#, \\
\qquad\qquad\quad (n_3\ \textsc{o}) = n_5, (n_5\ \textsc{p}) = \textsc{n}, \\
\qquad\qquad\qquad\qquad\quad (n_5\ \textsc{a}) = \textsc{c}, \\
(root\ \textsc{x}) = n_4, (n_4\ \textsc{p}) = \textsc{v}, \\
\qquad\qquad\quad (n_4\ \textsc{x}) = n_6, (n_6\ \textsc{p}) = \textsc{v}, \\
\qquad\qquad\qquad\qquad\quad (n_6\ \textsc{x}) = \#, \\
(root\ \textsc{p}) = \textsc{v}, \\
(root\ \textsc{s a}) = \textsc{c}
\end{array}\right\}$$

The equivalence classes that result from $FD_{\setminus\mathcal{R}}$ are $\{root\}$, $\{n_1\}$, $\{n_2\}$, $\{n_3, n_4\}$, $\{n_5\}$, and $\{n_6\}$.

Next, we exhibit the atomic-valued information that is inherited to the nodes in $[n]$ from the nodes in higher-level equivalence classes. For this purpose, we replace each occurrence of a node n of c in FD by $[n]$ and denote this description by ${}^eFD$. From the description in (14) we thus obtain the description ${}^eFD_{\setminus\mathcal{R}}$ in (15).

an arbitrary LFG grammar G and an arbitrary acyclic input f-structure F a context-free grammar G_F that describes exactly the set of strings that the given LFG grammar associates with that f-structure. For k-bounded LFGs all node equalities follow from the instantiated function-assigning annotations of an f-description. Thus, even if the input f-structure contains cycles, the context-free grammar can be constructed on the basis of the terms that are defined in the canonical expansion of the f-structure but do not encompass a cycle. This is again a finite set. Then for any G and arbitrary F, G derives a terminal string with F iff $L(G_F) \neq \varnothing$ (which is decidable for context-free grammars).

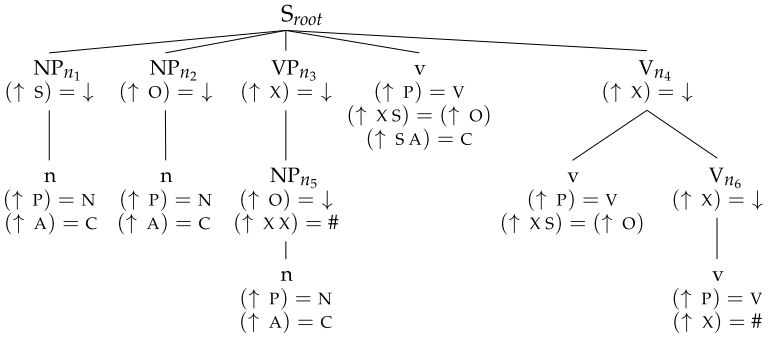


Figure 4
The annotated c-structure of the terminal string nnnvvv in the $\uparrow = \downarrow$-free grammar comprising the rules in (13).

(15)
$$\left\{\begin{array}{lll}
(\{root\}\ \textsc{s}) = \{n_1\}, & (\{n_1\}\ \textsc{p}) = \textsc{n}, & \\
 & (\{n_1\}\ \textsc{a}) = \textsc{c}, & \\
(\{root\}\ \textsc{o}) = \{n_2\}, & (\{n_2\}\ \textsc{p}) = \textsc{n}, & \\
 & (\{n_2\}\ \textsc{a}) = \textsc{c}, & \\
(\{root\}\ \textsc{x}) = \{n_3, n_4\}, & (\{n_3, n_4\}\ \textsc{x}\,\textsc{x}) = \#, & \\
 & (\{n_3, n_4\}\ \textsc{o}) = \{n_5\}, & (\{n_5\}\ \textsc{p}) = \textsc{n}, \\
 & & (\{n_5\}\ \textsc{a}) = \textsc{c}, \\
 & (\{n_3, n_4\}\ \textsc{p}) = \textsc{v}, & \\
 & (\{n_3, n_4\}\ \textsc{x}) = \{n_6\}, & (\{n_6\}\ \textsc{p}) = \textsc{v}, \\
 & & (\{n_6\}\ \textsc{x}) = \#, \\
(\{root\}\ \textsc{p}) = \textsc{v}, & & \\
(\{root\}\ \textsc{s}\,\textsc{a}) = \textsc{c} & &
\end{array}\right\}$$

Here and in our further considerations we use the following closure construction to derive the inherited atomic-valued equations.

Definition 16
Let *FD* be a description obtained from linguistically suitable annotations by instantiating the metavariables with specific elements, such as classes, as denoted by $\boldsymbol{a}$ and $\boldsymbol{b}$. Then the closure $\overline{FD}$ of *FD* is defined as being the smallest set that includes *FD* and is closed under the rule: if $(\boldsymbol{a}\ \textsc{f}\ \sigma) = v \in \overline{FD}$ and $(\boldsymbol{a}\ \textsc{f}) = (\boldsymbol{b}\ \sigma') \in FD$, then $(\boldsymbol{b}\ \sigma'\ \sigma) = v \in \overline{FD}$.

For the description provided by the reentrancy-free kernel it is always the case that σ' is empty. If ${}^{e}FD_{\backslash\mathcal{R}}$ contains an atom-value equation $([n]\ \textsc{f}\ \sigma) = v$ and a function assignment $([n]\ \textsc{f}) = [n']$, the closure of ${}^{e}FD_{\backslash\mathcal{R}}$ will contain the shortened value assignment $([n]\ \sigma) = v$. In our simple example we can apply $(\{root\}\ \textsc{s}) = \{n_1\}$ to $(\{root\}\ \textsc{s}\,\textsc{a}) = \textsc{c}$ and $(\{n_3, n_4\}\ \textsc{x}) = \{n_6\}$ to $(\{n_3, n_4\}\ \textsc{x}\,\textsc{x}) = \#$, but in these cases the resulting equations $(\{n_1\}\ \textsc{a}) = \textsc{c}$, $(\{n_6\}\ \textsc{x}) = \#$ are already contained in ${}^{e}FD_{\backslash\mathcal{R}}$.

Shrinking depends on comparing the atomic-valued information that the closure associates with the node equivalence classes to find matching classes at different levels of a derivation. To that end, we abstract away from particular class instantiations by defining, for any description *FD*, a function h_{FD} that assigns to an instantiating element a characterization of all and only the atomic-valued equations that a description associates with that element. This function is defined in (16).

(16) $h_{FD}(\boldsymbol{a}) = \{(* \ \sigma) = v \mid (\boldsymbol{a} \ \sigma) = v \in FD\}$

Note that by construction of $\overline{^{e}FD_{\setminus\mathcal{R}}}$ the length of σ is bounded by the maximum length of the attribute sequences occurring in the atomic-valued annotations of G. This length is in the following denoted by ℓ. The values of $h_{\overline{^{e}FD_{\setminus\mathcal{R}}}}$ are members of the set $\mathcal{E}$ of all possible clash-free atomic-valued specifications.

Definition 17
The (**atomic-valued**) **description space** is the set

$$\mathcal{E} = \{E \mid E \subseteq \{(* \ \sigma) = v \mid \sigma \in \mathcal{A}^{+}, |\sigma| \leq \ell, \text{ and } v \in \mathcal{V}\} \text{ and } E \text{ is clash-free}\}.$$

For our example we then obtain the following $h_{\overline{^{e}FD_{\setminus\mathcal{R}}}}$ assignment. Here and in the following, we omit the reference to the description if it is clear from the context. (The shrinking arguments, in particular, are all based on descriptions of the reentrancy-free kernel and thus depend on $FD_{\setminus\mathcal{R}}$.)

(17)
$$h(\{root\}) = \left\{\begin{matrix}(* \ \text{P}) = \text{V}, \\ (* \ \text{S A}) = \text{C}\end{matrix}\right\}$$
$$h(\{n_1\}) = h(\{n_2\}) = h(\{n_5\}) = \left\{\begin{matrix}(* \ \text{P}) = \text{N}, \\ (* \ \text{A}) = \text{C}\end{matrix}\right\}$$
$$h(\{n_3, n_4\}) = \left\{\begin{matrix}(* \ \text{X X}) = \#, \\ (* \ \text{P}) = \text{V}\end{matrix}\right\}$$
$$h(\{n_6\}) = \left\{\begin{matrix}(* \ \text{P}) = \text{V}, \\ (* \ \text{X}) = \#\end{matrix}\right\}$$

We can shrink a derivation if there are node classes $[n^l]$ and $[\check{n}]$ where at least one node in $[n^l]$ dominates a node in $[\check{n}]$, h assigns to $[n^l]$ and $[\check{n}]$ the same sets of atomic-valued schemata, and there is a mapping g from $[n^l]$ to $[\check{n}]$ such that n^i and $g(n^i)$ are licensed by the same rule ($\rho_{n^i} = \rho_{g(n^i)}$), for each n^i in $[n^l]$.

Shrinking is accomplished by replacing the subderivations under nodes in the equivalence class $[n^l]$ by the subderivations under nodes determined by the g mapping. Specifically, we define the replacement $\mathbf{Repl}^{g}_{(c,\rho)}([n^l])$ as the derivation produced from (c, ρ) by removing the subtree and rule mapping under each node n^i in $[n^l]$ and then inserting under n^i a copy of the subtree and rule mapping under $g(n^i)$. The shrinking process is illustrated in Figure 5 and formally defined as follows.

Definition 18
Let (c, ρ) be a derivation in a suitable LFG grammar $G^{\uparrow=\downarrow}$. For any pair of classes $[n^l]$ and $[\check{n}]$ satisfying the conditions

(i) at least one node in $[n^l]$ dominates a node in $[\check{n}]$,

(ii) $h([n^l]) = h([\check{n}])$,

(iii) there is a function g from $[n^l]$ to $[\check{n}]$ such that $\rho_{n^i} = \rho_{g(n^i)}$ for each n^i in $[n^l]$,

we define the shrink operation sh by

$$sh_{(c,\rho)}([n^l], [\check{n}]) = \mathbf{Repl}^{g}_{(c,\rho)}([n^l]).$$

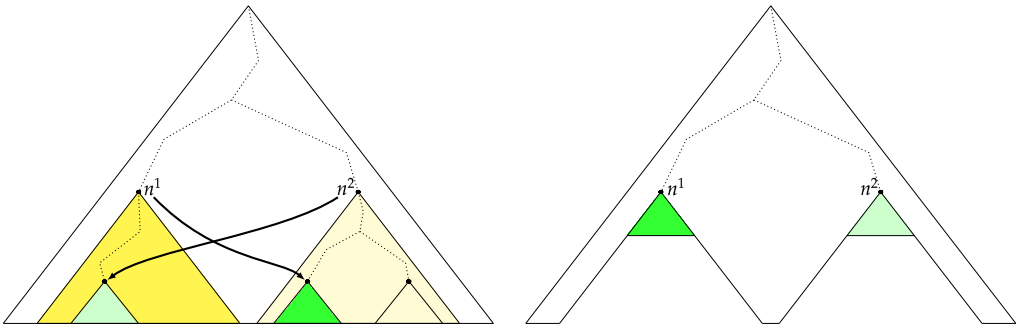


Figure 5
Illustration of the shrinking process. In the derivation tree on the left-hand side, the node set $\{n^1, n^2\}$ corresponds to the class $[n^l]$ and the set comprising the three lower nodes corresponds to $[\check{n}]$. The dotted lines indicate dominance and the solid arrows the g mapping. If $h([n^l]) = h([\check{n}])$, then shrinking is performed by lifting (distinct copies of) the subderivations dominated by $g(n^1)$ and $g(n^2)$ to n^1 and n^2, respectively. The result is shown on the right-hand side.

The shrink operation $sh^{1-1}_{(c,\rho)}([n^l],[\check{n}])$ is defined as $sh_{(c,\rho)}([n^l],[\check{n}])$ except that g is a one-to-one correspondence.

Because $sh_{(c,\rho)}([n^l],[\check{n}])$ is obviously a derivation in $G^{\uparrow=\downarrow}$, it remains to be shown that the f-description that G's reentrancy-free kernel provides for this derivation is clash-free. Let FD^t be the f-description of the (partial) derivation obtained from (c,ρ) by removing the subtree and rule mapping under each node n^i in $[n^l]$, and FD^b be the union of the f-descriptions obtained from the copies of the subderivations rooted at $g(n^i)$. Then, by construction, $sh_{(c,\rho)}([n^l],[\check{n}])$ gets assigned the f-description $FD' = FD^t \cup FD^b$. Certainly, $FD^t_{\backslash\mathcal{R}}$ and $FD^b_{\backslash\mathcal{R}} \cup \{n^i = n^k \mid n^i, n^k \in [n^l]\}$ are both clash-free. Because node equalities can only be established through function assignments, $FD^t_{\backslash\mathcal{R}} \vdash n = n'$ if $FD_{\backslash\mathcal{R}} \vdash n = n'$, for all n, n' occurring in FD^t. Thus $\{n' \mid FD^t_{\backslash\mathcal{R}} \vdash n^l = n'\}$ is equal to $[n^l]$, and, for the h mapping h' of $sh_{(c,\rho)}([n^l],[\check{n}])$, $h'([n^l]) = h([n^l])$ by condition (iii). Hence $FD'_{\backslash\mathcal{R}}$ must be clash-free because of condition (ii).

Lemma 2
Let (c,ρ) be a derivation in a suitable LFG grammar $G^{\uparrow=\downarrow}$ with clash-free $FD_{\backslash\mathcal{R}}$ and $[n^l]$ and $[\check{n}]$ be a pair of classes satisfying (i–iii) of Definition 18. If FD' is the f-description of $sh_{(c,\rho)}([n^l],[\check{n}])$ (respectively $sh^{1-1}_{(c,\rho)}([n^l],[\check{n}])$), then $FD'_{\backslash\mathcal{R}}$ is clash-free.

For the following notice that sh^{1-1} preserves the multiplicities of the licensing rules and clash-freeness whereas sh only preserves clash-freeness, but both operations abstract from the left-to-right c-structure order. To see this, note first that we can assign to each $[n]$ of a derivation a multiset (P,m) whose support P consists of the rules licensing the nodes in $[n]$ ($P = \{\rho_{\bar{n}} \mid \bar{n} \in [n]\}$) and whose multiplicity function m specifies for each rule r in P the number of nodes in $[n]$ that are licensed by r ($m(r) = |\{\bar{n} \in [n] \mid \rho_{\bar{n}} = r\}|$ for all $r \in P$). Let f^m be the function that assigns to each $[n]$ both the multiset assigned to $[n]$ and its h value, that is, $f^m([n]) = ((P,m), h([n]))$. Because there is a one-to-one correspondence g between $[n^l]$ and $[\check{n}]$ with $\rho_{n^i} = \rho_{g(n^i)}$ for each n^i in $[n^l]$, if and only if the multisets assigned to $[n^l]$ and $[\check{n}]$ are identical, we can apply sh^{1-1} to a derivation if there are nodes n^l and $\check{n}$ on a path with $f^m([n^l]) = f^m([\check{n}])$.

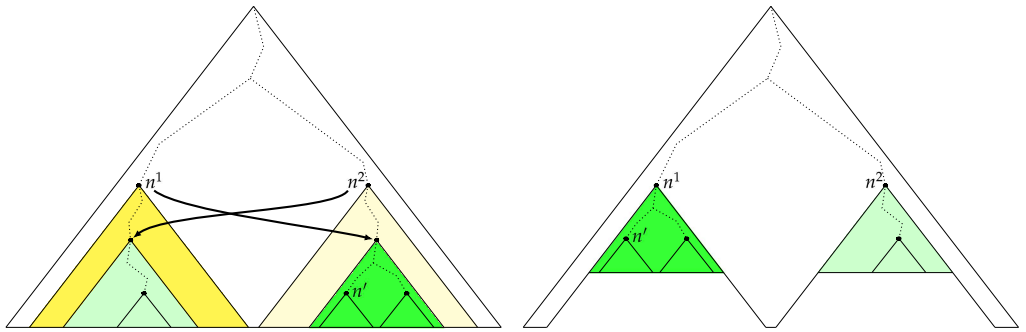


Figure 6
A sample derivation violating the k-boundedness condition for $k = 2$ is shown on the left-hand side. The set comprising the three lower nodes corresponds to $[n']$, the violating instance that is highest in c. Suppose that there are two equivalence classes containing nodes of the path from *root* to the mother of n' that satisfy the conditions for applying $sh^{1-1}_{(c,\rho)}$: the set $[n^l] = \{n^1, n^2\}$ and the image $[\check{n}]$ of $\{n^1, n^2\}$ under g (indicated by the solid arrows). Then the smaller derivation $sh^{1-1}_{(c,\,\rho)}([n^l], [\check{n}])$, depicted on the right-hand side, still violates the k-boundedness condition.

With respect to sh, observe first that by definition, $sh_{(c,\rho)}$ can be applied to a pair $([n^l], [\check{n}])$, if the two classes agree in their h values ($h([n^l]) = h([\check{n}])$) and there is a mapping g from $[n^l]$ to $[\check{n}]$ such that $\rho_{n^i} = \rho_{g(n^i)}$, for each n^i in $[n^l]$. Because there is such a g mapping from $[n^l]$ to $[\check{n}]$ if and only if each rule that licenses a node in $[n^l]$ also licenses a node in $[\check{n}]$ ($\{\rho_{n^i} \mid n^i \in [n^l]\} \subseteq \{\rho_{\check{n}'} \mid \check{n}' \in [\check{n}]\}$), we can shrink if $h([n^l]) = h([\check{n}])$ and the support of the multiset assigned to $[n^l]$ is included in the support of the multiset assigned to $[\check{n}]$.

Now we demonstrate that it can be decided whether the reentrancy-free kernel of a suitable LFG G is k-bounded by inspecting derivations in $G^{\setminus\uparrow=\downarrow}$ whose (c-structure) depth does not exceed some fixed upper bound.

Lemma 3
For any suitable LFG G and any constant k, it is decidable whether G's reentrancy-free kernel is k-bounded.[13]

Proof
Suppose there is a derivation (c, ρ) of a terminal string in $G^{\setminus\uparrow=\downarrow}$ that violates the k-boundedness condition. Let n' be one of the topmost nodes of c with $|[n']| > k$ and consider the equivalence classes that contain the nodes of the path from *root* to the mother of n'. Observe first that for any of these equivalence classes $[n^l]$ and $[\check{n}]$ that satisfy the conditions for applying $sh^{1-1}_{(c,\rho)}$, $sh^{1-1}_{(c,\rho)}([n^l], [\check{n}])$ must still violate the k-boundedness condition because sh^{1-1} preserves the multiplicities of the licensing rules and therefore all nodes in $[n']$ are still dominated by nodes in $[\check{n}]$. This situation is illustrated in Figure 6.
To determine the upper bound on the path length from *root* to n', recall that we can shrink the derivation if there are nodes n^l and $\check{n}$ on the path from *root* to the mother of n' with $f^m([n^l]) = f^m([\check{n}])$. Because the set of all multisets over R with cardinality less than or equal to k, in the following denoted by $\mathcal{M}_{\leq k}(R)$, is finite and the number of h

13 We show in Appendix B that for any suitable LFG it is also decidable whether or not there is a finite bound on its reentrancy-free kernel.

values is bounded by $|\mathcal{E}|$, a path from *root* to n' longer than $|\mathcal{M}_{\leq k}(R) \times \mathcal{E}|$ must contain equivalence classes that satisfy the conditions for shrinking. Thus, if the grammar violates the k-boundedness condition then there must be a derivation with a class $[n']$, with $|[n']| > k$, and the depth of n' is less than or equal to $|\mathcal{M}_{\leq k}(R) \times \mathcal{E}|$ (recall here that the depth of *root* is 0).

Now we determine an upper bound on the path length through the remaining equivalence classes of the derivation. If we can apply $sh_{(c,\rho)}$ to a pair $([n^l], [\check{n}])$ of these classes (with a node in $[n^l]$ properly dominating a node in $[\check{n}]$), then, obviously, the violating class $[n']$ remains in $sh_{(c,\rho)}([n^l], [\check{n}])$. Recall that we can apply $sh_{(c,\rho)}$ to $([n^l], [\check{n}])$ if $h([n^l]) = h([\check{n}])$ and if the support of the rule multiset assigned to $[n^l]$ is included in the support of the one assigned to $[\check{n}]$. Because the supports are nonempty subsets of R, shrinking with $sh_{(c,\rho)}$ is thus possible if the path length between two nodes is greater or equal to $|Pow^+(R) \times \mathcal{E}|$, where $Pow^+(R)$ denotes the set of all nonempty subsets of R.

Thus if $G^{\uparrow=\downarrow}$ violates the k-boundedness condition then there must be a derivation (c, ρ) with a c-structure depth less than or equal to $|\mathcal{M}_{\leq k}(R) \times \mathcal{E}| + |Pow^+(R) \times \mathcal{E}| + 1$ that violates k-boundedness. Hence, for any suitable LFG G it can be determined whether G's reentrancy-free kernel is k-bounded. ■

4.2 Nonconstructive Reentrancies

In this section, we show that the nonconstructivity condition of Definition 13 is decidable for suitable LFGs with k-bounded reentrancy-free kernel. For the proof we use a shrinking argument similar to that used for proving the decidability of the k-boundedness of the reentrancy-free kernel. The shrinking argument crucially depends on the fact that all long reentrancies can be shortened in a valid derivation, because, for descriptions including only instantiated short reentrancies and function assignments, nonconstructivity violations can be established through substitution proofs of a certain structure.

To see this, suppose that the descriptions are closed under symmetry and recall that the rule of substituting equals for equals has the form

$$\frac{e \qquad t = t'}{e'}$$

where e is an equation containing subterm t and e' is obtained from e by replacing one occurrence of t in e by t'. Then we know that any proof of an equation $t_{m+1} = t_0$ can be converted into a left-branching substitution proof of the form (18) where $t_{m+1} = t_{m+1}$ is either in *FD* (or follows from *FD* by partial reflexivity[14]) and where t_{m+1} is rewritten to t_0 by a sequence of substitutions all justified by equations $e_i \in FD$, $i = 1, .., m+1$ (see Statman [1977] for details on the proof conversions).

(18)
$$\dfrac{\dfrac{\dfrac{\dfrac{t_{m+1} = t_{m+1} \qquad e_{m+1}}{t_{m+1} = t_m} \qquad e_m}{t_{m+1} = t_{m-1}}}{\ddots}}{\dfrac{\dfrac{t_{m+1} = t_2 \qquad e_2}{t_{m+1} = t_1} \qquad e_1}{t_{m+1} = t_0}}$$

14 It may be the case that such a left-branching proof must start with a reflexive equation $t_{m+1} = t_{m+1}$ that is not in *FD* but can be inferred by partial reflexivity from an equation $(t_{m+1}\ \sigma) = t''$ in *FD*. Partial

The decidability of the nonconstructivity condition then results from the fact that there are always canonical proofs of the form (19) of equalities that violate this condition if the descriptions contain only function assignments and short reentrancies.

(19)
$$\dfrac{\dfrac{\dfrac{\boldsymbol{a}=\boldsymbol{a} \quad \boldsymbol{a}=(\boldsymbol{a}^{m}\ \mathrm{F}^{m})}{\boldsymbol{a}=(\boldsymbol{a}^{m}\ \mathrm{F}^{m})} \quad (\boldsymbol{a}^{m}\ \mathrm{F}^{m})=(\boldsymbol{a}^{m-1}\ \mathrm{F}^{m-1})}{\boldsymbol{a}=(\boldsymbol{a}^{m-1}\ \mathrm{F}^{m-1})}}{\begin{array}{c}\vdots\\ \dfrac{\boldsymbol{a}=(\boldsymbol{a}^{2}\ \mathrm{F}^{2}) \quad (\boldsymbol{a}^{2}\ \mathrm{F}^{2})=(\boldsymbol{a}^{1}\ \mathrm{F}^{1})}{\boldsymbol{a}=(\boldsymbol{a}^{1}\ \mathrm{F}^{1})} \quad (\boldsymbol{a}^{1}\ \mathrm{F}^{1})=\boldsymbol{a}'\end{array}}}{\boldsymbol{a}=\boldsymbol{a}'}$$

Notice for the following that we can always shorten a proof of the form (18) (or (19)) if the right-hand terms of two derived equations are identical.

Lemma 4
Let FD be a description obtained from short reentrancies and function-assigning annotations by instantiating the metavariables with classes (or other elements) denoted by $\boldsymbol{a}$, $\boldsymbol{a}^1$, $\boldsymbol{a}^2$, .., $\boldsymbol{b}$, $\boldsymbol{b}^1$, ... *If the equality of two distinct classes follows from FD, then there is at least one pair of distinct classes* $\boldsymbol{a}$ *and* $\boldsymbol{a}'$ *whose equality is established by a proof of the form (19).*

Proof
Suppose that $FD \vdash \boldsymbol{b}=\boldsymbol{b}'$, for two distinct elements $\boldsymbol{b}$ and $\boldsymbol{b}'$. Then there is a substitution proof of the form

$$\dfrac{\dfrac{\dfrac{\boldsymbol{b}=\boldsymbol{b}\ \ (=t_{m+1}) \quad e_{m+1}}{\boldsymbol{b}=t_{m} \quad e_{m}}}{\boldsymbol{b}=t_{m-1}}}{\begin{array}{c}\vdots\\ \dfrac{\boldsymbol{b}=t_{2} \quad e_{2}}{\boldsymbol{b}=t_{1} \quad e_{1}}\end{array}}}{\boldsymbol{b}=\boldsymbol{b}'\ (=t_{0})}$$

where $\boldsymbol{b}$ is rewritten to $\boldsymbol{b}'$. (This proof structure coincides with the one given in (18).) By assumption, the premises e_i are either class-valued (of the form $(\boldsymbol{b}\ \mathrm{F})=\boldsymbol{c}$) or complex-valued ($(\boldsymbol{b}\ \mathrm{F})=(\boldsymbol{c}\ \mathrm{G})$). Without loss of generality, we can assume that the right-hand terms are pairwise distinct ($t_j \neq t_i$, for $m+1 \geq j > i \geq 0$). (If $t_j = t_i$, $j > i$, we obtain a shorter proof of $\boldsymbol{b}=\boldsymbol{b}'$ by deleting the inference of $\boldsymbol{b}=t_{i+1}$ from $\boldsymbol{b}=t_j$.) Suppose that the proof is not already (up to biunique renaming) of the form (19). Then there are two cases to consider.
(a) There are one or more terms t_i, $1 \leq i \leq m$, whose attribute sequences are longer than one. Let $t_k = (\boldsymbol{b}^{k}\ \mathrm{F}^{k}\ \sigma)$, $\sigma \neq \epsilon$ be one of the longest terms (i.e., there is no other term that is longer than t_k). Then step k must lie between a previous step l that lengthens the attribute sequence and a following step j that shrinks it, with $e_{l+1} = \boldsymbol{b}^{l+1} = (\boldsymbol{b}^{l}\ \mathrm{F}^{l})$, $t_l = (\boldsymbol{b}^{l}\ \mathrm{F}^{l}\ \sigma)$, $t_j = (\boldsymbol{b}^{j}\ \mathrm{F}^{j}\ \sigma)$ and $e_j = (\boldsymbol{b}^{j}\ \mathrm{F}^{j}) = \boldsymbol{b}^{j-1}$. Obviously, $\boldsymbol{b}^{l+1}$ and $\boldsymbol{b}^{j-1}$ must be distinct because otherwise t_{l+1} and t_{j-1} would not be distinct, contradicting our assumption. But then there is a proof

reflexivity is the restriction of reflexivity to well-defined (object denoting) terms. It is a sound inference rule for the theory of partial functions for which full reflexivity does not hold.

$$\dfrac{\boldsymbol{b}^{l+1} = \boldsymbol{b}^{l+1} \qquad \boldsymbol{b}^{l+1} = (\boldsymbol{b}^{l}\ \mathrm{F}^{l})\ (= e_{l+1})}{\boldsymbol{b}^{l+1} = (\boldsymbol{b}^{l}\ \mathrm{F}^{l})}$$

$$\vdots$$

$$\dfrac{\boldsymbol{b}^{l+1} = (\boldsymbol{b}^{j}\ \mathrm{F}^{j}) \qquad (\boldsymbol{b}^{j}\ \mathrm{F}^{j}) = \boldsymbol{b}^{j-1}\ (= e_{j})}{\boldsymbol{b}^{l+1} = \boldsymbol{b}^{j-1}}$$

that has up to biunique renaming the form (19).
(b) All terms t_i are one-attribute terms or node classes. Because the proof is not of the form (19) there must be one or more t_j, $m > j > 1$, that are node classes. Let t_l be the first inferred term that is a node class and suppose that t_l is $\boldsymbol{b}^{l}$. Then, by our assumption, $\boldsymbol{b}$ and $\boldsymbol{b}^{l}$ are distinct. Hence the inference of $\boldsymbol{b} = \boldsymbol{b}^{l}\ (= t_l)$ from $\boldsymbol{b} = \boldsymbol{b}$ must have (up to biunique renaming) the form (19). ■

For the decidability proof we use the description ${}^{s}FD$, which is similar to ${}^{e}FD$ except that the long reentrancies are shortened. Let ${}^{\dagger}FD$ be the f-description obtained from FD by shortening all long reentrancies. Obviously, this f-description is logically equivalent to FD. Then ${}^{s}FD$ is the description that we obtain from ${}^{\dagger}FD$ by replacing (similar to ${}^{e}FD$) each occurrence of a node n of c in ${}^{\dagger}FD$ by $[n]$. This substitution is safe because for any proof of an equality $t = t'$, involving at most n and n', in ${}^{\dagger}FD$ there is a corresponding proof of $(t = t')[n/[n], n'/[n']]$ in ${}^{s}FD$, and vice versa.[15] The following lemma that establishes the decidability of the k-boundedness entails the decidability of the nonconstructivity.

Lemma 5
For any basic LFG grammar it is decidable whether it is k-bounded.

Proof
Let G be an arbitrary basic LFG grammar. We know already that it is decidable whether G is suitable and G's reentrancy-free kernel is k-bounded. Thus assume that G satisfies these conditions. We show that it can be decided whether $G^{\setminus\uparrow=\downarrow}$ satisfies the nonconstructivity condition of Definition 13 by inspecting derivations of (c-structure) depth less than or equal to $3|\mathcal{M}_{\leq k}(R) \times \mathcal{E}| + 1$.
Suppose that $G^{\setminus\uparrow=\downarrow}$ does not satisfy the condition of Definition 13. Then there is a derivation (c, ρ) of a terminal string s from S with f-description FD and clash-free $FD_{\setminus\mathcal{R}}$ in $G^{\setminus\uparrow=\downarrow}$, and $[n] = [n']$, with $[n]$ distinct from $[n']$, can be derived from ${}^{s}FD_{\setminus\mathcal{A}}$ by a proof of the form (19). Now assume that the depth of the derivation exceeds this upper bound and we could not shrink (c, ρ) without invalidating the provability of $[n] = [n']$. Then there must be at least four (distinct) nonterminal nodes $\bar{n}^{i_1}$, .., $\bar{n}^{i_4}$ on a path of

15 We demonstrate this for a proof of an equation of the form $(n\ \sigma) = (n'\ \sigma')$. For equations of the form $(n\ \sigma) = v'$, $v = (n'\ \sigma')$, and $v = v'$ the proof is similar. Obviously, for any proof of $(n\ \sigma) = (n'\ \sigma')$ in ${}^{\dagger}FD$ we obtain a proof of $([n]\ \sigma) = ([n']\ \sigma')$ in ${}^{s}FD$ just by replacing the equivalence classes for the nodes.
Now consider a proof of $([n]\ \sigma) = ([n']\ \sigma')$ of the form (18) in ${}^{s}FD$. Choose a canonical representative from each class and substitute the classes by their representatives. For each premise $(\bar{n}\ \chi) = (\bar{n}'\ \chi')$ (with $([\bar{n}]\ \chi) = ([\bar{n}']\ \chi')$ in ${}^{s}FD$) that is not in ${}^{\dagger}FD$, there must be an equation $(\check{n}\ \chi) = (\check{n}'\ \chi')$ in ${}^{\dagger}FD$ with $\check{n} \in [\bar{n}]$ and $\check{n}' \in [\bar{n}']$. Since $\check{n} = \bar{n}$ and $\check{n}' = \bar{n}'$ follow from $FD_{\setminus\mathcal{R}}$, we obtain a proof in ${}^{\dagger}FD$ by replacing each such premise $(\bar{n}\ \chi) = (\bar{n}'\ \chi')$ by a proof of $(\bar{n}\ \chi) = (\bar{n}'\ \chi')$ from $(\check{n}\ \chi) = (\check{n}'\ \chi')$ where $\check{n}$ is rewritten to $\bar{n}$ and $\check{n}'$ to $\bar{n}'$ by a sequence of substitutions all justified by equations in $FD_{\setminus\mathcal{R}}$.
Note that this result holds also for ${}^{\dagger}FD_{\setminus\mathcal{A}}$ and ${}^{s}FD_{\setminus\mathcal{A}}$, because the function-assigning annotations that generate the equivalence classes are also included in the atom-free kernel.

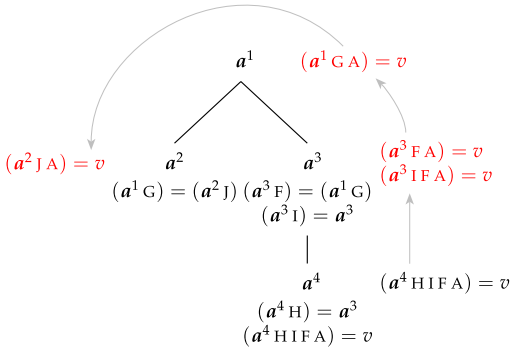


Figure 7
Feature propagation through reentrancies. The derived equations are shown in red. Here, $(a^4\,\mathrm{H}) = a^3$ derives $(a^3\,\mathrm{I\,F\,A}) = v$ from $(a^4\,\mathrm{H\,I\,F\,A}) = v$, $(a^3\,\mathrm{I}) = a^3$ shrinks $(a^3\,\mathrm{I\,F\,A}) = v$ to $(a^3\,\mathrm{F\,A}) = v$, and the reentrancies $(a^3\,\mathrm{F}) = (a^1\,\mathrm{G})$ and $(a^1\,\mathrm{G}) = (a^2\,\mathrm{J})$ produce $(a^2\,\mathrm{J\,A}) = v$ from $(a^3\,\mathrm{F\,A}) = v$.

length greater than $3|\mathcal{M}_{\leq k}(R) \times \mathcal{E}|$, with $\bar{n}^{i_j}$ dominating $\bar{n}^{i_{j+1}}$ and $f^m([\bar{n}^{i_j}]) = f^m([\bar{n}^{i_{j+1}}])$ $(j = 1, .., 3)$, and for any pair $[\bar{n}^{i_j}]$, $[\bar{n}^{i_{j+1}}]$, the shrink operation $sh^{1-1}_{(c,\rho)}$ would break up the proof of $[n] = [n']$ by eliminating necessary right-hand side premises. Now recall that two distinct nodes can be related in a single equation only if the nodes stand in a mother–daughter relationship and that, by definition of $sh^{1-1}_{(c,\rho)}$, the nodes of $[\bar{n}^{i_1}]$, .., $[\bar{n}^{i_4}]$ are licensed by the same rules and can therefore provide the same premises. Thus, the right-hand side premises must originate from equivalence classes from a subpath $\bar{n}..\bar{n}'$ of the path from $\bar{n}^{i_1}$ to $\bar{n}^{i_4}$ with $\bar{n}$ properly dominating $\bar{n}^{i_2}$ and $\bar{n}^{i_3}$ properly dominating $\bar{n}'$. (If they would originate only from equivalence classes from the path from $\bar{n}^{i_2}$ to $\bar{n}^{i_3}$, then we could apply $sh^{1-1}_{(c,\rho)}$ to $[\bar{n}^{i_1}]$, $[\bar{n}^{i_2}]$ and $[\bar{n}^{i_3}]$, $[\bar{n}^{i_4}]$ and $[n] = [n']$ would still be provable.) From the properties recalled above, it also follows that we would obtain a shorter canonical proof of $[n] = [n']$ if no class-valued premise of the proof (the first and last premise in (19)) is eliminated. Because there are three distinct pairs but only two class-valued premises, for at least one pair the provability of $[n] = [n']$ must be preserved under $sh^{1-1}_{(c,\rho)}$, in contradiction to our assumption.[16] ■

If we apply the closure to ${}^{s}FD$, instead of ${}^{e}FD_{\backslash\mathcal{R}}$, then atomic-valued information is not only inherited downward (as in the case of ${}^{e}FD_{\backslash\mathcal{R}}$), but reentrancies can cause it to propagate upward, shrink, or distribute across equivalence classes. An illustration of the propagation process is given in Figure 7. However, the term length of the atomic-valued equations in $\overline{{}^{s}FD}$ can again not exceed ℓ, because all long reentrancies have been shortened ($|\sigma'| \leq 1$). The following lemma shows that clash-freeness of any f-description FD can be determined just by considering for each node class all atomic- and class-valued equations of $\overline{{}^{s}FD}$ that pertain to that class.

16 If G does not contain reentrancies of the form $(\uparrow\ \mathrm{F\,G}) = (\uparrow\ \mathrm{H})$, $(\uparrow\ \mathrm{F}) = (\uparrow\ \mathrm{H})$, and $(\downarrow\ \mathrm{G}) = (\uparrow\ \mathrm{H})$, then, for the proof in (19), $m = 1$ and $\boldsymbol{a} = \boldsymbol{a}'$ follows from $\boldsymbol{a} = (\boldsymbol{a}^1\,\mathrm{F}^1)$ and $(\boldsymbol{a}^1\,\mathrm{F}^1) = \boldsymbol{a}'$. Thus the node class equality must arise from an undesirable interaction between instantiations of cyclic or function-assigning annotations or annotations of the form $(\downarrow\ \mathrm{G}) = \uparrow$. For such grammars G, the condition of Definition 13 can be decided by inspecting derivations of depth less than or equal to $2|\mathcal{M}_{\leq k}(R) \times \mathcal{E}| + 1$.

Lemma 6
Let FD be a description obtained from short reentrancies and function-assigning and atomic-valued annotations by instantiating the metavariables with classes denoted by $\boldsymbol{a}$, $\boldsymbol{a}^1$, $\boldsymbol{a}^2$, .., $\boldsymbol{b}$, $\boldsymbol{b}^1$, ... *If* $FD \vdash \boldsymbol{a} = \boldsymbol{b}$ *does not hold for two distinct classes* $\boldsymbol{a}$ *and* $\boldsymbol{b}$, *then FD is clash-free iff* $\overline{FD}$ *does not contain equations of the form* $(\boldsymbol{a}\ \sigma) = v$ *and either* $(\boldsymbol{a}\ \sigma) = v'$ $(v \neq v')$, *or* $(\boldsymbol{a}\ \sigma) = \boldsymbol{b}$, *or* $(\boldsymbol{a}\ \sigma\ \sigma') = v'$ *with* σ *and* σ' *both not equal to* ϵ, *for any class* $\boldsymbol{a}$ *occurring in FD.*

Proof
Let *FD* be a description satisfying the conditions of the lemma. If *FD* is clash-free then $\overline{FD}$ clearly satisfies the claim of the lemma. Now suppose a clash would follow from *FD*. We know that there is a proof of the form (18), repeated here for clarity of exposition,

$$\dfrac{\dfrac{\dfrac{t_{m+1} = t_{m+1} \quad e_{m+1}}{t_{m+1} = t_m \quad e_m}}{t_{m+1} = t_{m-1}}}{\begin{array}{c}\ddots \\ \dfrac{\dfrac{t_{m+1} = t_2 \quad e_2}{t_{m+1} = t_1 \quad e_1}}{t_{m+1} = t_0}\end{array}}$$

where (a) t_{m+1} and t_0 are either two distinct atomic values v and v', or (b) one is a class $\boldsymbol{b}$ and the other is an atomic value v, or (c) t_0 is a term of the form $(v\ \sigma)$, $\sigma \neq \epsilon$, and t_{m+1} is a term or subterm occurring in *FD*. From arguments used in the proof of Lemma 4, we can assume that

(i) the right-hand side terms are pairwise distinct,

(ii) none of the terms t_i, $i = 1, .., m$, is a class or atomic value and the premises e_j, $j = 2, .., m$, are included in $FD_{\backslash \mathcal{A}}$, because, under assumption (i), we would otherwise obtain a shorter proof of a clash or a violation of the condition of the lemma (thus the term length decreases or increases by at most one attribute at each rewriting step from t_m to t_1),

(iii) the length of the terms $t_m, .., t_1$ does not increase and then decrease.

Given these properties, the proof of the lemma can be completed as follows.
In case (a), t_{m+1} and e_{m+1} must have the form v' and $v' = (\boldsymbol{a}^m\ \sigma^m)$, and t_1 and e_1 the form $(\boldsymbol{a}^1\ \sigma^1)$ and $(\boldsymbol{a}^1\ \sigma^1) = v$. Let t_j be one of the shortest terms (this exists because of (iii)) and suppose it has the form $(\boldsymbol{a}\ \sigma)$. Then $(\boldsymbol{a}\ \sigma) = v$ and $(\boldsymbol{a}\ \sigma) = v'$ must be in $\overline{FD}$ because of (ii) and (iii).
Case (b) is similar to case (a). However, because t_m (or t_1) is a one-attribute term and thus one of the shortest terms, we can here simply assume $j = m$ (or $j = 1$).
In case (c), t_0 has the form $(v\ \sigma)$, with $\sigma \neq \epsilon$. Thus t_1 and e_1 must have the form $(\boldsymbol{a}^1\ \zeta\ \sigma)$ and $(\boldsymbol{a}^1\ \zeta) = v$. Let t_j be one of the shortest terms that contains σ, i.e., $t_j = (\boldsymbol{a}\ \sigma'\ \sigma)$. Without loss of generality, we can assume $\sigma' \neq \epsilon$, because otherwise $e_j, .., e_2$ must rewrite $\boldsymbol{a}$ to $(\boldsymbol{a}^1\ \zeta)$ and there must be a shorter proof of $\boldsymbol{a} = v$. Hence, by (ii), the terms $t_{j+1}, .., t_{m+1}$ cannot be shorter than t_j. Because t_j contains at least two attributes, t_{m+1} must be a (sub)term occurring in an atomic-valued equation $v' = (t_{m+1}\ \xi)$. Thus, $(\boldsymbol{a}\ \sigma'\ \sigma\ \xi) = v'$ and $(\boldsymbol{a}\ \sigma') = v$ must be in $\overline{FD}$ because of (ii) and (iii). ■

The weak equivalence of k-bounded LFGs and k-LCFRSs (that we prove in Section 5.2) crucially depends on the ability to decide clash-freeness of every derivable f-description FD in a k-bounded LFG $G^{\uparrow=\downarrow}$ through local clash tests on the closure of ${}^{s}FD$ as stated in Lemma 6. This is what enables us to simulate all the conditions for correct LFG derivation with a finite set of LCFRS predicate symbols and a finite set of LCFRS productions, and thus to rely on the finite control of the rule-by-rule predicate matching process of LCFRS derivation to produce the strings in $L(G)$.

5. k-Bounded LFGs and k-LCFRSs

In this section we recall the formal definition of linear context-free rewriting systems. Then we establish the (weak) equivalence of k-bounded LFGs and k-LCFRSs by proving that for each k-bounded LFG there is a weakly equivalent k-LCFRS. The proof is constructive in that it provides a procedure for constructing for any k-bounded LFG G a weakly equivalent k-LCFRS G'. The other direction of the equivalence follows trivially from Seki et al.'s (1993) result for finite-copying LFGs (which are properly included in the class of k-bounded LFG grammars). Thereafter, we refine the LCFRS grammar-construction algorithm so that the f-structure that is assigned to a derived string s in a k-bounded LFG G can be read out (in linear time) from the derivation of s in the corresponding k-LCFRS G'. Finally, we demonstrate that LCFRS equivalence extends to k-bounded LFG grammars that make use of the additional descriptive devices proposed by Kaplan and Bresnan (1982) and still in common use.

For the constructions in this section, we assume that the k-bounded LFG grammars have only unannotated terminals and epsilons. This is without loss of generality, because k-bounded LFGs can easily be transformed into equivalent k-bounded grammars in this more restricted format. The transformation is done by replacing each annotated terminal symbol (a, D) in the right-hand side of a rule by (A^a, D) and adding a rule $A^a \rightarrow a$, where A^a is a new unique nonterminal, for each terminal a. Annotated epsilons are eliminated by replacing each rule of the form $A \rightarrow (\epsilon, D)$ by two rules of the form $A \rightarrow (A^\epsilon, D)$, $A^\epsilon \rightarrow \epsilon$, where A^ϵ is a new dummy preterminal. Obviously, the resulting grammar derives the same string/f-structure mapping as the original.

For the $\uparrow = \downarrow$-free 2-bounded LFG grammar with the rules given in (13) we then obtain a grammar with the rules in (20).

(20) a. $\mathrm{S} \rightarrow \underset{(\uparrow \mathrm{S}) = \downarrow}{\mathrm{NP}} \; \underset{(\uparrow \mathrm{O}) = \downarrow}{\mathrm{NP}} \; \underset{(\uparrow \mathrm{X}) = \downarrow}{\overline{\mathrm{VP}}} \; \underset{\substack{(\uparrow \mathrm{P}) = \mathrm{V} \\ (\uparrow \mathrm{X\,S}) = (\uparrow \mathrm{O}) \\ (\uparrow \mathrm{S\,A}) = \mathrm{C}}}{\mathrm{A^v}} \; \underset{(\uparrow \mathrm{X}) = \downarrow}{\overline{\mathrm{V}}}$

b. $\overline{\mathrm{VP}} \rightarrow \underset{(\uparrow \mathrm{O}) = \downarrow}{\mathrm{NP}} \; \underset{(\uparrow \mathrm{X}) = \downarrow}{\overline{\mathrm{VP}}}$

c. $\overline{\mathrm{VP}} \rightarrow \underset{\substack{(\uparrow \mathrm{O}) = \downarrow \\ (\uparrow \mathrm{X\,X}) = \#}}{\mathrm{NP}}$

d. $\overline{\mathrm{V}} \rightarrow \underset{\substack{(\uparrow \mathrm{P}) = \mathrm{V} \\ (\uparrow \mathrm{X\,S}) = (\uparrow \mathrm{O})}}{\mathrm{A^v}} \; \underset{(\uparrow \mathrm{X}) = \downarrow}{\overline{\mathrm{V}}}$

e. $\mathrm{NP} \rightarrow \underset{\substack{(\uparrow \mathrm{P}) = \mathrm{N} \\ (\uparrow \mathrm{A}) = \mathrm{C}}}{\mathrm{A^n}}$

f. $\overline{\mathrm{V}} \rightarrow \underset{\substack{(\uparrow \mathrm{P}) = \mathrm{V} \\ (\uparrow \mathrm{X}) = \#}}{\mathrm{A^v}}$

g. $\mathrm{A^v} \rightarrow \mathrm{v}$

h. $\mathrm{A^n} \rightarrow \mathrm{n}$

5.1 Linear Context-Free Rewriting Systems (LCFRSs)

We now introduce linear context-free rewriting systems and their languages (see Kallmeyer [2013] for a survey on LCFRSs). For the definition we assume that $\mathcal{V}$ is a set of variables.

Definition 19
A (***k*-)LCFRS** G is a 4-tuple (N, T, S, R) where N is a finite set of nonterminal categories (also called predicate symbols), each with a fixed arity $\mathfrak{a}(A)$ ($1 \leq \mathfrak{a}(A) \leq k$), T is a finite set of terminal symbols ($T \cap \mathcal{V} = \varnothing$), $\mathrm{S} \in N$ is the start symbol with $\mathfrak{a}(\mathrm{S}) = 1$, and R is a finite set of rules of the form

$$A(\alpha_1, .., \alpha_{\mathfrak{a}(A)}) \rightarrow A^1(Y^1_1, .., Y^1_{\mathfrak{a}(A^1)})..A^m(Y^m_1, .., Y^m_{\mathfrak{a}(A^m)})$$

with $m \geq 0$, A and A^j in N, $Y^j_i \in \mathcal{V}$, for $1 \leq j \leq m$ and $1 \leq i \leq \mathfrak{a}(A^j)$, and $\alpha_l \in (T \cup \mathcal{V})^*$, for $1 \leq l \leq \mathfrak{a}(A)$. Moreover, every variable occurring in a rule $r \in R$ occurs exactly once in the left-hand side and exactly once in the right-hand side of r.

An instantiation of an LCFRS rule r is obtained by replacing all variables in r with terminal strings. This is formally defined as follows.

Definition 20
Let $r = A(\alpha_1, .., \alpha_{\mathfrak{a}(A)}) \rightarrow A^1(Y^1_1, .., Y^1_{\mathfrak{a}(A^1)})..A^m(Y^m_1, .., Y^m_{\mathfrak{a}(A^m)})$. Then r' is an **instantiation** of r if there is an $\eta = \{(Y^j_i, s^j_i) \mid s^j_i \in T^* \text{ and } 1 \leq j \leq m \text{ and } 1 \leq i \leq \mathfrak{a}(A^j)\}$ and $r' = r[\eta]$.

In the following we abbreviate $A(\alpha_1, .., \alpha_{\mathfrak{a}(A)})$ ($\alpha_l \in (T \cup \mathcal{V})^*$) by $A(\vec{\alpha})$ if there is no need to refer to the particular arguments. The language of an LCFRS G is defined in terms of the set of instantiated nonterminals that G derives.

Definition 21
Let G be an LCFRS. Then the set of instantiated nonterminals derivable by G is the smallest set $L_N(G)$ satisfying the following conditions:

(i) if $A(\vec{\alpha}) \rightarrow \epsilon \in R$, then $A(\vec{\alpha}) \in L_N(G)$,

(ii) if $A^1(\vec{\alpha}^1), .., A^m(\vec{\alpha}^m)$ are in $L_N(G)$ and $A(\vec{\alpha}) \rightarrow A^1(\vec{\alpha}^1)..A^m(\vec{\alpha}^m)$ is an instantiation of a rule in R, then $A(\vec{\alpha}) \in L_N(G)$.

The language of G is the set

$$L(G) = \{s \mid \mathrm{S}(s) \in L_N(G)\}.$$

Note that the language defined in this way does not depend on the order of predicates in the right sides of rules. Predicate order is merely a notational convenience.

As an illustration, consider the 3-LCFRS $G = (\{\mathrm{S}, \mathrm{A}\}, \{\mathrm{a}, \mathrm{b}\}, \mathrm{S}, R)$, with R as given in (21), which derives the double copy language $\{www \mid w \in \{\mathrm{a}, \mathrm{b}\}^+\}$.

(21) $r_1 = \mathrm{S}(XYZ) \rightarrow \mathrm{A}(X, Y, Z)$
$r_2 = \mathrm{A}(\mathrm{a}X, \mathrm{a}Y, \mathrm{a}Z) \rightarrow \mathrm{A}(X, Y, Z)$ $\quad$ $r_3 = \mathrm{A}(\mathrm{b}X, \mathrm{b}Y, \mathrm{b}Z) \rightarrow \mathrm{A}(X, Y, Z)$
$r_4 = \mathrm{A}(\mathrm{a}, \mathrm{a}, \mathrm{a}) \rightarrow \epsilon$ $\quad$ $r_5 = \mathrm{A}(\mathrm{b}, \mathrm{b}, \mathrm{b}) \rightarrow \epsilon$

This LCFRS derives, for example, the string abbabbabb because the instantiated nonterminal S(abbabbabb) is derivable as shown in (22).

(22) $L_N(G)$

A(b, b, b)	A(b, b, b) $\rightarrow \epsilon \in R$
A(bb, bb, bb)	A(bb, bb, bb) $\rightarrow$ A(b, b, b) instantiation of r_3
A(abb, abb, abb)	A(abb, abb, abb) $\rightarrow$ A(bb, bb, bb) instantiation of r_2
S(abbabbabb)	S(abbabbabb) $\rightarrow$ A(abb, abb, abb) instantiation of r_1

Derivability in LCFRSs can also be trivially restated in terms of labeled trees and licensing rule mappings, as an analogy to derivations in basic LFGs.

Definition 22
A pair (c, ρ) consisting of a labeled tree c and a mapping ρ from the nodes of c into R is a **derivation** of a tuple of strings $(s^1, .., s^j)$, $s^i \in T^*$, $1 \leq i \leq j$, from B in LCFRS G iff

(i) for each terminal node n with label $A(\vec{\alpha})$, $\rho_n = A(\vec{\alpha}) \rightarrow \epsilon$,

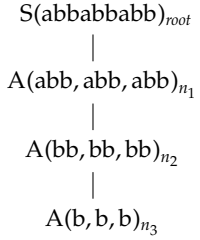
(ii) for each nonterminal node n with label $A(\vec{\alpha})$ and $dts(n) = n_1..n_m$ with labels $A^1(\vec{\alpha}^1), .., A^m(\vec{\alpha}^m)$, $A(\vec{\alpha}) \rightarrow A^1(\vec{\alpha}^1)..A^m(\vec{\alpha}^m)$ is an instantiation of ρ_n,

(iii) the label of *root* is $B(s^1, .., s^j)$.

A terminal string s is **derivable** in LCFRS G iff there is a derivation of (s) from S. Because a terminal string s is derivable in G iff S$(s) \in L_N(G)$ the language of G is equal to $\{s \in T^* \mid s \text{ is derivable in } G\}$.

The simple derivation tree for the string abbabbabb in the LCFRS comprising the rules in (21) is depicted in (23a) and its licensing rule mapping is given in (23b).

(23) a.



b. $\rho_{root} = r_1$, $\rho_{n_1} = r_2$, $\rho_{n_2} = r_3$, $\rho_{n_3} = r_5$

5.2 Weak Equivalence of *k*-Bounded LFGs and *k*-LCFRSs

For a given k-bounded LFG G we construct a k-LCFRS G' and then show that $L(G) = L(G')$. The rules of G' are based on the rules and categories of $G^{\setminus\uparrow=\downarrow}$ and are constructed so as to simulate the derivations of that grammar. We have established that node equivalence classes $[n]$ in $G^{\setminus\uparrow=\downarrow}$ arise only through zipper unifications and that equivalent nodes are therefore at the same level of the c-structure. We also know that the daughters of equivalent nodes are themselves equivalent if and only if the rules that expand their mothers have the same function assignment annotations, and that those rules must contain function assignments that can be used to shorten any long reentrancies. Our construction makes use of these properties.

We first illustrate the LCFRS rule construction for LFG rules with nonterminal or preterminal daughters.[17] Suppose that an equivalence class $[n]$ (of not more than k nodes) arises in the course of a $G^{\uparrow=\downarrow}$ derivation. Because the nodes in $[n]$ are at the same level, they can be ordered according to the linear precedence relation $\prec$ on their terminal yields. We can thus assign to $[n]$ a sequence of nonterminals Γ that label its ordered nodes ($|\Gamma| = |[n]|$), and use that label sequence in building the left-hand predicates of candidate LCFRS rules that might simulate how daughters of nodes with those labels are derived. We hypothesize rule sequences ϱ that could have been used to expand the nonterminals of Γ in a valid $G^{\uparrow=\downarrow}$ derivation (i.e., a sequence ϱ of the form $\varrho_1 = \Gamma_1 \rightarrow \psi^1, .., \varrho_{|\Gamma|} = \Gamma_{|\Gamma|} \rightarrow \psi^{|\Gamma|}$). To take an example from the 2-bounded LFG grammar in (20), one such rule sequence for the sequence $\Gamma = \overline{\text{VP}}\,\overline{\text{V}}$ is given in (24).

(24) $\varrho_1 = \overline{\text{VP}} \rightarrow \underset{(\uparrow \text{O}) = \downarrow}{\text{NP}} \quad \underset{(\uparrow \text{X}) = \downarrow}{\overline{\text{VP}}} \qquad \varrho_2 = \overline{\text{V}} \rightarrow \underset{\substack{(\uparrow \text{P}) = \text{V} \\ (\uparrow \text{X S}) = (\uparrow \text{O})}}{\text{A}^{\text{V}}} \quad \underset{(\uparrow \text{X}) = \downarrow}{\overline{\text{V}}}$

We test such a candidate rule sequence to determine whether its rule annotations give rise to locally clash-free descriptions and whether they include the function assignments needed to shorten all long reentrancies. Rule sequences that do not meet these conditions cannot participate in a valid $G^{\uparrow=\downarrow}$ derivation and can safely be removed from consideration.

The test is conducted on the description FD_ϱ. This description is constructed by instantiating $\uparrow$ in the annotations of ϱ_l by a constant b^l and $\downarrow$ in the daughter annotations of ϱ_l by constants of a sequence β^l whose length coincides with that of the right-hand side of ϱ_l. The b's and the constants in the concatenation $\beta = \beta^1..\beta^{|\Gamma|}$ must be pairwise distinct so that we make the same discriminations as the nodes of a valid derivation from these mothers. Because the mother nodes are all equivalent by hypothesis we include in FD_ϱ equations $b^1 = b^i$, for all $i = 2, .., |\Gamma|$. For this example we can instantiate the mothers with b^1 and b^2 and the daughter sequences with $\beta^1 = \beta^1_1\beta^1_2 = YZ$ and $\beta^2 = \beta^2_1\beta^2_2 = QR$. For the rule sequence in (24) and these mother and daughter constants, FD_ϱ is the description in (25).

(25) $\{b^1 = b^2, (b^1\ \text{O}) = Y, (b^1\ \text{X}) = Z, (b^2\ \text{P}) = \text{V}, (b^2\ \text{X S}) = (b^2\ \text{O}), (b^2\ \text{X}) = R\}$

This description is clash-free and the long reentrancy $(b^2\ \text{X S}) = (b^2\ \text{O})$ can be shortened either through $(b^2\ \text{X}) = R$, or $(b^1\ \text{X}) = Z$ and $b^1 = b^2$. This rule sequence thus remains as a candidate for further consideration. The constants used for daughter instantiation will also serve as the variables in constructed LCFRS rules (and henceforth we refer to them interchangeably also as variables).

The right-side predicates account for the possibility that FD_ϱ entails the equality of instantiating variables for separate nonterminal daughters of the rules in ϱ, that is, $FD_\varrho \vdash \beta_i = \beta_j$. The equality of variables Z and R, for example, follows from the instantiated assignments $(b^1\ \text{X}) = Z$ and $(b^2\ \text{X}) = R$, and the equality $b^1 = b^2$. The equalities between daughter variables give rise to equivalence classes $[\beta_i]$ corresponding to the node equivalence classes of a simulated derivation. We also include classes $[\beta_i] = \{\beta_i\}$ for variables that do not appear in FD_ϱ because $\downarrow$ does not appear in the annotations of corresponding nonterminal daughter nodes. We impose an order on the variables within each equivalence class according to their order in β (which reflects the c-structure

17 Because of the elimination of annotated terminals the LFG rules are partitioned into unannotated terminal rules and nonterminal rules with nonterminal or preterminal daughters.

precedence relation $\prec$ of corresponding daughters). That is, we define a precedence order between the variables of β by $\beta_i \prec \beta_j$ if $i < j$.

In our construction, the variables range over the yields of the right-side nonterminals in simulated derivations of $G^{\uparrow=\downarrow}$. The left- and right-side LCFRS predicates are set up so that a derivable string instantiation of the ith argument of a predicate is the yield of the ith nonterminal of its predicate symbol in a $G^{\uparrow=\downarrow}$ derivation.

We first construct one right-side predicate for each of the $[\beta_i]$ equivalence classes. The predicate symbol of the predicate is the concatenation of nonterminal categories corresponding to the ordered variables in $[\beta_i]$ and its arguments are just the ordered variables of the class. This depends on the fact that there is a one-to-one alignment of the variable sequence β and the sequence of categories projected from the concatenated daughters of the ϱ rules. The category projection *Cat* is defined for every annotated category (X, D) by $Cat(X, D) = X$ and then extended in the natural way to strings of annotated categories. The alignment can be represented as a function θ that assigns to the ith element of β the ith element of $\phi = Cat(\psi^1..\psi^{|\Gamma|})$, that is, $\theta(\beta_i) = \phi_i$ for $i = 1, .., |\beta|$. Applied to the concatenation of the right-hand sides of the rules ϱ_1 and ϱ_2 the *Cat* projection yields $\mathrm{NP}\,\overline{\mathrm{VP}}\,\mathrm{A}^{\mathrm{v}}\,\overline{\mathrm{V}}$, and, for $\beta = YZQR$, θ is given by $\theta(Y) = \mathrm{NP}$, $\theta(Z) = \overline{\mathrm{VP}}$, $\theta(Q) = \mathrm{A}^{\mathrm{v}}$, and $\theta(R) = \overline{\mathrm{V}}$. For the equivalence classes of our example, $\{\beta_1\} = \{Y\}$, $\{\beta_2, \beta_4\} = \{Z, R\}$ (with $Z \prec R$), and $\{\beta_3\} = \{Q\}$, we then obtain the predicates $\theta(Y)(Y) = \mathrm{NP}(Y)$, $\theta(Z)\theta(R)(Z, R) = \overline{\mathrm{VP}}\,\overline{\mathrm{V}}(Z, R)$, and $\theta(Q)(Q) = \mathrm{A}^{\mathrm{v}}(Q)$.

In LCFRS rules, the arguments of the left-side predicates describe how their yields are assembled from the yields instantiating the variables of the right-side predicates. We thus construct the lth argument of the left-side predicate Γ by substituting for each daughter in the *Cat* projection of the right-hand side of rule ϱ_l the variable that occurs in the corresponding position in β^l. This ensures that the LCFRS rule assembles for each Γ_l the terminal string instantiation of the lth argument in accordance with $Cat(\psi^l)$. For our example we thus obtain the left-side predicate $\overline{\mathrm{VP}}\,\overline{\mathrm{V}}(YZ, QR)$.

Because the right-side predicates of LCFRS rules must be arranged in a sequence, we extend the relation $\prec$ on variables to equivalence classes by ordering classes according to their least elements. Using the extended $\prec$ to arrange the corresponding predicates, we obtain the rule (26).

(26) $\overline{\mathrm{VP}}\,\overline{\mathrm{V}}(YZ, QR) \rightarrow \mathrm{NP}(Y)\ \ \overline{\mathrm{VP}}\,\overline{\mathrm{V}}(Z, R)\ \ \mathrm{A}^{\mathrm{v}}(Q)$

We have already determined that FD_ϱ is locally clash-free. But we have not yet ensured that clashes do not arise from atomic values that other steps of a simulated derivation might associate with an equivalence class of nodes or with their daughter classes. To control for these (and other) potential clashes of the simulated derivations, we refine the left-side predicate symbol Γ and the right-side predicate symbols Γ^j with sets of atomic-valued equations E, E_j of $\mathcal{E}$ that could arise in any valid LFG derivation as $h_{\overline{FD}}$ values for the corresponding equivalence classes. One such refinement of the skeletal rule (26) is shown in (27).

(27) $\overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{\{(* \mathrm{P}) = \mathrm{V}\}}(YZ, QR) \rightarrow \mathrm{NP}_{\{(* \mathrm{P}) = \mathrm{N}, (* \mathrm{A}) = \mathrm{C}\}}(Y)\ \ \overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{\{(* \mathrm{P}) = \mathrm{V}\}}(Z, R)\ \ \mathrm{A}^{\mathrm{v}}_{\varnothing}(Q)$

Because these refinements must give rise to clash-free descriptions we require them to be mutually compatible and consistent with the annotations of the hypothesized ϱ rule sequences. This constraint is implemented by checking for clash-freeness the f-description FD' created by combining ${}^{s}FD_\varrho$ with the descriptions formed by appropriately instantiating the selected refinements. Specifically, we substitute $[b^1]$ for $*$ in the

refinement of the left-side predicate, and we replace $*$ in the refinement of each right-side predicate by the predicate's associated equivalence class of variables.

The atomic-valued information that the E refinements associate with the rule equivalence classes is also required to be invariant under closure of FD', that is, the refinement of each equivalence class is assumed to record exactly the information that $\overline{FD'}$ associates with that class. Under this requirement, the clash-freeness of the FD' of each rule and Lemma 6 ensure that the finite control of the rule-by-rule predicate matching process of LCFRS derivations will simulate LFG derivations of G that have clash-free f-descriptions.

Finally, for the rules that expand preterminal categories into unannotated terminals $A^a \to a$ and epsilons $A^\epsilon \to \epsilon$, we add trivial LCFRS rules of the form $A^a_\varnothing(a) \to \epsilon$ and $A^\epsilon_\varnothing(\epsilon) \to \epsilon$, and we include a particular set of start rules. The start rules expand the root predicate $\mathrm{S}'(X)$ of the new grammar to unary predicates $\mathrm{S}_E(X)$ consisting of the original start symbol and one clash-free set of atomic-valued equations E in $\mathcal{E}$.

More precisely, the construction is as follows.

Definition 23
For any k-bounded LFG G with $G^{\uparrow=\downarrow} = (N, T, \mathrm{S}, R)$ we construct a k-LCFRS $G' = (N', T, \mathrm{S}', R')$ in the following way. The collection of nonterminals is

$$N' = \{\mathrm{S}'\} \cup \{\Gamma_E \mid \Gamma \in N^+, |\Gamma| \leq k, \text{ and } E \in \mathcal{E}\}$$

where $\mathfrak{a}(\mathrm{S}') = 1$ and $\mathfrak{a}(\Gamma_E) = |\Gamma|$. R' includes start rules $\mathrm{S}'(X) \to \mathrm{S}_E(X)$ for any $E \in \mathcal{E}$, a rule $A^a_\varnothing(a) \to \epsilon$ for any terminal rule $A^a \to a$, and a rule $A^\epsilon_\varnothing(\epsilon) \to \epsilon$ for any epsilon rule $A^\epsilon \to \epsilon$. The other rules of R' are created as follows. Let ϱ be a sequence of nonterminal rules $\varrho_1 = \Gamma_1 \to \psi^1, .., \varrho_{|\Gamma|} = \Gamma_{|\Gamma|} \to \psi^{|\Gamma|}$ with $|\Gamma| \leq k$. Now let $b^1, .., b^{|\Gamma|}$ be constants and β^l be sequences of constants with $|\beta^l| = |\psi^l|$, for $l = 1, .., |\Gamma|$, such that the b's and the constants in the concatenation $\beta = \beta^1..\beta^{|\Gamma|}$ are pairwise distinct. Set

$$FD_\varrho = \bigcup_{l=1}^{|\Gamma|} Inst(\varrho_l, (b^l, \beta^l)) \cup \{b^1 = b^i \mid i = 2, .., |\Gamma|\}.$$

If FD_ϱ is clash-free and the long reentrancies in FD_ϱ can be shortened through the function assignments and b^1 equations, we consider the classes $[b^1] = \{b^1, .., b^{|\Gamma|}\}$ and $[\beta_i] = \{\beta_j \mid FD_\varrho \vdash \beta_i = \beta_j\}$ together with singleton classes $[\beta_i] = \{\beta_i\}$ for any constant β_i that does not occur in FD_ϱ. Let $\prec$ be a precedence order on the constants of β defined by $\beta_i \prec \beta_j$ if $i < j$ and set $[\beta_i] \prec [\beta_j]$ if there is an element in $[\beta_i]$ that precedes all elements of $[\beta_j]$ in β. Now let $\boldsymbol{b} = [b^1]$ and $\boldsymbol{b}_1, .., \boldsymbol{b}_m$ be the classes $[\beta_i]$ ordered according to $\prec$. For every collection of elements E, $E_1, .., E_m$ of $\mathcal{E}$ large enough so that the description

$$FD' = {}^{\mathrm{s}}FD_\varrho \cup E[*/\boldsymbol{b}] \cup \bigcup_{j=1}^{m} E_j[*/\boldsymbol{b}_j]$$

is clash-free and $h_{\overline{FD'}}(\boldsymbol{b}) = E$ and $h_{\overline{FD'}}(\boldsymbol{b}_j) = E_j$, for $j = 1, .., m$, R' contains a rule of the form

$$\Gamma_E(\beta_1, .., \beta_{|\Gamma|}) \to \Gamma^1_{E_1}(Y^1_1, .., Y^1_{|\Gamma^1|})..\Gamma^m_{E_m}(Y^m_1, .., Y^m_{|\Gamma^m|})$$

where Y^j_i is the ith greatest element of $\boldsymbol{b}_j$ and $\Gamma^j_i = \theta(Y^j_i)$.

This construction produces for the 2-bounded LFG grammar with the rules in (20) an LCFRS that includes the rules in (28). These rules generate the same language as the original LFG. The construction creates many other categories and rules that are not shown here because they are either useless or redundant. Useless rules cannot participate in the simulation of any LFG derivation while redundant ones simulate only the same derivations as other rules and categories in the grammar.

(28) $\mathrm{S}'(X) \to \mathrm{S}_E(X)$

$\mathrm{S}_E(XYZQR) \to \mathrm{NP}_{E_1}(X)\ \ \mathrm{NP}_{E_1}(Y)\ \ \overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_2}(Z,R)\ \ \mathrm{A}^{\mathrm{v}}_{\varnothing}(Q)$

$\mathrm{S}_E(XYZQR) \to \mathrm{NP}_{E_1}(X)\ \ \mathrm{NP}_{E_1}(Y)\ \ \overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_3}(Z,R)\ \ \mathrm{A}^{\mathrm{v}}_{\varnothing}(Q)$

$\overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_2}(YZ,QR) \to \mathrm{NP}_{E_1}(Y)\ \ \overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_2}(Z,R)\ \ \mathrm{A}^{\mathrm{v}}_{\varnothing}(Q)$

$\overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_2}(YZ,QR) \to \mathrm{NP}_{E_1}(Y)\ \ \overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_3}(Z,R)\ \ \mathrm{A}^{\mathrm{v}}_{\varnothing}(Q)$

$\overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_3}(Y,QR) \to \mathrm{NP}_{E_1}(Y)\ \ \mathrm{A}^{\mathrm{v}}_{\varnothing}(Q)\ \ \overline{\mathrm{V}}_{E_4}(R)$

$\mathrm{NP}_{E_1}(X) \to \mathrm{A}^{\mathrm{n}}_{\varnothing}(X)$

$\overline{\mathrm{V}}_{E_4}(X) \to \mathrm{A}^{\mathrm{v}}_{\varnothing}(X)$

$\mathrm{A}^{\mathrm{n}}_{\varnothing}(\mathrm{n}) \to \epsilon$

$\mathrm{A}^{\mathrm{v}}_{\varnothing}(\mathrm{v}) \to \epsilon$

with $E = \{(*\ \mathrm{P}) = \mathrm{V}, (*\ \mathrm{S\ A}) = \mathrm{C}\}$, $E_1 = \{(*\ \mathrm{P}) = \mathrm{N}, (*\ \mathrm{A}) = \mathrm{C}\}$, $E_2 = \{(*\ \mathrm{P}) = \mathrm{V}\}$,
$E_3 = \{(*\ \mathrm{P}) = \mathrm{V}, (*\ \mathrm{X\ X}) = \#\}$, and $E_4 = \{(*\ \mathrm{P}) = \mathrm{V}, (*\ \mathrm{X}) = \#\}$.

Weak equivalence then follows by bottom–up induction on the depth of the derivations.

Lemma 7
For any k-bounded LFG G there exists a k-LCFRS G′ with L(G) = L(G′).

Proof
Let G' be the k-LCFRS constructed for a k-bounded LFG G as in Definition 23.
We first show $L(G) \subseteq L(G')$. Let (c, ρ) be a derivation of a terminal string s from S with f-description FD and f-structure F in $G^{\backslash\uparrow=\downarrow}$. Consider the equivalence classes $[n]$ for the nonterminal nodes n of c. Obviously, $|[n]| \leq k$ for each n. Assign to each $[n]$ a predicate $\Gamma_E(Y_1, .., Y_{|\Gamma|})$, with $\mathfrak{a}(\Gamma) = |[n]|$, $|\Gamma| = |[n]|$, and $E = h_{\overline{FD}}([n])$, where Y_l is the lth greatest element of $[n]$ and Γ_l its label. The assignment of predicates to node classes, as well as the inductive step, are illustrated in Figure 8. We show bottom–up for each $[n]$ that $\Gamma_E(s_1, .., s_{|\Gamma|}) \in L_N(G')$ where s_l is the yield of Y_l. Because $G^{\backslash\uparrow=\downarrow}$ has no annotated terminals and the annotations of the preterminals do not contain $\downarrow$, $[n]$ must be a singleton set and $h_{\overline{FD}}([n]) = \varnothing$, for any preterminal node n. Then ρ_{Y_1} must be of the form $A^a \to a$ or $A^\epsilon \to \epsilon$, and $A^a_\varnothing(a) \to \epsilon$ or $A^\epsilon_\varnothing(\epsilon) \to \epsilon$ must be in R' by construction of G'. Hence $A^a_\varnothing(a)$ or $A^\epsilon_\varnothing(\epsilon)$ is in $L_N(G')$. Now suppose that $[n]$ does not contain only preterminal nodes. Let $[n^1], .., [n^m]$ be the equivalence classes of the nonterminal daughters of the nodes in $[n]$ ordered according to their least elements and $\Gamma^1_{E_1}(Y^1_1, .., Y^1_{|\Gamma^1|}), .., \Gamma^m_{E_m}(Y^m_1, .., Y^m_{|\Gamma^m|})$ be the corresponding predicates. Obviously, the rule $r = \Gamma_E(dts(Y_1), .., dts(Y_{|\Gamma|})) \to \Gamma^1_{E_1}(Y^1_1, .., Y^1_{|\Gamma^1|})..\Gamma^m_{E_m}(Y^m_1, .., Y^m_{|\Gamma^m|})$ must be in R' if we construct LCFRS rules for the sequence $\varrho_1 = \rho_{Y_1}, .., \varrho_{|\Gamma|} = \rho_{Y_{|\Gamma|}}$ according to Definition 23 with $\beta^l = dts(Y_l)$ and $E, E_1, .., E_m$ as specified above. We know by inductive hypothesis

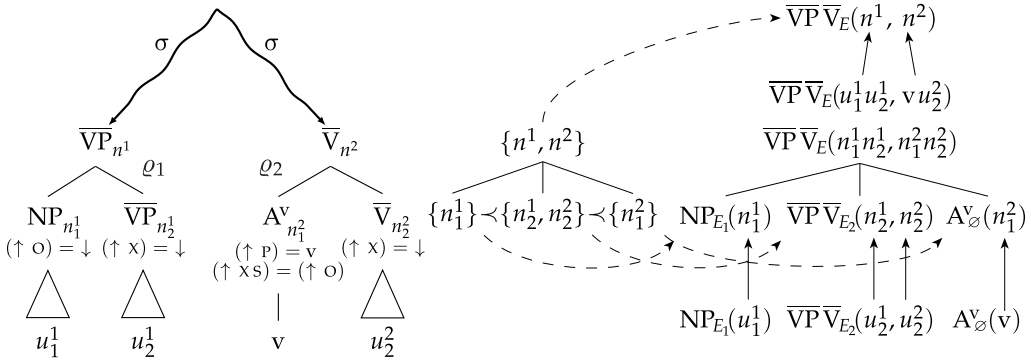


Figure 8
Illustration of the assignment of predicates to node equivalence classes and of the inductive step in the $L(G) \subseteq L(G')$ argument for an LFG derivation with the rule sequence ϱ in (24). The assignment is indicated by the dashed arrows and the E components are $E = h_{\overline{s_{FD}}}(\{n^1, n^2\})$, $E_1 = h_{\overline{s_{FD}}}(\{n_1^1\})$, and $E_2 = h_{\overline{s_{FD}}}(\{n_2^1, n_2^2\})$. The LCFRS rule $\overline{\mathrm{VP}}\,\overline{\mathrm{V}}_E(n_1^1\ n_2^1, n_1^2 n_2^2) \to \mathrm{NP}_{E_1}(n_1^1)\ \overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_2}(n_2^1, n_2^2)\ \ \mathrm{A}^{\mathrm{v}}_{\varnothing}(n_1^2)$ is constructed for ϱ with $\beta^1 = n_1^1 n_2^1$ and $\beta^2 = n_1^2 n_2^2$ (and $E, E_1, .., E_m$ as defined above). The derived terminal strings are u_1^1, u_2^1, v, and u_2^2 and the instantiation of the variables by terminal strings is indicated by solid straight arrows.

that $\Gamma^j_{E_j}(s^j_1, .., s^j_{|\Gamma^j|}) \in L_N(G')$ where s^j_i is the yield of Y^j_i. Define a substitution η by $\eta(Y^j_i) = s^j_i$ and let $\Gamma_E(dts(Y_1), .., dts(Y_{|\Gamma|}))[\eta] = \Gamma_E(s_1, .., s_{|\Gamma|})$. Then $\Gamma_E(s_1, .., s_{|\Gamma|}) \in L_N(G')$ and the claim that s_l is the yield of Y_l holds by construction of r (see the illustration on the right-hand side of Figure 8). Hence, $\mathrm{S}_E(s)$ is in $L_N(G')$ and $\mathrm{S}'(s)$ in $L_N(G')$ by $\mathrm{S}'(X) \to \mathrm{S}_E(X)$.

We then establish $L(G') \subseteq L(G)$. Let (c', ρ') be a derivation of (s) from S' in G'. We show bottom–up for each node n' (except *root*) of c' with label $\Gamma_E(s_1, .., s_{|\Gamma|})$ that there are derivations of s_l from Γ_l with root node n^l and f-description FD_l in $G^{\setminus\uparrow=\downarrow}$ (as defined in Definition 3) such that all long reentrancies in $FD = \bigcup_{l=1}^{|\Gamma|} FD_l \cup \{n^1 = n^i \mid i = 2, .., |\Gamma|\}$ can be shortened using the n^1 equations and function assignments. Along the induction, we define descriptions ${}^{\S}_{\mathcal{E}}FD$ (${}^{s}FD$ enlarged by the instantiated E components of the derivation in G') such that (a) ${}^{s}FD \subseteq {}^{\S}_{\mathcal{E}}FD$, (b) $h_{\overline{{}^{\S}_{\mathcal{E}}FD}}([n^1]) = E$, and (c) ${}^{\S}_{\mathcal{E}}FD$ is clash-free. The clash-freeness of FD (and FD_l) then follows from (a) and (c). If n' is a terminal node then its label has the form $A^a_{\varnothing}(a)$ or $A^{\epsilon}_{\varnothing}(\epsilon)$, and $\rho_{n'}$ is $A^a_{\varnothing}(a) \to \epsilon$ or $A^{\epsilon}_{\varnothing}(\epsilon) \to \epsilon$. Hence $A^a \to a$ or $A^{\epsilon} \to \epsilon$ is in R by construction of G' and there must trivially be a derivation of the required form. Moreover, with ${}^{\S}_{\mathcal{E}}FD = \varnothing$ also the other claims hold trivially because FD and E are empty. Now suppose that n' is a nonterminal node with label $\Gamma_E(s_1, .., s_{|\Gamma|})$ and daughters $n'_1, .., n'_m$ with labels $\Gamma^1_{E_1}(s^1_1, .., s^1_{|\Gamma^1|}), .., \Gamma^m_{E_m}(s^m_1, .., s^m_{|\Gamma^m|})$, $\Gamma_E(s_1, .., s_{|\Gamma|}) \to \Gamma^1_{E_1}(s^1_1, .., s^1_{|\Gamma^1|})..\Gamma^m_{E_m}(s^m_1, .., s^m_{|\Gamma^m|})$ is an instantiation of $\rho'_{n'} = \Gamma_E(\beta_1, .., \beta_{|\Gamma|}) \to \Gamma^1_{E_1}(Y^1_1, .., Y^1_{|\Gamma^1|})..\Gamma^m_{E_m}(Y^m_1, .., Y^m_{|\Gamma^m|})$, and the claims hold for the daughter predicates by inductive hypothesis. The inductive step is illustrated in Figure 9. Without loss of generality we can assume that Y^j_i is the root node of the derivation of s^j_i for each $j = 1, .., m$ and $i = 1, .., |\Gamma^j|$ (rename the nodes if necessary). Now let n^l be new (root) nodes and assume $n^l = b^l$ $(l = 1, .., |\Gamma|)$. We obtain derivations

of s_l from Γ_l with root n^l and f-description FD_l, $l = 1, .., |\Gamma|$, by licensing n^l through $\Gamma_l \to \psi^l$ and by expanding the ith occurrence in $Cat(\psi^l)$ using the derivation with root β_i^l ($i = 1, .., |\psi^l|$). Then each FD_l is the union of $Inst(\Gamma_l \to \psi^l, (b^l, \beta^l))$ and the f-descriptions of the derivations with root β_i^l, and FD must be $FD_\varrho \cup \bigcup_{\substack{j=1,..,m \\ i=1,..,|\Gamma^j|}} FD_i^j$. It follows directly from the construction of $\rho'_{n'}$ and the inductive hypothesis that all long reentrancies in FD can be shortened. From the construction of G' and the derivations, it is easy to see that ${}^sFD = {}^sFD_\varrho \cup \bigcup_{j=1}^{m} {}^sFD^j$. Now let ${}^s_{\mathcal{E}}FD = {}^sFD_\varrho \cup E[*/[n^1]] \cup \bigcup_{j=1}^{m} {}^s_{\mathcal{E}}FD^j$. Then, obviously, ${}^sFD \subseteq {}^s_{\mathcal{E}}FD$. To show (b) and (c), recall that two nodes can be related in a single instantiated annotation only if the nodes stand in a mother–daughter relationship or if they are identical. Thus, ${}^sFD_\varrho$ and each ${}^s_{\mathcal{E}}FD^j$ can share only the class $[Y_1^j]$ and only equations of the form $([Y_1^j] \; \mathrm{F} \; \sigma) = v$ can factor into the closure of ${}^sFD_\varrho \cup E[*/[n^1]]$ and ${}^s_{\mathcal{E}}FD^j$. Set $FD' = {}^sFD_\varrho \cup E[*/[n^1]] \cup \bigcup_{j=1}^{m} E_j[*/[Y_1^j]]$ as in the definition of G'. Since $h_{\overline{{}^s_{\mathcal{E}}FD^j}}([Y_1^j]) = E_j$ by inductive hypothesis and $h_{\overline{FD'}}([Y_1^j]) = E_j$, $j = 1, .., m$, and $h_{\overline{FD'}}([n^1]) = E$ by construction of $\rho'_{n'}$, we obtain (b) $h_{\overline{{}^s_{\mathcal{E}}FD}}([n^1]) = E$; moreover, for each $j = 1, .., m$, the restriction of $h_{\overline{{}^s_{\mathcal{E}}FD}}$ to the classes in ${}^s_{\mathcal{E}}FD^j$ is equal to $h_{\overline{{}^s_{\mathcal{E}}FD^j}}$. Hence, the h values are preserved and the clash-freeness of ${}^s_{\mathcal{E}}FD$ follows by Lemma 6 from the clash-freeness of FD' and the ${}^s_{\mathcal{E}}FD^j$, which concludes the induction step. Thus $G^{\setminus\uparrow=\downarrow}$ derives s from S if G' derives (s) from S_E for some $E \in \mathcal{E}$, and hence $s \in L(G^{\setminus\uparrow=\downarrow})$ if $s \in L(G')$. ■

Lemma 8 follows trivially from the corresponding result for finite-copying LFGs (Seki et al. 1993).

Lemma 8
For any k-LCFRS G there exists a k-bounded LFG G′ with L(G) = L(G′).

Hence we have Theorem 1.

Theorem 1
k-Bounded LFGs are weakly equivalent to k-LCFRSs.

Because the recognition and emptiness problems are decidable for LCFRSs, the same must hold for k-bounded LFGs.

Corollary 3
For k-bounded LFGs the recognition and emptiness problems are decidable.

5.3 The Construction of Strongly Equivalent LCFRSs

It is straightforward to augment the right-side predicate symbols of weakly equivalent LCFRSs with an additional component that records the annotations of the nonterminal daughters from which they originate. We can thus acquire access to the f-structure if we arbitrarily hypothesize at each left-side predicate symbol a subset of the set of $G^{\setminus\uparrow=\downarrow}$ annotations. The predicates in N' other than the start predicate S′ will have the

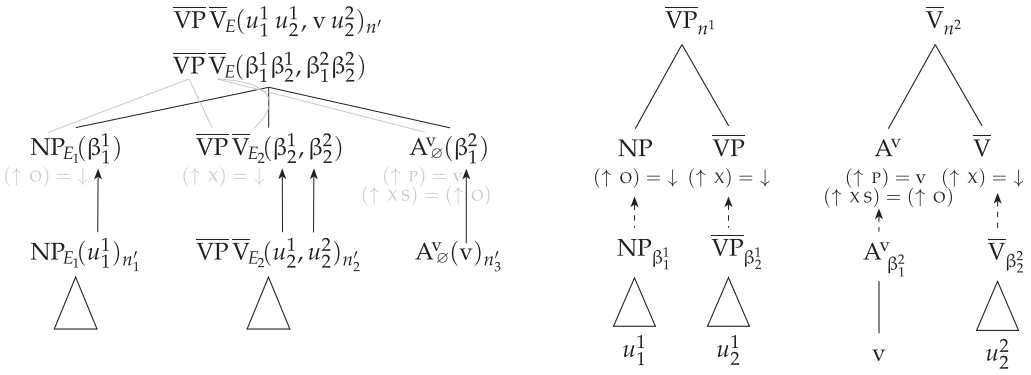


Figure 9
Illustration of the inductive step in the proof of $L(G') \subseteq L(G)$ for an LCFRS derivation with a rule obtained from the sequence ϱ in (24). The immediate dominance properties and the daughter annotations of the original LFG rules are indicated in gray.

form $\Gamma_E{:}D$ where D is a set of admissible (non-trivial) annotations. To illustrate the rule construction, consider as an example for a predicate symbol $\Gamma_E{:}D$ the refinement $\overline{\text{VP}}\,\overline{\text{V}}_{\{(* \text{ P}) = \text{V}\}}{:}\{(\uparrow \text{X}) = \downarrow\}$ of the predicate symbol $\overline{\text{VP}}\,\overline{\text{V}}_{\{(* \text{ P}) = \text{V}\}}$ that we used in our previous illustration of the grammar construction. Here, the D component records that the co-referring categories $\overline{\text{VP}}$ and $\overline{\text{V}}$ are (both) annotated by $(\uparrow \text{X}) = \downarrow$ in $G^{\setminus\uparrow=\downarrow}$. Again for the sequence of rules in (24) we proceed along the lines of Definition 23. The annotation component of the right-hand side predicates then just collects the annotations of the right-hand side daughter categories from which they are created. For our example rules we thus obtain, for example, the LCFRS rule in (29).

(29) $\overline{\text{VP}}\,\overline{\text{V}}_{\{(* \text{ P}) = \text{V}\}}{:}\{(\uparrow \text{X}) = \downarrow\}(YZ, QR) \rightarrow$
$\text{NP}_{\left\{\begin{smallmatrix}(* \text{ P}) = \text{N},\\ (* \text{ A}) = \text{C}\end{smallmatrix}\right\}}{:}\{(\uparrow \text{O}) = \downarrow\}(Y)\ \ \overline{\text{VP}}\,\overline{\text{V}}_{\{(* \text{ P}) = \text{V}\}}{:}\{(\uparrow \text{X}) = \downarrow\}(Z, R)\ \ \text{A}^{\text{v}}_{\varnothing}{:}\left\{\begin{smallmatrix}(\uparrow \text{P}) = \text{V},\\ (\uparrow \text{X S}) = (\uparrow \text{O})\end{smallmatrix}\right\}(Q)$

This component is empty for the right-hand side of the start rules $\text{S}'(X) \rightarrow \text{S}_E{:}\varnothing(X)$ (because the root category of an LFG derivation is not annotated).

The additional properties of strongly equivalent LCFRSs that enable f-structure recovery are made more precise in the following definition. In the definition we use the annotation projection An to access the annotations of the LFG rules. The An projection is defined for every annotated category (X, D) by $An(X, D) = D$.

Definition 24
Let G be a k-bounded LFG. For $G^{\setminus\uparrow=\downarrow} = (N, T, \text{S}, R)$ we construct a k-LCFRS $G' = (N', T, \text{S}', R')$ as follows. The set of nonterminals N' includes S' and all symbols of the form $\Gamma_E{:}D$, where $\Gamma \in N^+$, $|\Gamma| \leq k$, $E \in \mathcal{E}$, and D is a subset of the set of $G^{\setminus\uparrow=\downarrow}$ annotations. The arity of the nonterminals is defined as in Definition 23. The rule set R' includes start rules $\text{S}'(X) \rightarrow \text{S}_E{:}\varnothing(X)$ for every $E \in \mathcal{E}$, and rules $A^a_{\varnothing}{:}D(a) \rightarrow \epsilon / A^{\epsilon}_{\varnothing}{:}D(\epsilon) \rightarrow \epsilon$, for any terminal rule $A^a \rightarrow a$/epsilon rule $A^{\epsilon} \rightarrow \epsilon$ and any subset D of $G^{\setminus\uparrow=\downarrow}$ annotations. For any sequence ϱ of nonterminal rules $\varrho_1 = \Gamma_1 \rightarrow \psi^1, .., \varrho_{|\Gamma|} = \Gamma_{|\Gamma|} \rightarrow \psi^{|\Gamma|}$ satisfying the conditions of Definition 23 and any subset of $G^{\setminus\uparrow=\downarrow}$ annotations D, we construct rules of the form

$$\Gamma_E{:}D(\beta_1,..,\beta_{|\Gamma|}) \rightarrow \Gamma^1_{E_1}{:}D_1(Y^1_1,..,Y^1_{|\Gamma^1|})..\Gamma^m_{E_m}{:}D_m(Y^m_1,..,Y^m_{|\Gamma^m|})$$

with Y^j_i, E, and E_j as defined there, and $D_j = \bigcup\{An(\psi_k) \mid \beta_k \in \boldsymbol{b}_j\}$ with $\psi = \psi^1..\psi^{|\Gamma|}$.

For the rules in (20) this construction produces the rules in (30). All other candidate rules are again either useless or redundant.

(30) a. $S'(X) \rightarrow S_E{:}\varnothing(X)$

b. $S_E{:}\varnothing(XYZQR) \rightarrow$
$\mathrm{NP}_{E_1}{:}\{(\uparrow \mathrm{S}) = \downarrow\}(X)\ \ \mathrm{NP}_{E_1}{:}\{(\uparrow \mathrm{O}) = \downarrow\}(Y)\ \ \overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_2}{:}\{(\uparrow \mathrm{X}) = \downarrow\}(Z,R)\ \ \mathrm{A}^{\mathrm{v}}_{\varnothing}{:}\left\{\begin{array}{c}(\uparrow \mathrm{P}) = \mathrm{V},\\ (\uparrow \mathrm{XS}) = (\uparrow \mathrm{O}),\\ (\uparrow \mathrm{SA}) = \mathrm{C}\end{array}\right\}(Q)$

c. $S_E{:}\varnothing(XYZQR) \rightarrow$
$\mathrm{NP}_{E_1}{:}\{(\uparrow \mathrm{S}) = \downarrow\}(X)\ \ \mathrm{NP}_{E_1}{:}\{(\uparrow \mathrm{O}) = \downarrow\}(Y)\ \ \overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_3}{:}\{(\uparrow \mathrm{X}) = \downarrow\}(Z,R)\ \ \mathrm{A}^{\mathrm{v}}_{\varnothing}{:}\left\{\begin{array}{c}(\uparrow \mathrm{P}) = \mathrm{V},\\ (\uparrow \mathrm{XS}) = (\uparrow \mathrm{O}),\\ (\uparrow \mathrm{SA}) = \mathrm{C}\end{array}\right\}(Q)$

d. $\overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_2}{:}\{(\uparrow \mathrm{X}) = \downarrow\}(YZ,QR) \rightarrow$
$\mathrm{NP}_{E_1}{:}\{(\uparrow \mathrm{O}) = \downarrow\}(Y)\ \ \overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_2}{:}\{(\uparrow \mathrm{X}) = \downarrow\}(Z,R)\ \ \mathrm{A}^{\mathrm{v}}_{\varnothing}{:}\left\{\begin{array}{c}(\uparrow \mathrm{P}) = \mathrm{V},\\ (\uparrow \mathrm{XS}) = (\uparrow \mathrm{O})\end{array}\right\}(Q)$

e. $\overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_2}{:}\{(\uparrow \mathrm{X}) = \downarrow\}(YZ,QR) \rightarrow$
$\mathrm{NP}_{E_1}{:}\{(\uparrow \mathrm{O}) = \downarrow\}(Y)\ \ \overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_3}{:}\{(\uparrow \mathrm{X}) = \downarrow\}(Z,R)\ \ \mathrm{A}^{\mathrm{v}}_{\varnothing}{:}\left\{\begin{array}{c}(\uparrow \mathrm{P}) = \mathrm{V},\\ (\uparrow \mathrm{XS}) = (\uparrow \mathrm{O})\end{array}\right\}(Q)$

f. $\overline{\mathrm{VP}}\,\overline{\mathrm{V}}_{E_3}{:}\{(\uparrow \mathrm{X}) = \downarrow\}(Y,QR) \rightarrow$
$\mathrm{NP}_{E_1}{:}\left\{\begin{array}{c}(\uparrow \mathrm{O}) = \downarrow,\\ (\uparrow \mathrm{XX}) = \#\end{array}\right\}(Y)\ \ \mathrm{A}^{\mathrm{v}}_{\varnothing}{:}\left\{\begin{array}{c}(\uparrow \mathrm{P}) = \mathrm{V},\\ (\uparrow \mathrm{XS}) = (\uparrow \mathrm{O})\end{array}\right\}(Q)\ \ \overline{\mathrm{V}}_{E_4}{:}\{(\uparrow \mathrm{X}) = \downarrow\}(R)$

g. $\mathrm{NP}_{E_1}{:}\{(\uparrow \mathrm{S}) = \downarrow\}(X) \rightarrow \mathrm{A}^{\mathrm{n}}_{\varnothing}{:}\left\{\begin{array}{c}(\uparrow \mathrm{P}) = \mathrm{N},\\ (\uparrow \mathrm{A}) = \mathrm{C}\end{array}\right\}(X)$

h. $\mathrm{NP}_{E_1}{:}\{(\uparrow \mathrm{O}) = \downarrow\}(X) \rightarrow \mathrm{A}^{\mathrm{n}}_{\varnothing}{:}\left\{\begin{array}{c}(\uparrow \mathrm{P}) = \mathrm{N},\\ (\uparrow \mathrm{A}) = \mathrm{C}\end{array}\right\}(X)$

i. $\mathrm{NP}_{E_1}{:}\left\{\begin{array}{c}(\uparrow \mathrm{O}) = \downarrow,\\ (\uparrow \mathrm{XX}) = \#\end{array}\right\}(X) \rightarrow \mathrm{A}^{\mathrm{n}}_{\varnothing}{:}\left\{\begin{array}{c}(\uparrow \mathrm{P}) = \mathrm{N},\\ (\uparrow \mathrm{A}) = \mathrm{C}\end{array}\right\}(X)$

j. $\overline{\mathrm{V}}_{E_4}{:}\{(\uparrow \mathrm{X}) = \downarrow\}(X) \rightarrow \mathrm{A}^{\mathrm{v}}_{\varnothing}{:}\left\{\begin{array}{c}(\uparrow \mathrm{P}) = \mathrm{V},\\ (\uparrow \mathrm{X}) = \#\end{array}\right\}(X)$

k. $\mathrm{A}^{\mathrm{n}}_{\varnothing}{:}\left\{\begin{array}{c}(\uparrow \mathrm{P}) = \mathrm{N},\\ (\uparrow \mathrm{A}) = \mathrm{C}\end{array}\right\}(\mathrm{n}) \rightarrow \epsilon$

l. $\mathrm{A}^{\mathrm{v}}_{\varnothing}{:}\left\{\begin{array}{c}(\uparrow \mathrm{P}) = \mathrm{V},\\ (\uparrow \mathrm{XS}) = (\uparrow \mathrm{O}),\\ (\uparrow \mathrm{SA}) = \mathrm{C}\end{array}\right\}(\mathrm{v}) \rightarrow \epsilon$

m. $\mathrm{A}^{\mathrm{v}}_{\varnothing}{:}\left\{\begin{array}{c}(\uparrow \mathrm{P}) = \mathrm{V},\\ (\uparrow \mathrm{XS}) = (\uparrow \mathrm{O})\end{array}\right\}(\mathrm{v}) \rightarrow \epsilon$

n. $\mathrm{A}^{\mathrm{v}}_{\varnothing}{:}\left\{\begin{array}{c}(\uparrow \mathrm{P}) = \mathrm{V},\\ (\uparrow \mathrm{X}) = \#\end{array}\right\}(\mathrm{v}) \rightarrow \epsilon$

with $E = \{(* \mathrm{P}) = \mathrm{V}, (* \mathrm{SA}) = \mathrm{C}\}$, $E_1 = \{(* \mathrm{P}) = \mathrm{N}, (* \mathrm{A}) = \mathrm{C}\}$, $E_2 = \{(* \mathrm{P}) = \mathrm{V}\}$,
$E_3 = \{(* \mathrm{P}) = \mathrm{V}, (* \mathrm{XX}) = \#\}$, and $E_4 = \{(* \mathrm{P}) = \mathrm{V}, (* \mathrm{X}) = \#\}$.

With these elaborated rules an f-structure can be recovered from an LCFRS derivation that has been formulated in terms of derivation trees as per Definition 22. The nodes of a derivation tree can then be used to instantiate the $\uparrow$ and $\downarrow$ metavariables in the annotations in exactly the same way as in derivations with LFG grammars. The following extension of this definition then provides the required access to the f-structure. The instantiation function *Inst* is defined for the refined LCFRS rules as specified in Definition 2.

Definition 25
Let G be a k-bounded LFG and G' be the k-LCFRS obtained from G as described in Definition 24. A pair (c, ρ) consisting of a labeled tree c and a mapping ρ from the nodes

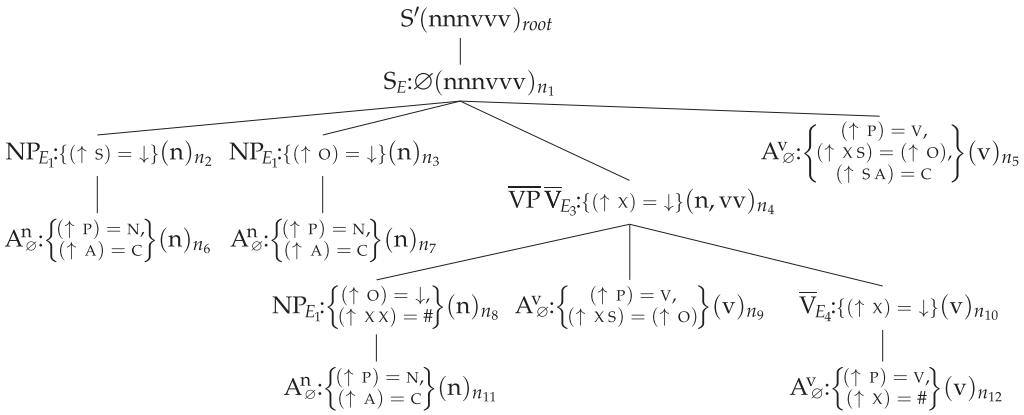


Figure 10
The derivation tree of nnnvvv in the LCFRS with the rules in (30).

of c into R' is a **derivation** of a tuple of terminal strings $(s^1, .., s^j)$ from B **with functional description** FD **and f-structure** F in G' iff

(i) (c, ρ) is a derivation of $(s^1, .., s^j)$ from B,

(ii) $FD = \bigcup\{Inst(\rho_n, (n, dts(n))) \mid n \in Dom(\rho)$ and n is a nonterminal node$\}$,

(iii) $F = M|_{\mathcal{A}\cup\mathcal{V}}$, where M is a minimal model of FD.

A terminal string s is **derivable** with f-structure F in G' ($\Delta_{G'}(s, F)$) iff there is a derivation of (s) from S′ with F (and some f-description FD) in G'.

As an illustration, consider the derivation of nnnvvv in the LCFRS comprising the rules in (30). The derivation tree of this derivation is depicted in Figure 10 and its licensing rule mapping is given in (31).

(31) $\rho_{root} = (30a)$, $\rho_{n_1} = (30c)$, $\rho_{n_2} = (30g)$, $\rho_{n_3} = (30h)$, $\rho_{n_4} = (30f)$, $\rho_{n_5} = (30l)$, $\rho_{n_6} = (30k)$, $\rho_{n_7} = (30k)$, $\rho_{n_8} = (30i)$, $\rho_{n_9} = (30m)$, $\rho_{n_{10}} = (30j)$, $\rho_{n_{11}} = (30k)$, $\rho_{n_{12}} = (30n)$

From the f-description depicted in (32), it is easy to see that this grammar associates with nnnvvv the same f-structure as the 2-bounded LFG grammar in (1) from which it was created.

(32)
$$\left\{\begin{array}{lll}
(n_1\ \mathrm{S}) = n_2, & (n_2\ \mathrm{P}) = \mathrm{N}, & \\
 & (n_2\ \mathrm{A}) = \mathrm{C}, & \\
(n_1\ \mathrm{O}) = n_3, & (n_3\ \mathrm{P}) = \mathrm{N}, & \\
 & (n_3\ \mathrm{A}) = \mathrm{C}, & \\
(n_1\ \mathrm{X}) = n_4, & (n_4\ \mathrm{XX}) = \#, & \\
 & (n_4\ \mathrm{O}) = n_8, & (n_8\ \mathrm{P}) = \mathrm{N}, \\
 & & (n_8\ \mathrm{A}) = \mathrm{C}, \\
 & (n_4\ \mathrm{X}) = n_{10}, & (n_{10}\ \mathrm{P}) = \mathrm{V}, \\
 & & (n_{10}\ \mathrm{X}) = \#, \\
 & (n_4\ \mathrm{P}) = \mathrm{V}, & \\
 & (n_4\ \mathrm{XS}) = (n_4\ \mathrm{O}), & \\
(n_1\ \mathrm{P}) = \mathrm{V}, & & \\
(n_1\ \mathrm{SA}) = \mathrm{C}, & & \\
(n_1\ \mathrm{XS}) = (n_1\ \mathrm{O}) & &
\end{array}\right\}$$

Along the lines of the argument used in the proof of Lemma 7 it is straightforward to see that G and G' derive the same string/f-structure mapping.

Theorem 2
Let G be a k-bounded LFG and G' be the k-LCFRS constructed for G as in Definition 24. Then $\Delta_G = \Delta_{G'}$.

Because the f-description of every derivation in G' is clash-free, the f-structure can be read out from any particular derivation in linear time.

5.4 Other Descriptive Devices

The LFG formalism as originally proposed by Kaplan and Bresnan (1982) includes formal devices beyond the primitive attribute and value mechanisms of the basic subclass, and LFG theory contemplates configurations of annotated rules that do not obviously meet the restrictions we depend on. In this section we show how those devices and configurations can be accommodated in the k-bounded framework, either by conversion to equivalent rules and annotations of the more restricted form or by extensions to the k-LCFRS translation procedure.

C-Structure Regular Predicates and Boolean Combinations of Elementary Annotations. The full LFG notation allows functional requirements to be stated as arbitrary Boolean combinations of basic annotations. It also allows the right-hand sides of c-structure rules to denote arbitrary regular languages over annotated categories. Rules with the richer notation, including Kleene-star iterations, can be linearized into collections of productions all of which are in conventional context-free format and have no internal disjunctions and which together define the same string/f-structure mapping as a grammar encoded in the original, linguistically more expressive, notation (Wedekind and Kaplan 2012). Although sufficient in principle to establish the LCFRS equivalences, such a conversion may not be possible in practice (see the discussion in the next section).

Kleene-star iterations are conventionally translated into equivalent right-linear expansions that make use of trivial annotations. For example, the iteration of the annotated category (A, D) in (33a) is replaced by a new optional trivially-annotated category A^D, and the rule (33b) is introduced to properly expand that category. (In (33b) we use the traditional parenthetic notation for optionality, to collapse into a single rule the expansion with or without the recursive category.)

(33) a. $X \rightarrow \phi \underset{D}{A^*} \psi$

b. $A^D \rightarrow \underset{D}{A} \ (A^D)$ $\uparrow = \downarrow$

Obviously, grammars with such recursions would violate condition (ii) of Definition 10, because this by itself expresses no bounds on the trivially annotated dominance chains. There must be additional properties of the grammar that prevent the annotations in D from creating unbounded zippers. We discuss this problem in the next section.

Adjuncts. In LFG the f-structures of modifiers that serve as adjuncts of a predicate are represented as elements of the set-valued attribute ADJUNCTS. Formally, this is accomplished by annotations of the form $\downarrow \in (\uparrow \text{ ADJUNCTS})$ stating that the value of the ADJUNCTS attribute is a set that includes as one of its elements the f-structure associated with the annotated constituent.[18] Set elements behave formally like daughters without

18 The following argument applies equally to open adjuncts (XADJUNCTS).

function assignments. They therefore can be handled by our original construction without further modification. However, because LFG allows the rules to contain Kleene-starred adjunct phrases as in (34), the number of adjuncts is potentially unbounded.

(34) XP^*
$\downarrow \in (\uparrow \text{ ADJUNCTS})$

The modifier Kleene-star iteration can be translated to an equivalent right-linear expansion as above. The trivial-annotations in this particular right-linear expansion do not have to be eliminated in order to locally disclose structure sharing through zipper unification: Derivations of set-valued adjuncts cannot create or add to zippers. We substitute for the recursive trivial a variant $\uparrow \doteq \downarrow$ that is opaque to the trivial-elimination procedure and thus not removed from the grammar. It is carried along by the LCFRS translation algorithm (using an extended ${}^{s}FD$ closure supporting equations of the form $\boldsymbol{a} \doteq \boldsymbol{b}$)[19] and interpreted as a node identity only during f-structure construction.

Positive and Negative Constraints. LFG annotations are divided into two classes: defining and constraining annotations. The instantiated constraining annotations are evaluated once all instantiated defining annotations have been processed and a minimal model (of the defining statements) has been constructed. The constraining devices introduced by Kaplan and Bresnan (1982) are constraining equations and inequalities, and existential and negative existential constraints. If a constraining statement is contained in an f-description FD, it is evaluated against a minimal model M of the defining statements of FD in the following way: $M \models t =_c t'$ iff $M \models t = t'$ (constraining equation), $M \models t$ iff $\exists t'(M \models t = t')$ (existential constraint), $M \models \neg\gamma$ iff $M \not\models \gamma$ (negation of a constraining or defining statement).

It is fairly straightforward to extend the LCFRS construction in Definition 23 to LFG grammars with negative constraints. If we assume the negative constraints, just as the atom-valued defining statements, to be propagated across the E components, then we can properly account for them by requiring the negative constraints of FD' to be satisfied in a minimal model M of the defining statements of FD'. The extension to grammars that make use of positive constraints is for several reasons far more challenging.

The construction of Definition 23 creates LCFRS rules that simulate the derivations of a corresponding LFG grammar. Consistency of atomic-value information that might be inherited from other rules of an LFG derivation is enforced by matching the predicates of individual LCFRS rules, given that the predicates are refined by E components that hypothesize a collection of potentially inherited values. The E values include those that are necessary to guarantee that derivations contain no clashing atomic values, but as specified they may also include information that is not defined elsewhere in the simulated derivation. Positive constraints thus cannot be tested against these overly large E components. An alternative refinement strategy is exploited to ensure that constraints can be checked against properly limited subsets of information.

Now, the LCFRS predicate symbols are augmented with $\check{E}$ components that record the ℓ-bounded atomic-value information and reentrancies (compactly represented by $(* \ \sigma) = v/(* \ \sigma')$) gathered bottom–up in the simulated derivations. The predicate symbols in LCFRS rules will thus be refined by $\check{E}$ components drawn from the set

$$\mathcal{E}' = \{E \mid E \subseteq \{(* \ \sigma) = v/(* \ \sigma') \mid \sigma, \sigma' \in \mathcal{A}^+ \wedge |\sigma|, |\sigma'| \leq \ell \wedge v \in \mathcal{V}\} \wedge E \text{ is clash-free}\}.$$

19 If $(\boldsymbol{a} \ \sigma) = v \in \overline{FD}$ and $\boldsymbol{a} \doteq \boldsymbol{b} \in FD$, then $(\boldsymbol{b} \ \sigma) = v \in \overline{FD}$.

We build on the specification of Definition 23 substituting $\check{E}$ refinements for the atom-value components of the LCFRS rules constructed there. For the terminal expansions $A^a \to a$ and $A^\epsilon \to \epsilon$, the LCFRS rules have the form $A^a_\varnothing(a) \to \epsilon$ and $A^\epsilon_\varnothing(\epsilon) \to \epsilon$, as in the original procedure.

For an LFG rule sequence $\varrho = \Gamma_1 \to \psi^1..\Gamma_{|\Gamma|} \to \psi^{|\Gamma|}$ whose instantiated class description is ${}^sFD_\varrho$, the LCFRS rules will be of the form

$$\Gamma^{}_{\check{E}}(\beta_1,..,\beta_{|\Gamma|}) \to \Gamma^1_{\check{E}_1}(Y^1_1,..,Y^1_{|\Gamma^1|})..\Gamma^m_{\check{E}_m}(Y^m_1,..,Y^m_{|\Gamma^m|})$$

where the $\check{E}_1,..,\check{E}_m$ components are drawn from $\mathcal{E}'$ such that

(i) $FD' = {}^sFD_\varrho \cup \bigcup_{j=1}^{m} \check{E}_j[*/\boldsymbol{b}_j]$ is clash-free,

(ii) $\check{E} = \{(*\ \sigma) = v/(*\ \sigma') \mid FD' \vdash (\boldsymbol{b}\ \sigma) = v/(\boldsymbol{b}\ \sigma')$ and $|\sigma|, |\sigma'| \leq \ell\}$.

Additionally, the start symbol S' is expanded by rules of the form $S'(X) \to S_{\check{E}}(X)$ where $\check{E}$ is drawn arbitrarily from $\mathcal{E}'$. This construction ensures that the $\check{E}$'s of an LCFRS derivation only contain equations whose node-class instantiations follow from the description ${}^sFD$ of a simulated LFG derivation, a necessary requirement to capture positive constraints.

Given that the positive constraints are ℓ-bounded, the description FD' now provides all the information necessary to determine whether or not a positive constraint in ${}^sFD_\varrho$ is satisfied below the ϱ-expanded nodes of an equivalence class of the simulated derivations.[20] Thus, only positive constraints that are not satisfied in the minimal model of the defining statements of FD' and are hence supposed to be satisfied higher up in the simulated derivations require some special attention.

We augment the LCFRS predicate symbols with an additional component $\check{C}$ that records the constraining equations $(*\ \sigma) =_c v$ and existential constraints $(*\ \sigma)$ $(|\sigma| \leq \ell)$, in the following abbreviated by $(*\ \sigma)[=_c v]$, that are supposed to be satisfied higher up in the simulated derivations. For the terminal expansions and the start rules these components are empty. Thus, these LCFRS rules have the form $A^a_{\varnothing:\varnothing}(a) \to \epsilon$, $A^\epsilon_{\varnothing:\varnothing}(\epsilon) \to \epsilon$, and $S'(X) \to S_{\check{E}:\varnothing}(X)$.

For an LFG rule sequence $\varrho = \Gamma_1 \to \psi^1..\Gamma_{|\Gamma|} \to \psi^{|\Gamma|}$ with instantiated class description ${}^sFD_\varrho$, the LCFRS rules will be of the form

$$\Gamma^{}_{\check{E}:\check{C}}(\beta_1,..,\beta_{|\Gamma|}) \to \Gamma^1_{\check{E}_1:\check{C}_1}(Y^1_1,..,Y^1_{|\Gamma^1|})..\Gamma^m_{\check{E}_m:\check{C}_m}(Y^m_1,..,Y^m_{|\Gamma^m|})$$

where the additional $\check{C}$ component is the smallest set satisfying the following conditions:

(i) if $(\boldsymbol{b}\ \sigma)[=_c v]$ is in ${}^sFD_\varrho$ but not satisfied in a minimal model of the defining statements of FD', then $(*\ \sigma)[=_c v]$ is included in $\check{C}$,

20 Note that the $\check{E}$ components also include reflexive equations of the form $(*\ \sigma) = (*\ \sigma)$ whose class instantiations follow from the ${}^sFD$ of the simulated derivations. This allows it to also account for existential constraints whose satisfying feature paths do not arise from atomic-valued equations.

(ii) if $(\boldsymbol{b}_j\ \sigma')[=_c v]$ is in ${}^{s}FD_{\varrho}$ or $(*\ \sigma')[=_c v]$ in $\check{C}_j$ and $(\boldsymbol{b}_j\ \sigma')[=_c v]$ is not satisfied in a minimal model of the defining statements of FD', then $\check{C}$ includes $(*\ \sigma)[=_c v]$ with $FD' \vdash (\boldsymbol{b}\ \sigma)[=_c v] \equiv (\boldsymbol{b}_j\ \sigma')[=_c v]$ and $|\sigma| \leq \ell$.

In this extended LCFRS construction, the unsatisfied constraints in ${}^{s}FD_{\varrho}$ and the daughter $\check{C}_j$ are propagated to the mother $\check{C}$. Because the $\check{C}$ components of the start rules are empty, all constraints in the lower $\check{C}$ components and instantiated class descriptions ${}^{s}FD_{\varrho}$ must be satisfied in the simulated derivations.

Completeness and Coherence. The Completeness and Coherence Conditions are the formal devices in LFG that enforce the subcategorization requirements of individual predicates. The governable grammatical functions that can and must appear in an f-structure are specified in its local semantic form, the single quoted values of its PRED atttribute. We can take these requirements into account by interpreting the subcategorization frame as a collection of existential constraints. For completeness, we introduce a positive existential constraint $(\uparrow \text{G})$ for each function G that a semantic form governs. For coherence, we pair with every assignment of a governable function G a positive existential constraint that tests whether $(\uparrow \text{PRED})$ designates a semantic form that subcategorizes for G. The result of these augmentations is a grammar that properly enforces all subcategorization requirements.

Indexing of Semantic Forms. Semantic forms are also indexed or instantiated in the sense that a new and distinct value is created to represent each semantic form as it enters into a derivation (Kaplan and Bresnan 1982). This would result in a clash if a description contains two instantiations $([n]\ \sigma) = s$ where s is a semantic form and not a simple atomic value. We elaborate the conditions of the LCFRS construction to ensure that this property of semantic forms is respected in all simulated derivations. Specifically, we exclude an LCFRS rule if the combination of ${}^{s}FD_{\varrho}$ and $\check{E}_j$ hypotheses that make up its justifying description would include two or more instances of the same semantic form.

6. Practical Considerations

We have shown that for every k-bounded LFG G there is a weakly equivalent k-LCFRS G' and that the f-structures assigned by G can be recovered easily from the derivations of a refinement of G'. We also know that the universal recognition problem for any k-LCFRS G' can be solved in time $\mathcal{O}(|G'| \cdot n^{k\cdot(\mathsf{r}+1)})$ (Seki et al. 1991). The recognition problem is tractable in the mathematical sense—polynomial in the length of the input sentence—but the polynomial term may be dwarfed by the grammar constants. In this formula, $|G'|$ is the number of rules in G', n is the length of the input string, r is the **rank** (the maximum number of nonterminals (predicates) on the right-hand side of G' rules), and the given k is the **fan-out** (the maximum arity). Thus the practical benefit of recognizing a k-bounded LFG language with an equivalent LCFRS depends on the fan-out and rank but also on the size of the LCFRS.

We can establish bounds on the parameters of the general complexity formula ($|G'|$, k, r) as a function of the characteristic properties of natural language grammars.[21] We

21 Without appealing to further linguistic restrictions, G' can be exponentially larger than G (as indicated by the fact that the universal recognition problem for k-bounded LFGs, with attributes drawn from an unbounded set, is NP-complete). Thus, even though k-bounded LFGs are weakly equivalent to k-LCFRSs and k-LCFRSs can be recognized in polynomial time, LCFRS recognition algorithms may, without further restrictions, fail to be practical for equivalent k-bounded LFGs. (This situation is similar to that in

first consider linguistically motivated grammars in the basic formalism we have defined here and thereafter grammars used in the more flexible format that is typically used in linguistic practice.

The translation of a given k-bounded LFG grammar G to an equivalent LCFRS G' is carried out in two phases, and each has an effect on the size of G'. In the first phase the trivial $\uparrow = \downarrow$ annotations are eliminated and in the second phase a collection of LCFRS rules is created for each sequence of up to k rules of $G^{\setminus\uparrow=\downarrow}$.

The first phase, the elimination of trivial annotations from a basic k-bounded G, can by itself result in a grammar $G^{\setminus\uparrow=\downarrow}$ that is much larger than G. The growth in the number and length of the rules depends on the bound h on the height of functional domains in G and the number of alternative rules, the degree of ambiguity, that expand the categories that are trivially-annotated in G. From Definition 10 we know that the size of the nonterminal category set $|N|$ must be an upper bound for h, but that may substantially overestimate what is needed for LCFRS translation of natural language grammars. In linguistically motivated grammars the distribution of trivial-annotations is regulated by the principles of X-bar theory and its structure–function mapping principles (Bresnan 2001, Chapter 6; Dalrymple 2001, Chapter 4). In this (ϵ-free) framework the components of an f-structure unit are introduced recursively by trivially-annotated categories, and the height of a functional domain is effectively bounded by the number of coheads that can associate to a single predicate, the number of discontinuous c-structure phrases that can realize a particular function, and the number of different grammatical functions that an individual predicate can govern. With g denoting the maximum number of governable grammatical functions that can occur together and c denoting the maximum number of cooccurring coheads plus 1 (accounting for the lexical head), the maximum height is given by $kg + c$. Then, because any daughter sequences must be promoted for any trivially annotated category in the worst case, the size of $G^{\setminus\uparrow=\downarrow}$ cannot exceed $|G|^{kg+c+1}$. (Increasing the exponent by one accounts for the trivial-free rules obtained from sequences shorter than $kg + c$.)

The g, c, and k parameters are typically rather small. For instance, the lexicons of the broad-coverage, commercial-grade Pargram grammars for English and German (approximately 25,000 words each) have no word that subcategorizes for more than four grammatical functions and very few words allow even that many (in English, only the word *bet*).[22] Thus, for these grammars g is 4.

The $|G|$ parameter in the $G^{\setminus\uparrow=\downarrow}$ size formula anticipates that every rule of the LFG grammar participates in the trivial elimination process, but this likely overstates what is necessary to ensure that all zippers can be properly identified. We noted in relation

Generalized Phrase-Structure Grammar (GPSG) where a corresponding (seeming) recognition paradox is also due to the effect of grammar size on recognition performance (see Ristad 1987, for details): Any particular GPSG grammar can be converted into an equivalent context-free grammar, but recognition with the equivalent grammar may be impractical because of its size.) And even with fixed bounds on the height of a functional domain, the rank and the maximum number of trivially-annotated categories in a rule, the number of attributes and values, and the length of the atomic-valued annotations, the k-LCFRS G' in principle can be astronomically larger than the k-bounded LFG G.

22 Kaplan and Wedekind (2019) observe that the schematic prescriptions of X-bar theory allow for arbitrary repetitions of discontinuous branches with complement and cohead nodes that map to the same f-structure and thus give rise to functional domains that appear to exceed any finite height bound and equivalence classes that are not k-bounded. Linguists may not be aware that their succinct X-bar grammars admit strings with unbounded derivations that are not accepted in the language, and that this descriptive inadequacy may be accompanied with undesirable computational properties as we have outlined in this article. Kaplan and Wedekind distill the formal issue down to a simple empirical question about the magnitude of these two extragrammatical parameters, the cohead repetition limit and the degree of discontinuity. This question has not yet been the focus of specific linguistic investigations.

to the recursive expansion of adjuncts (Section 5.4) that trivials that cannot affect the equivalence classes of a derivation do not interfere with the LCFRS construction, and therefore do not have to be removed. The recursive adjunct categories are a special case of a more general class of categories that are inert with respect to zipper interactions. A category is inert in this sense if the zippers within its expansions in all possible derivations do not unify with the zippers outside. It is safe to protect from elimination all trivial annotations attached to inert categories simply by replacing them with $\uparrow \doteq \downarrow$. Kaplan and Wedekind (2019) observe that internal adjuncts of discontinuous NPs are another instance of inertness. Thus even certain function-assigned categories can be inert and their function assignment $(\uparrow \text{F}) = \downarrow$ can therefore be replaced with a variant annotation of the form $(\uparrow \text{F}) \doteq \downarrow$ that explicitly indicates that this function assignment does not conceal zipper interactions. The effect of converting $=$ to $\doteq$ on inert categories is to break the chain of daughter-sequence promotions and to reduce, perhaps substantially, the number of rules of G that actually contribute to the size of $G^{\setminus\uparrow=\downarrow}$.

The second phase of the LCFRS construction is the refinement of LCFRS predicates by the elements of $\mathcal{E}$ that represent the possible combinations of atomic values. Here we see that the potential growth is limited by the properties of linguistically motivated feature systems. The attribute sequences σ in atomic-value annotations can be characterized more precisely as consisting of a sequence γ of zero or more grammatical function attributes followed either by the PRED attribute or a sequence μ of one or more attributes drawn from a set of morphosyntactic features. The morphosyntactic attributes do not appear in reentrancies and so remain stable as reentrancies introduce variation in their γ prefixes. Morphosyntactic attributes may therefore be ignored when determining the length limit of attribute chains in $\mathcal{E}$ components. The parameter ℓ is thus the length of the longest γ subsequence, and according to the Functional Locality Principle its maximum value is 2. The γ attributes can be distinguished from the μ attributes as those that appear only in reentrancies. But in theory and practice, the attributes in μ and the atomic values are statically typed in feature declarations (King et al. 2005) so that the set of values is partitioned across the direct morphosyntactic features (e.g., NUM), and the direct features are partitioned across the grouping features (e.g., AGREEMENT). Typically, the grouping features are merely for conceptual convenience, in which case they do not by themselves give rise to an increase of the description space. It may also be the case that different combinations of morphosyntactic features may be associated with different grammatical functions: The feature NUM may be associated with SUBJ and OBJ but not XCOMP while XCOMP but not the nominal functions may take PASSIVE. Thus for linguistic grammars the atom-valued information space $\mathcal{E}$ is much more sparse than it might be for an unregulated feature system, and the ℓ expansion factor has a tight universal bound.

However, even for linguistically motivated feature systems, the second phase might still result in a huge number of E refinements. This can arise, for example, if lexical items are assigned alternative combinations of agreement features and if reentrancies have the effect of manipulating their γ prefixes so that they propagate unpredictably across the equivalence classes in derivations. But different kinds of reentrancies propagate information in different ways and impose different requirements on the E rule refinements that correctly simulate their operation. Reentrancies of the form $(\downarrow \text{G}) = \uparrow$ are the bottom–up counterpart of top–down function assignments. When they appear in the annotations of a candidate rule sequence, an agreement feature matching G in a daughter E component must be reflected (with G removed) in the mother component. These lifting reentrancies require information to propagate bottom–up, but the degree of expansion is strictly limited by the ℓ bound, just as for function assignments.

Reentrancies in a second subclass introduce more feature variability and hence larger E refinements. Reentrancies of the form $(\uparrow \text{G}) = (\uparrow \text{H})$, for example, result in the replacement of the initial G by H in otherwise required elements of the mother E component. The effect is still limited by the local annotations because it depends on what other agreement features and function assignments are separately required by the candidate rule sequence. It is also limited by the fact that the E components are restricted to the morphosyntactic feature that can associate with both G and H functions.

The most harmful reentrancies, in terms of grammar expansion, are those of the form $(\downarrow \text{G}) = (\uparrow \text{H})$ that specify, for example, the relationship of an XADJUNCT SUBJ to a matrix function or when a long functional control equation (e.g., $(\uparrow \text{XCOMP SUBJ}) = (\uparrow \text{OBJ})$) is reduced to a short reentrancy $(\downarrow \text{SUBJ}) = (\uparrow \text{OBJ})$. These propagate information through candidate rule sequences without regard to other local properties and thus must give rise to E components that are locally unconstrained. Of course, as just noted, only morphosyntactic features common to both G and H must be considered, and this removes many features from consideration in linguistic grammars where control typically ranges over nominal functions with common feature sets. This is not an accidental property of control equations in linguistically motivated grammars; it was claimed as a universal principle of language in the earliest formulations of LFG theory (Bresnan 1982). The Lexical Rule of Functional Control stipulates that lexical control equations can only be of the form $(\uparrow \text{XCOMP SUBJ}) = (\uparrow \text{GF})$, where GF is one of SUBJ, OBJ, or OBJ2, and the Constructional Rule of Functional Control provides for phrasal annotations that pair the function assignment $(\uparrow \text{XADJUNCT}) = \downarrow$ with the short control reentrancy $(\downarrow \text{SUBJ}) = (\uparrow \text{GF})$. These principles may gain their explanatory and descriptive power because natural languages are organized to minimize the computational impact of these most promiscuous reentrancies.

We then obtain a bound on the size of G' for a linguistically motivated G as follows. Because the rank of the LCFRS G' is bounded by $g + c$ and the LCFRS predicates for the grammatical functions can at most be k-ary and the predicates for the coheads are unary, the size bound on $G^{\setminus\uparrow=\downarrow}$ already accounts for rule sequences of length up to k and therefore the number of $\mathcal{E}$-unrefined skeleton LCFRS rules can also not exceed $|G|^{kg+c+1}$. With a denoting the maximum number of attested agreement/PRED feature combinations, the size of G' is bounded by $a^{g+c+1}|G|^{kg+c+1}$.

The broad-coverage grammars written by practicing linguists and grammar engineers diverge from our basic formalism in two ways: They come equipped with separate and quite extensive lexicons and they use single rules with regular right sides to succinctly represent the many alternative daughter sequences of particular categories. Obviously, LCFRS equivalents can be constructed for grammars in this format by applying a preprocessing step that converts them to the terminals and rules of the basic formalism, but this initial expansion step may result in starting grammars for LCFRS analysis that are already of impractical size.

It is not difficult to preserve the modularity of a separate lexicon and to insulate the LCFRS construction from the sheer number of lexical entries. An entry associates a word to a set of senses, each of which is a prelexical category/annotation pair. The naive approach would create a new terminal and prelexical rule for each such word-sense combination, but this does not take advantage of the fact that sense specifications are shared by large subsets of the vocabulary. Almost all intransitive verbs, for example, are marked with the same category and with annotations that differ only in the particular relation (WALK vs. DIE) embedded in their PRED semantic forms. A simple technique is to remove the specific relations from semantic forms, group the senses into classes according to the categories and relation-abstracted annotations, let those abstracted

senses enter as terminals into the LCFRS construction, and keep for later use a separate lexical table that maintains the association between semantic relations and abstracted senses. LCFRS size is then a function of the relatively small number of abstract-sense classes, not the overall size of the vocabulary.

It remains to address the second growth factor in the LCFRS construction for broad-coverage LFG grammars, the conversion from the regular right side format of typical LFG rules (as compiled into finite-state machines) and the linearized representation of the basic rule format that we have exploited in our theoretical approach to LCFRS construction. As an example, the finite-state encoding of the English progressive VP rule before trivial elimination has 15 states and 31 transitions, defining 324 linear paths with an average length of 11. Trivial elimination can be carried out on the finite-state representation, and this produces a larger but still quite manageable finite-state machine with 52 states and 96 transitions. But the number of paths encoded in that machine is substantially greater, 110,000, and that would be the size of the equivalent set of linearized, basic rules that would enter into the equivalence-class construction. Thus, linearization of regular expression right sides is theoretically possible, as described in Section 5.4, but the resulting expansion in the number of basic rules, essentially a conversion to disjunctive normal form, may then become the major challenge to practicality.

A potential solution is to perform some or all of the equivalence class calculations directly on the transitions of the finite-state machine before enumerating all of its paths. It is easy to extract all the transition labels (categories and annotations) that assign a particular grammatical function and to test them for mutual satisfiability. Finite-state operations can be applied to prune the machine of paths that contain inconsistent combinations, and the remaining transitions can be reorganized to bring together the labels of consistent combinations. This changes the sequential order of the categories, but in exactly the way this is done in the LCFRS rule construction. The difference is that a single adjustment might produce a class representation that is common to a large number of basic rules. The machine that emerges from this process would be a compact representation, in finite-state form, of all and only the LCFRS rules that meet the conditions of Definition 23; no unsatisfactory paths would remain in the machine.

In the worst case, where equivalent categories are randomly distributed throughout the machine, the machine will incrementally grow to approximate a disjunctive set of independent rule sequences. But experience with XLE and the Pargram grammars suggests that equivalent categories for natural language grammars are distributed more systematically. The algorithms in XLE that are key to its high-speed performance are optimized for the situation where categorial annotations are relatively independent of the annotations on distant categories, that annotations interact in a mostly context-free-like way (Maxwell and Kaplan 1996). The incremental enumeration of equivalence classes in a finite-state setting should also benefit implicitly from this property. XLE operates directly on the finite-state machine representation of the grammar, interpreting on the fly the alternative transitions leaving a state. If an LCFRS parsing algorithm can be augmented to also operate in this way, it may never be necessary to read out the full set of paths.

Because of the complexity of the regular expressions and the massive use of disjunctions, $|G'|$ can still be impractically large. Thus, for realistic grammars it may be worth applying an alternative strategy to parsing that avoids constructing the LCFRS for all rules and features of G and instead builds an LCFRS at parse-time only for a given input. This strategy relies on the fact that for many grammar formalisms, such as context-free grammars, the result of the intersection of a language $L(G)$ with a singleton

string-set $\{s\}$ is describable as a specialization G_s of G that assigns to s effectively the same parses as G would assign (Bar-Hillel et al. 1961; Lang 1994). The parsing problem is divided into four steps. In the first step, the LFG G is specialized to an LFG whose context-free backbone language consists only of the given input string s (if s belongs to the backbone language of G), as Lang and others have pointed out. This can be done in cubic time by any number of context-free parsing algorithms, modified simply to record the annotations associated with the nonterminal categories. The size of the resulting grammar G_s is proportional to $|s|^3$, and G_s is k_s-bounded if G is k-bounded, for $k_s \leq k$. The other parameters affecting LCFRS size and parsing complexity, g_s, c_s, and a_s), are also typically much smaller than those for G (for example, g_s is less than 4 for English sentences that do not contain *bet*). In the second step, G_s is translated into the equivalent but correspondingly small LCFRS G'_s just as described earlier. In the third step (recognition) an LCFRS recognition algorithm determines whether there is at least one parse that satisfies the restrictions defined by the original annotations. The effect is that $s \in L(G)$ iff $s \in L(G'_s)$. In the last step (enumeration) f-structures from alternative parses can be produced one by one. This strategy has the practical advantage that the LCFRS translation does not involve rules or features (including predicates and their subcategorization frames) that are not relevant to the given input. Notice that the rules for the specialized grammar are no longer in regular-expression/finite-state machine format, and sophisticated algorithms that operate on finite-state machine encodings of LCFRS rules will provide little computational benefit.

As mentioned, parsing and generation systems—for example, the XLE system—have been developed that are practical for broad-coverage grammars and naturally occurring sentences (Crouch et al. 2008). XLE is based on the lazy contexted constraint satisfaction method developed by Maxwell and Kaplan (1991, 1996) that is optimized for context-free structures in which disjunctions arising from words and phrases that are distant from each other in the string do not interact. This is because XLE multiplies disjunctions at context-free constituents only if needed. Moreover, XLE does not require the regular right sides to be linearized. These features of the XLE parsing algorithm, which have been proven extremely effective, suggest a third, presumably more efficient parsing strategy.

If the core XLE strategy is modified so that it takes account of the zippers (of the reentrancy-free kernel) before the contexted constraint satisfaction algorithm is applied for testing clash-freeness, the algorithm would also take advantage of the multiple context-freeness (of natural languages). Then, for k-bounded LFGs the modified algorithm would also perform well for discontinuous constructions, and only the relatively rare multiply embedded control constructions could force the algorithm to expand into DNF the specifications of the broader scopes associated with the controller and controllees.

7. Concluding Remarks

The k-bounded LFG grammars form a decidable proper subclass of the LFG formalism, restricted both in notation and in the properties of their derivations. We have suggested that the notation for annotating individual c-structure categories is still linguistically suitable in that it allows for function assignments, trivial annotations to identify heads and coheads, long (but respecting Functional Locality) reentrancies for control, height-bounded functional domains, and value specifications for feature instantiation and agreement. Further, the derivational conditions of Definitions 13 and 14 provide a technical characterization of the informal linguistic notion that reentrancies in general

are not constructive. Whereas off-line parsability ensures only that the parsing problem is decidable, the restrictions developed in this article are sufficient for the decidability of the emptiness and generation problems as well. We demonstrated that these requirements make possible the translation into equivalent LCFRSs, and we explored the effectiveness of some alternative strategies for LFG parsing.

We have focused in this article on the formal devices proposed by Kaplan and Bresnan (1982) and still in common use. Modern LFG, however, includes a number of more sophisticated mechanisms that were later on woven into the theory. Among these are devices for the f-structure characterization of long distance dependencies and coordination: functional uncertainty (Kaplan and Zaenen 1989; Kaplan and Maxwell 1988a), set distribution for coordination, and the interaction of uncertainty and set distribution (Kaplan and Maxwell 1988b); and devices whose evaluation depends on properties of the c-structure to f-structure correspondence, namely, functional categories and extended heads (Zaenen and Kaplan 1995; Kaplan and Maxwell 1996) and functional precedence (Bresnan 1995; Zaenen and Kaplan 1995).

We know that some of these devices are needed for the analysis of phenomena that can only be modeled by tractable extensions of LCFRS but not LCFRS per se (Becker et al. 1992; Boullier 1999; Kallmeyer 2010a). Boullier (1999) has demonstrated that range concatenation grammars (RCGs) can handle long-distance scrambling (in German) and Kallmeyer (2010a) that gapping in coordinated structures can be modeled by literal movement grammars of constant non-linearity. We thus conjecture that the LFG devices used for describing these phenomena can—at least to the extent needed in order to provide linguistically motivated analyses—be captured in these extensions of LCFRS. Some evidence for this conjecture is provided by Rambow (1997), who sketches how to model functional uncertainty in unordered vector grammars with dominance links, a TAG related (tractable) framework. Peled and Wintner (2015) describe a unification-based formalisms that is equivalent to RCG. Although this formalism differs considerably from the more elaborate versions of the LFG formalism that include a number of separate but interacting mechanisms, their restrictions might still help in modeling these LFG mechanisms in tractable extensions of LCFRS.

The LFG formalism has proven to be rich enough in its descriptive power to enable concise characterizations of complex syntactic dependencies in languages of many different types. The formalism is also rich enough so that many computational problems are undecidable and others are intractable. The question we have addressed here is whether there are limitations in the notations or its interpretation that eliminate the computational excesses of the formalism while preserving its descriptive utility, and perhaps even provide a deeper computational explanation for some principles of linguistic organization. Taking the LCFRS formalism and the class of multiple context-free languages as the touchstone of our investigation has enabled us to specify a small number of conditions that are sufficient for tractability and may still be linguistically suitable. One consequence is that the methods and formal results pertaining to multiple context-free languages may lead directly to a better understanding of other aspects of LFG and perhaps other unification-based formalisms. This may also lead to incremental improvements in existing LFG systems, such as XLE. At a higher level, the interplay between the descriptive succinctness of the LFG formalism and its mildly context-sensitive equivalents may help to define a tighter formal envelope around the class of natural languages.

Appendix A. The Elimination of Trivial Annotations

Given the height bound of Definition 10, we can remove all trivial annotations. For the elimination we assume that any occurrence of $\downarrow$ in the annotated reentrancies and atomic-valued annotations of a nonterminal carrying $\uparrow = \downarrow$ is replaced by $\uparrow$. This replacement is legitimate because it is equivalence-preserving and the resulting annotations are admissible.

Lemma 1
For any suitable LFG G we can construct a suitable LFG without $\uparrow = \downarrow$ annotations, denoted by $G^{\setminus\uparrow=\downarrow}$, such that $\Delta_G = \Delta_{G^{\setminus\uparrow=\downarrow}}$.

Proof
Let $G = (N, T, S, R)$ be a suitable LFG grammar. Without loss of generality assume that the reentrancies and atomic-valued annotations of nonterminals annotated with $\uparrow = \downarrow$ do not contain $\downarrow$. Now let $\overline{R}$ be the smallest set that includes R and that is closed under the following rule: if $A \to \phi(B, D)\psi \in \overline{R}$, with $\uparrow = \downarrow \in D$, and $B \to (X_1, D_1)\omega \in \overline{R}$, then $A \to \phi(X_1, D_1 \cup (D \setminus \{\uparrow = \downarrow\}))\omega\psi \in \overline{R}$. Define R' as the subset of $\overline{R}$ that only includes rules $A \to \phi$ where for any (B, D) in ϕ with $B \in N$, $\uparrow = \downarrow \notin D$. By the condition of Definition 10, R' and $\overline{R}$ are finite. Set $G^{\setminus\uparrow=\downarrow} = (N, T, S, R')$. Obviously, $G^{\setminus\uparrow=\downarrow}$ is suitable. Then $\Delta_G \subseteq \Delta_{G^{\setminus\uparrow=\downarrow}}$ by a simple bottom–up induction on the derivations in G and $\Delta_{G^{\setminus\uparrow=\downarrow}} \subseteq \Delta_G$ because for any rule $r = A \to (X_1, D_1)..(X_m, D_m)$ in R' there is by construction of R' a corresponding derivation of $X_1..X_m$ from A in G that yields the same f-description as r if every instantiated trivial annotation $n = n_j$ is removed and the instantiating daughter node n_j is eliminated by substitution $[n_j/n]$. ■

Appendix B. Suitable LFGs with (Finitely) Bounded Reentrancy-Free Kernel

A suitable LFG G has a **bounded reentrancy-free kernel** if there is a k such that for every derivation (c, ρ) of a terminal string from S with f-description FD and clash-free $FD_{\setminus\mathcal{R}}$ in $G^{\setminus\uparrow=\downarrow}$, $|\{n' \mid FD_{\setminus\mathcal{R}} \vdash n = n'\}| \leq k$, for any node n of c. The decidability proof for this boundedness problem is (similar to that of the k-boundedness problem presented in Section 4.1) based on an enumeration of all derivations of $G^{\setminus\uparrow=\downarrow}$ up to a certain depth. Here, however, we attempt to find a derivation that can be "pumped", that is, one that can be used to show that the equivalence classes are not finitely bounded, and not just a derivation with an equivalence class greater than some given k. Before we present the proof we introduce the pumping operation.

Let (c, ρ) be a derivation and $[\hat{n}]$ and $[n^l]$ be two node classes where at least one node in $[\hat{n}]$ dominates a node in $[n^l]$. Suppose that h assigns to $[\hat{n}]$ and $[n^l]$ the same set of atomic-valued schemata and there is a mapping $\bar{g}$ from $[n^l]$ to $[\hat{n}]$ such that for all n^i in $[n^l]$, $\bar{g}(n^i)$ and n^i are licensed by the same rule. Now let δ be a function that maps the nodes in $[n^l]$ to their dominating nodes in $[\hat{n}]$. Then we can pump the derivation if there is a nonempty subset V of $[n^l]$, the image of V under $\bar{g}$ ($\bar{g}(V)$) is smaller than V, and the nodes in V are dominated by the nodes in the image of V under $\bar{g}$ ($\bar{g}(V) = \delta(V)$). This situation is illustrated in the left-hand derivation tree of Figure 11. For defining the pumped derivation, we make use of the **Repl** operator specified in Section 4.1. Figure 11 illustrates on the right the result of the pumping process for the derivation tree on the left. The pumping operation (*pu*) is formally defined as follows.

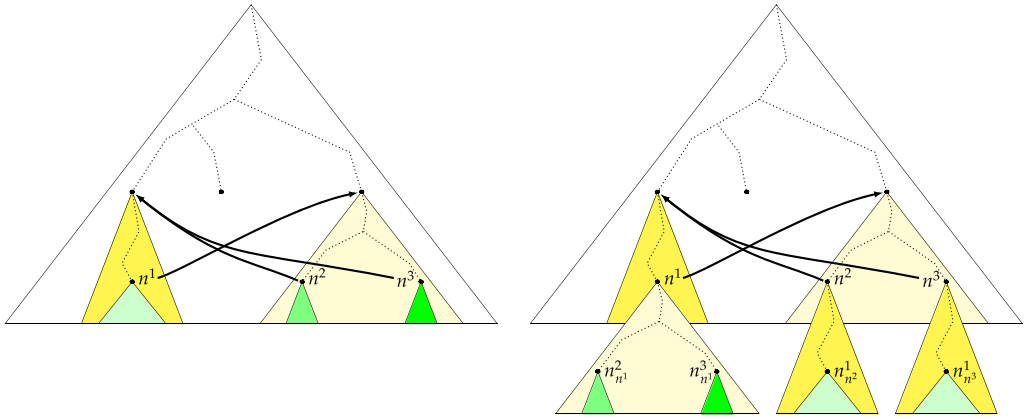


Figure 11
Illustration of the pumping process. In the derivation tree on the left-hand side, the node set $\{n^1, n^2, n^3\}$ corresponds to the class $[n^l]$ and the set comprising the three upper nodes corresponds to $[\hat{n}]$. The dotted lines indicate dominance and the solid arrows the $\bar{g}$ mapping. Suppose that $h([n^l]) = h([\hat{n}])$ and $\rho_{\bar{g}(n^i)} = \rho_{n^i}$, for $n^i \in [n^l]$. Obviously, for $V = [n^l]$, $|\bar{g}(V)| < |V|$ and $\bar{g}(V) = \delta(V)$. Then pumping is performed by expanding n^1, n^2, and n^3 with distinct copies of the subderivations dominated by $\bar{g}(n^1)$, $\bar{g}(n^2)$, and $\bar{g}(n^3)$, respectively. The result is shown on the right-hand side (with the $\bar{g}$ mapping still indicated).

Definition 26
Let (c, ρ) be a derivation in a suitable LFG grammar $G^{\uparrow=\downarrow}$. For any pair of classes $[\hat{n}]$ and $[n^l]$ satisfying the conditions

(i) $h([\hat{n}]) = h([n^l])$,

(ii) there is a function $\bar{g}$ from $[n^l]$ to $[\hat{n}]$ such that

(a) $\rho_{g(n^i)} = \rho_{n^i}$, for each $n^i \in [n^l]$, and

(b) there is a nonempty subset V of $[n^l]$ such that $|\bar{g}(V)| < |V|$ and $\bar{g}(V) = \delta(V)$,[23]

we define the pump operation pu by

$$pu_{(c,\,\rho)}([\hat{n}], [n^l]) = \mathbf{Repl}^{\bar{g}}_{(c,\,\rho)}([n^l]).$$

For any given derivation (c, ρ) with two equivalence classes $[n^l]$ and $[\hat{n}]$ satisfying the conditions of Definition 26, $pu_{(c,\rho)}([\hat{n}], [n^l])$ is also a derivation of $G^{\uparrow=\downarrow}$ that by construction still gives rise to the equivalence classes $[n^l]$ and $[\hat{n}]$. Now let $\bar{g}$ be the mapping from $[n^l]$ to $[\hat{n}]$ as defined for the original derivation (shown in the right derivation of Figure 11). Because the original derivation satisfied $\rho_{\bar{g}(n^i)} = \rho_{n^i}$, for each $n^i \in [n^l]$, and we expanded every n^i by the subderivation rooted at $\bar{g}(n^i)$, the new derivation must also satisfy $\rho_{\bar{g}(n^i)} = \rho_{n^i}$. Hence also $h([n^l]) = h([\hat{n}])$ and, as a consequence, the f-description that G's reentrancy-free kernel provides for this derivation is clash-free. (Note that this would not be guaranteed if $\bar{g}$ were not total.) Thus, the classes $[n^l]$ and

23 Note that the specification of V implies that at least one $\bar{g}$ value branches into two or more nodes of V.

$[\hat{n}]$ of $pu_{(c,\rho)}([\hat{n}],[n^l])$ satisfy the conditions of Definition 26 and the pumping operation can be applied again.

Because the renamed n^i from V of the pushed down copies of the $\bar{g}$ value derivations must be equivalent ($n^2_{n^1}$, $n^3_{n^1}$, $n^1_{n^2}$, and $n^1_{n^3}$ in the example of Figure 11) and $\bar{g}$ is required to satisfy $|\bar{g}(V)| < |V|$, the number of these equivalent copies must be greater than $|V|$. Thus, by further pumping, the number of equivalent nodes that trace back to V through renaming must stepwise grow. Under these conditions the reentrancy-free kernel of G is not finitely bounded.

The decidability for finite boundedness results from a pumping lemma whose proof is reminiscent of the proof of the pumping lemma for context-free languages in that it relies on the branching of c-structure nodes in derivations that are sufficiently deep. We ensure with provisions specific to our setting that c-structure branching also gives rise to growth in the size of equivalence classes, as in Definition 26. The pumping lemma states that a derivation can be pumped if there is a node equivalence class whose size exceeds some fixed finite bound (the pumping size for the reentrancy-free kernel of $G^{\setminus\uparrow=\downarrow}$). This bound depends on the maximum number of equivalent daughters that any rule of $G^{\setminus\uparrow=\downarrow}$ can derive. We will refer to this maximum as the **maximum functional branching factor** b of $G^{\setminus\uparrow=\downarrow}$. Formally, b is defined as follows. We first define for each rule $r = A \rightarrow \psi$ in $G^{\setminus\uparrow=\downarrow}$ its maximum functional branching factor, denoted by b_r. Let a be a constant and β be a sequence of pairwise distinct constants of length $|\psi|$, each of them distinct from a, then

$$\mathsf{b}_r = \max\{|[\beta_i]| \mid i = 1,..,|\psi| \text{ and } [\beta_i] = \{\beta_j \mid \mathit{Inst}(r,(a,\beta))_{\setminus\mathcal{R}} \vdash \beta_i = \beta_j\}\}.$$

Then the maximum functional branching factor of the grammar is the (non-zero) maximum of the branching factors b_r of all rules r of $G^{\setminus\uparrow=\downarrow}$: $\mathsf{b} = \max(\{\mathsf{b}_r \mid r \in R\} \cup \{1\})$. In the proof of the following pumping lemma, the image of a subset of nonterminal nodes $\boldsymbol{n}$ of c under ρ is denoted by $\rho(\boldsymbol{n})$.

Lemma 9

The reentrancy-free kernel of a suitable LFG G is unbounded if and only if there is a derivation (c,ρ) with clash-free $FD_{\setminus\mathcal{R}}$ in $G^{\setminus\uparrow=\downarrow}$ and $|[n]| > \mathsf{b}^{\mathsf{d}}$, with $\mathsf{d} = \left(\sum_{P \in Pow^+(R)} |Pow^+(P)|\right) \cdot |\mathcal{E}|$, for at least one node n of c.

Proof

Let (c,ρ) be a derivation in $G^{\setminus\uparrow=\downarrow}$ with clash-free $FD_{\setminus\mathcal{R}}$ and suppose that $|[n]| > \mathsf{b}^{\mathsf{d}}$ for some node n of c. Let $root = \bar{n}^0..\bar{n}^s = n$ be the path to n with $\bar{n}^i$ immediately dominating $\bar{n}^{i+1}$. For each $i = 0,..,s-1$, let $\bar{\boldsymbol{n}}^i$ be the nodes of $[\bar{n}^i]$ that dominate the nodes in $[\bar{n}^s]$. For s, set $\bar{\boldsymbol{n}}^s = [\bar{n}^s]$. Now let f be a function that assigns to each $[\bar{n}^j]$, $j = 0,..,s$, the triple $(\rho([\bar{n}^j]), \rho(\bar{\boldsymbol{n}}^j), h([\bar{n}^j]))$ consisting of the rules that license the nodes in $[\bar{n}^j]$, the subset of those rules that license the nodes of $[\bar{n}^j]$ that dominate the nodes in $[\bar{n}^s]$, and the description assigned to $[\bar{n}^j]$. Since $|[\bar{n}^s]| > \mathsf{b}^{\mathsf{d}}$, there must be nodes $\bar{n}^{i_1},..,\bar{n}^{i_q} = \bar{n}^s$ on the path $\bar{n}^0..\bar{n}^s$ with $q > \mathsf{d}$ and $|\bar{\boldsymbol{n}}^{i_j}| < |\bar{\boldsymbol{n}}^{i_{j+1}}|$, for $j = 1,..,q-1$. Because, for each $f([\bar{n}^{i_j}]) = (P,Q,E)$, $j = 1,..,q$, P is a nonempty subset of R, Q is a nonempty subset of P, and E is an element of $\mathcal{E}$ the number of distinct f values is bounded by d. Thus, if $q > \mathsf{d}$, there must be two classes $[\bar{n}^{i_k}]$, $[\bar{n}^{i_l}]$ that (with $V = \bar{\boldsymbol{n}}^{i_l}$) satisfy the conditions of Definition 26 ■

Given the proofs of the pumping lemma and the decidability of the k-boundedness (Lemma 3), it is now easy to see that finite boundedness can be determined by inspecting all derivations of (c-structure) depth less than or equal to $2\mathsf{d} + |Pow^{+}(R) \times \mathcal{E}|$. We show here by contradiction that it can be decided whether there is a derivation in $G^{\uparrow=\downarrow}$ that satisfies the conditions of Definition 26 by inspecting $G^{\uparrow=\downarrow}$'s derivations only up to depth $2\mathsf{d} - 1$. The rest of the proof (that justifies $|Pow^{+}(R) \times \mathcal{E}| + 1$) is similar to the proof of Lemma 3. Thus suppose there were a derivation (c, ρ) in $G^{\uparrow=\downarrow}$ with clash-free $FD_{\backslash_{\mathcal{R}}}$ with a path $root = \bar{n}^0..\bar{n}^s$ with $s \geq 2\mathsf{d}$ and the pair $[\bar{n}^l]$, $[\bar{n}^s]$, $l < s$, but no higher pair would satisfy the conditions of Definition 26. Because $s \geq 2\mathsf{d}$, there must be a pair of classes $[\bar{n}^i]$, $[\bar{n}^j]$, $i < j < s$ such that $f([\bar{n}^i]) = f([\bar{n}^j])$ and either $l \leq i$ or $l \geq j$. If this pair does not allow pumping then $|\bar{\boldsymbol{n}}^i| = |\bar{\boldsymbol{n}}^j|$ and there must be a g from $[\bar{n}^i]$ to $[\bar{n}^j]$ that includes $(\bar{g}|_{\bar{\boldsymbol{n}}^j})^{-1}$ and allows it to shrink the derivation. Shrinking then lifts the pair of classes satisfying the conditions of Definition 26, which leads to a contradiction with our assumption.

If the reentrancy-free kernel of $G^{\uparrow=\downarrow}$ is finitely bounded then the k bound is the maximum size of the equivalence classes of all these derivations.

Lemma 10
For any suitable LFG G it is decidable whether there is a k such that for every derivation (c, ρ) *of a terminal string from* S *with f-description FD and clash-free* $FD_{\backslash_{\mathcal{R}}}$ *in* $G^{\uparrow=\downarrow}$, $|\{n' \mid FD_{\backslash_{\mathcal{R}}} \vdash n = n'\}| \leq k$, *for any node n of c.*

Acknowledgments
The authors would like to thank the three anonymous reviewers for their helpful suggestions and comments on earlier drafts of this article.

References

Bar-Hillel, Yehoshua, Micha Perlis, and Eliahu Shamir. 1961. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14, 143–172.

Becker, Tilman, Owen Rambow, Michael Niv. 1992. The derivational power of formal systems, or scrambling is beyond LCFRS. Technical report IRCS 92-38, Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia, PA.

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

Blackburn, Patrick and Edith Spaan. 1993. Decidability and undecidability in stand-alone feature logics. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics*, pages 30–36, Utrecht.

Boullier, Pierre. 1999. Chinese numbers, mix, scrambling, and range concatenation grammars. *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics*, pages 53–60, Bergen.

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

Bresnan, Joan. 1982. Control and complementation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA, pages 282–390.

Bresnan, Joan. 1995. Linear order, syntactic rank, and empty categories: On weak crossover. In M. Dalrymple, R. M. Kaplan, J. T. Maxwell III, and A. Zaenen, editors, *Formal Issues in Lexical-Functional Grammar*. CSLI Publications, Stanford, CA, pages 241–274.

Bresnan, Joan. 2001. *Lexical-Functional Syntax*. Blackwell Publishers, Oxford.

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

Crouch, Richard, Mary Dalrymple, Ronald M. Kaplan, Tracy Holloway King, John T. Maxwell III, and Paula Newman. 2008. XLE documentation. Technical report, Palo Alto Research Center, Palo Alto, CA. Available at `http://www2.parc.com/isl/groups/nltt/xle/doc/xle_toc.html`.

Dalrymple, Mary. 2001. *Lexical Functional Grammar*. Academic Press, New York, NY.

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

Dalrymple, Mary, Ronald M. Kaplan, John T. Maxwell III, and Annie Zaenen. 1995b. Non-local dependencies. In M. Dalrymple, R. M. Kaplan, J. T. Maxwell III, and A. Zaenen, editors, *Formal Issues in Lexical-Functional Grammar*. CSLI Publications, Stanford, CA, pages 131–135.

Dalrymple, Mary, Ronald M. Kaplan, and Tracy Holloway King. 2015. Economy of expression as a principle of syntax. *Journal of Language Modelling*, 3(2):377–412.

Feinstein, Daniel and Shuly Wintner. 2008. Highly constrained unification grammars. *Journal of Logic, Language, and Information*, 17(3):345–381.

Jaeger, Efrat, Nissim Francez, and Shuly Wintner. 2005. Unification grammars and off-line parsability. *Journal of Logic, Language, and Information*, 14(2):199–234.

Johnson, Mark. 1988. *Attribute–Value Logic and the Theory of Grammar*. CSLI Publications, Stanford, CA.

Joshi, Aravind K. 1985. Tree Adjoining Grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In D. R. Dowty, L. Karttunen, and A. M. Zwicky, editors, *Natural Language Parsing: Theoretical, Computational, and Psychological Perspectives*. Cambridge University Press, New York, NY, pages 206–250.

Kallmeyer, Laura. 2010a. On mildly context-sensitive non-linear rewriting. *Research on Language and Computation*, 8(4):341–363.

Kallmeyer, Laura. 2010b. *Parsing Beyond Context-Free Grammars*. Springer, Berlin, Heidelberg.

Kallmeyer, Laura. 2013. Linear context-free rewriting systems. *Language and Linguistics Compass*, 7(1):22–38.

Kaplan, Ronald M. 1995. The formal architecture of Lexical-Functional Grammar. In M. Dalrymple, R. M. Kaplan, J. T. Maxwell III, and A. Zaenen, editors, *Formal Issues in Lexical-Functional Grammar*. Stanford, CA, pages 7–27.

Kaplan, Ronald M., and J. Bresnan. 1982. Lexical-Functional Grammar: A formal system for grammatical representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA, pages 173–281.

Kaplan, Ronald M. and John T. Maxwell III. 1988a. An algorithm for functional uncertainty. In *Proceedings of the 12th International Conference on Computational Linguistics*, pages 297–302, Budapest.

Kaplan, Ronald M. and John T. Maxwell III. 1988b. Constituent coordination in Lexical-Functional Grammar. In *Proceedings of the 12th International Conference on Computational Linguistics*, pages 303–305, Budapest.

Kaplan, Ronald M. and John T. Maxwell III. 1996. LFG grammar writer's workbench. Technical report, Xerox Palo Alto Research Center, Palo Alto, CA. Available at `ftp://ftp.parc.xerox.com/pub/lfg/lfgmanual.ps`.

Kaplan, Ronald M. and Jürgen Wedekind. 2019. Tractability and discontinuity. In *Proceedings of the International Lexical-Functional Grammar Conference 2019*, pages 130–148, CSLI Publications, Stanford, CA.

Kaplan, Ronald M. and Annie Zaenen. 1989. Long-distance dependencies, constituent structure, and functional uncertainty. In M. Baltin and A. Kroch, editors, *Alternative Conceptions of Phrase Structure*. Chicago University Press, Chicago, IL, pages 17–42.

King, Tracy Holloway, Martin Forst, Jonas Kuhn, and Miriam Butt. 2005. The feature space in parallel grammar writing. *Research on Language and Computation*, 3(2–3):139–163.

Lang, Bernard. 1994. Recognition can be harder than parsing. *Computational Intelligence*, 10(4):486–494.

Maxwell III, John T. and Ronald M. Kaplan. 1991. A method for disjunctive constraint satisfaction. In M. Tomita, editor, *Current Issues in Parsing Technology*. Kluwer, Dordrecht, pages 173–190.

Maxwell III, John T. and Ronald M. Kaplan. 1996. Unification-based parsers that automatically take advantage of context-freeness. In *Proceedings of the International Lexical-Functional Grammar Conference 1996*, CSLI Publications, Stanford, CA.

Nelson, Greg and Derek C. Oppen. 1980. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364.

Nishino, Tetsuro. 1991. Mathematical analysis of Lexical-Functional Grammars. *Language Research*, 27(1):119–141.

Peled, Hadas and Shuly Wintner. 2015. Polynomially parsable unification grammars. *Journal of Logic and Computation*, 25(5):1167–1202.

Rambow, Owen. 1997. A polynomial model for unrestricted functional uncertainty. In *Proceedings of the 5th Meeting on Mathematics of Language*, pages 138–145, Schloß Dagstuhl.

Ristad, Eric S. 1987. GPSG-recognition is NP-hard. *Linguistic Inquiry*, 18(3):530–536.

Roach, Kelly. 1983. LFG languages over a one-letter alphabet. Manuscript, Xerox PARC, Palo Alto, CA.

Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

Seki, Hiroyuki, Ryuichi Nakanishi, Yuichi Kaji, Sachiko Ando, and Tadao Kasami. 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, Columbus, OH.

Statman, Richard. 1977. Herbrand's theorem and Gentzen's notion of a direct proof. In Jon Barwise, editor, *Handbook of Mathematical Logic*. North-Holland, Amsterdam, pages 897–912.

Vijay-Shanker, K., David J. Weir, and Aravind K. 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, CA.

Wedekind, Jürgen. 1999. Semantic-driven generation with LFG- and PATR-style grammars. *Computational Linguistics*, 25(2):277–281.

Wedekind, Jürgen. 2014. On the universal generation problem for unification grammars. *Computational Linguistics*, 40(3):533–538.

Wedekind, Jürgen and Ronald M. Kaplan. 2012. LFG generation by grammar specialization. *Computational Linguistics*, 38(4):867–915.

Weir, David J. 1988. Characterizing mildly context-sensitive grammar formalisms. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Zaenen, Annie and Ronald M. Kaplan. 1995. Formal devices for linguistic generalizations: West Germanic word order in LFG. In M. Dalrymple, R. M. Kaplan, J. T. Maxwell III, and A. Zaenen, editors, *Formal Issues in Lexical-Functional Grammar*. CSLI Publications, Stanford, CA, pages 215–239.