# LFG Generation Produces Context-free Languages

**Ronald M. Kaplan**
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304 USA
kaplan@parc.xerox.com

**Jürgen Wedekind**
Center for Language Technology
Njalsgade 80
2300 Copenhagen S, Denmark
juergen@cst.ku.dk

## Abstract

This paper examines the generation problem for a certain linguistically relevant subclass of LFG grammars. Our main result is that the set of strings that such a grammar relates to a particular f-structure is a context-free language. This result obviously extends to other context-free based grammatical formalisms, such as PATR, and also to formalisms that permit a context-free skeleton to be extracted (perhaps some variants of HPSG). The proof is constructive: from the given f-structure a particular context-free grammar is created whose yield is the desired set of strings. Many existing generation strategies (top-down, bottom-up, head-driven) can be understood as alternative ways of avoiding the creation of useless context-free productions. Our result can be established for the more general class of LFG grammars, but that is beyond the scope of the present paper.

## 1 Introduction and Preliminaries

This paper examines the generation problem for a certain linguistically motivated subclass of LFG grammars. Our main result is that the set of strings that such a grammar relates to a particular f-structure is a context-free language. This result extends easily to other context-free based grammatical formalisms, such as PATR (Shieber et al. 1983), and perhaps also to formalisms that permit a context-free skeleton to be extracted from richer representations.

We begin with some background and formal definitions so that we can make the problem and its solution explicit. An LFG grammar $G$ assigns to every string in its language at least one c-structure/f-structure pair that are set in correspondence by a piecewise function $\phi$ (Kaplan 1995). The situation can be characterized in terms of a derivation relation $\Delta_G$, defined as follows:

(1) $\Delta_G(s, c, \phi, f)$ iff $G$ assigns to the string $s$ a c-structure $c$ that piecewise-corresponds to f-structure $f$ via the function $\phi$.

The 'piecewise-corresponds' notion means that $\phi$ maps individual nodes of a c-structure tree to units of the f-structure. The arrangement of the four components of an LFG representation is illustrated in the diagram of Figure 1. This representation belongs to the $\Delta_G$ relation for a grammar that includes the annotated (nonterminal) rules in (2) and lexical rules in (3).

(2) a. S → NP (↑ SUBJ) =↓ (↓ CASE) = NOM; VP ↑=↓ (↑ TENSE)

b. NP → DET ↑=↓; N ↑=↓

c. VP → V ↑=↓

(3) a. DET → a (↑ SPEC) = INDEF (↑ NUM) = SG

b. N → student (↑ PRED) = 'STUDENT' (↑ NUM) = SG (↑ SPEC)

c. V → fell (↑ PRED) = 'FALL⟨(SUBJ)⟩' (↑ TENSE) = PAST

The c-structure in Figure 1 is derived by applying a sequence of rules from (2) to rewrite the symbol S, the grammar's start symbol, and then rewriting the preterminal categories according to the lexical rules. Lexical rules are just notational variants of traditional LFG lexical entries.

The $\phi$ correspondence and the f-structure in Figure 1 are associated with that c-structure because the f-structure satisfies the $\phi$-instantiated description constructed from the annotated c-structure derivation, and furthermore, it is a minimal model for the set of instantiated descriptions collected from all the nodes of the annotated c-structure. The $\phi$-instantiated description for a local mother-daughters configuration justified by a rule is created in the following way. First, all occurrences of the symbol ↑ in the functional annotations of the daughters are replaced by a variable standing for the f-structure unit that $\phi$ assigns to the mother node. Then for each of the daughter categories, all occurrences of the symbol ↓ in its annotations are replaced by a variable

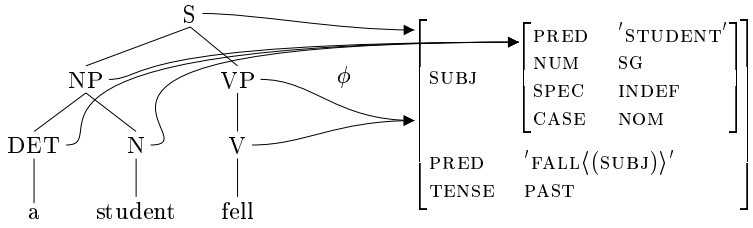


Figure 1: Piecewise c- and f-structure correspondence.

standing for the $\phi$ assignment of the daughter node. Observe that all variables denote f-structure units in the range of $\phi$, and that the $\downarrow$ on a category and the $\uparrow$ on the daughters that further expand that category are always instantiated with the same variable.

We now turn to the generation problem. A generator for $G$ provides for any given f-structure $F$ the set of strings that are related to it by the grammar:

(4) $Gen_G(F) = \{s \mid \exists c, \phi \text{ s.t. } \langle s, c, \phi, F\rangle \in \Delta_G\}$.

Our main result is that for a certain subclass of LFG grammars the set $Gen_G(F)$ is a context-free language. In the next section we prove that this is the case by constructing a context-free grammar that accepts exactly this set of strings. Our proof depends on the fact that the input $F$—and hence the range of $\phi$—is fully specified; Dymetman (1991), van Noord (1993), and Wedekind (1999) have shown that the general problem of generating from an underspecified input is unsolvable. We return to this issue at the end of the paper and observe that for certain limited forms of underspecification the context-free result can still be established. Our proof also depends on the fact that, with minor exceptions, the instantiated descriptions are idempotent: if $p$ is a particular instantiated proposition, then a description containing two occurrences of $p$ is logically equivalent to one containing just a single occurrence. This means that descriptions can be collected by the union operator for ordinary sets rather than by multi-set union.

The standard LFG formalism includes a number of notational conveniences that make it easy to express linguistic generalizations but which would add complexity to our mathematical analysis. We make a number of simplifying transformations, without loss of generality. The LFG c-structure notation allows the right-hand sides of rules to denote arbitrary regular languages, expressed by Boolean combinations of regular predicates (Kaplan 1995, Kaplan and Maxwell 1996). We assume that these languages are normalized to standard regular expressions involving only concatenation, disjunction, and Kleene-star, and then transform the grammar so that the right sides of the productions denote only finite sequences of annotated categories. First, the effects of any Kleene-stars are removed in the usual way by the introduction of additional nonterminal categories and the rules necessary to expand them appropriately. Second, every category X with disjunctive annotations is replaced by a disjunction of X's each associated with one of the alternatives of the original disjunction. Finally, rules with disjunctive right sides are replaced by sets of rules each of which expands to one of the alternative right-side category sequences. The result of these transformations is a set 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. The Kleene-star conversions produce c-structures from which the original ones can be systematically recovered.

The full LFG formalism allows for grammars that assign cyclic and otherwise linguistically unmotivated structures to sentences. The context-free result can be established for these grammars, but the argument would require a longer and more technical presentation than we can provide in this paper. Thus, without loss of linguistic relevance, we concentrate here on a restricted class of LFG grammars, those that assign acyclic f-structures to sentences. For our purposes, then, an LFG grammar $G$ is a 4-tuple $\langle N, T, \mathrm{S}, R\rangle$ where $N$ is the set of nonterminal categories, $T$ is the set of terminal symbols (the lexical items), $\mathrm{S} \in N$ is the root category, and $R$ is the set of annotated productions. The context-free skeletons of the rules are of the form $\mathrm{X}_0 \rightarrow \mathrm{X}_1..\mathrm{X}_n$ or $\mathrm{X} \rightarrow \mathrm{a}$, with $\mathrm{X}_1..\mathrm{X}_n \in N^*$ and $\mathrm{a} \in T$. If the annotations of a nonterminal daughter establish a relationship between $\downarrow$ and $\uparrow$, then $\downarrow$ is either identified with $\uparrow$, the value of an attribute in $\uparrow$ ($(\uparrow\ \sigma) = \downarrow$), or the member of a set in $\uparrow$ ($\downarrow \in (\uparrow\ \sigma)$), where $\sigma$ is a possibly empty sequence of attributes.

## 2 A Context-free Grammar for $Gen_G(F)$

An input structure $F$ for generation is presented as a hierarchical attribute-value matrix such as the one in Figure 1, repeated here in (5).

(5)
$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'STUDENT'} \\ \text{NUM} & \text{SG} \\ \text{SPEC} & \text{INDEF} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\ \text{PRED} & \text{'FALL}\langle(\text{SUBJ})\rangle\text{'} \\ \text{TENSE} & \text{PAST} \end{bmatrix}$$

An f-structure is an attribute-value structure where the values are either subsidiary attribute-value matrices, symbols, semantic forms, or sets of subsidiary structures (not shown in this example).

(6) A structure $g$ is *contained in* a structure $f$ if and only if:
$g = f$,
$f$ is a set and $g$ is contained in an element of $f$, or
$f$ is an f-structure and $g$ is contained in $(fa)$ for some attribute $a$.

In essence, $g$ is contained in $f$ if $g$ can be located in $f$ by ignoring some enclosing superstructure. For any f-structure $f$, the set of all units contained in $f$ is then defined as in (7).

(7) $\mathit{Units}(f) = \{g \mid g \text{ is contained in } f\}$

Note that $\mathit{Units}(f)$ is a finite set for any $f$, and $\mathit{Units}(f)$ is the range of any $\phi$ that $\Delta_G$ associates with a particular input $F$.

The c-structures and $\phi$ correspondences for $F$ are the unknowns that must be discovered in the process of generation so that the proper instantiated descriptions can be constructed and evaluated. However, since there is only a finite number of possible terms that can be used to designate the units of $F$, we can produce a (finite) superset of the proper instantiated descriptions without knowing in advance the details of either the c-structure or a particular $\phi$.

Let $F$ be an f-structure that has $m$ ($m \geq 0$) set elements. We introduce $m+1$ distinct variables $v_0, .., v_m$ which denote biuniquely the root unit of $F$ ($v_0$) and each set element of $F$ ($v_i$, $i > 0$).[1] We consider the set of all designators of the form $(v_i\ \sigma)$ which are defined in $F$, where $\sigma$ is a (possibly empty) sequence of attributes. The set of designators for a particular unit corresponds, of course, to the set of all possible f-structure paths from one of the $v_i$ roots to that unit. Thus, the set of designators for all units of $F$ is finite, since the number of units of $F$ is finite and there are no cycles in $F$.

The set of variables that we will use to construct the instantiated descriptions is the set $V$ consisting of all $v_t$ where $t$ is a designator of the set just defined. If $l$ is the maximal arity of the rules in $G$, we will consider for the instantiation the set $\mathcal{I}$ consisting of all sequences $\langle v_{t_0}, v_{t_1}, .., v_{t_j}\rangle$ of variables of $V$ of length $1, .., n+1$, not containing any set element variable $v_{v_i}$ ($i = 0, .., m$) more than once. On the basis of this (finite) set of sequences, we define a (partial) function *ID* which assigns to each rule $r \in R$ and each sequence $I \in \mathcal{I}$ that is appropriate for $r$ an instantiated description.

Let $r$ be an n-ary LFG rule

$$\begin{array}{ccc} \mathrm{X}_0 & \rightarrow & \mathrm{X}_1 \ldots \mathrm{X}_n \\ & & S_1 \quad S_n \end{array}$$

with annotated functional schemata $S_1 ... S_n$. A sequence of variables $I \in \mathcal{I}$ is *appropriate for* $r$ if $I = \langle v_{t_0}, v_{t_1}, .., v_{t_n}\rangle$ is of length $n+1$ and

$$t_j = \begin{cases} (v_i\ \sigma'\sigma) & \text{if } t_0 = (v_i\ \sigma') \text{ and } (\uparrow\ \sigma) = \downarrow \in S_j \\ \text{a set element variable } v_j & \text{if } \downarrow \in (\uparrow\ \sigma) \in S_j \end{cases}$$

for all $j = 1, .., n$ ($\sigma'$ and $\sigma$ are (possibly empty) sequences of attributes). (Note that $(\uparrow\ \sigma) = \downarrow$ reduces to $\uparrow = \downarrow$ if $\sigma$ is empty.) If $I$ is appropriate for $r$, then $\mathit{ID}(r, I)$, the instantiated description for $r$ and $I$, is defined as follows:

(8) $$\mathit{ID}(r, I) = \bigcup_{j=1}^{n} \mathit{Inst}(S_j, v_{t_0}, v_{t_j}),$$

where $\mathit{Inst}(S_j, v_{t_0}, v_{t_j})$ is the instantiated description produced by substituting $v_{t_0}$ for all occurrences of $\uparrow$ in $S_j$ and substituting $v_{t_j}$ for all occurences of $\downarrow$ in $S_j$.

If $r$ is a lexical rule with a context-free skeleton of the form $\mathrm{X} \rightarrow$ a every sequence $I = \langle v_{t_0}\rangle$ of length 1 is *appropriate for* $r$ and *ID* is defined by:

(9) $\mathit{ID}(r, I) = \mathit{Inst}(S_1, v_{t_0})$.

The instantiation using appropriate sequences of variables, although finite, permits an effective discrimination of the f-structure variables, since it provides different variables for the $\downarrow$'s associated with different daughters that have different function assignments (i.e., annotations of the form $(\uparrow\ \sigma) = \downarrow$ and $(\uparrow\ \sigma') = \downarrow$ with $\sigma \neq \sigma'$), but identifies variables where f-structure variables are identified explicitly ($\uparrow = \downarrow$) or where the identity follows by unification, as in cases where the annotations of two different daughters contain the same function-assigning equation $(\uparrow\ \sigma) = \downarrow$. Hence, we in fact have enough variables to make all the distinctions that could arise from any c-structure and $\phi$ correspondence for the given f-structure.

The set of all possible instantiated descriptions is large but finite, since $R$ and $\mathcal{I}$ are finite. Thus, the set $\mathit{IP}(F)$ of all possible instantiated propositions for $G$ and $F$ is also large but finite.

(10) $\mathit{IP}(F) = \bigcup \mathit{Range}(\mathit{ID})$

For the construction of the context-free grammar we have to consider those subsets of $\mathit{IP}(F)$ which have $F$ as their minimal model. This is the set $D(F)$, again finite.

[1] Multi-rooted structures would require a whole set of root variables, similar to set elements.

(11) $D(F)$ is the set of all $D \subseteq IP(F)$ such that $F$ is a minimal model for $D$.

We are now prepared to establish the main result of this paper:

(12) *Let $G$ be an LFG grammar conforming to the restrictions we have described. Then for any f-structure $F$, the set $Gen_G(F)$ is a context-free language.*

*Proof:* If $F$ is incomplete or incoherent, then $Gen_G(F)$ is the empty context-free language. Let $G = \langle N, T, \mathrm{S}, R\rangle$ be an LFG grammar. If $D(F)$ is empty, then $Gen_G(F)$ is again the empty context-free language. If $D(F)$ is not empty, we construct a context-free grammar $G_F = \langle N_F, T_F, \mathrm{S}_F, R_F\rangle$ in the following way.

The collection of nonterminals $N_F$ is the (finite) set $\{\mathrm{S}_F\} \cup N \times V \times Pow(IP(F))$, where $\mathrm{S}_F$ is a new root category. Categories in $N_F$ other than $\mathrm{S}_F$ are written X:$v$:$D$, where X is a category in $N$, $v$ is contained in $V$, and $D$ is an instantiated description in $Pow(IP(F))$. $T_F$ is the set $T \times \{\emptyset\} \times \{\emptyset\}$. The rules $R_F$ are constructed from the annotated rules $R$ of $G$. We include all rules of the form:

(i) $\mathrm{S}_F \rightarrow \mathrm{S}{:}v_{v_0}{:}D$, for every $D \in D(F)$

(ii) $\mathrm{X}_0{:}v_{t_0}{:}D_0 \rightarrow \mathrm{X}_1{:}v_{t_1}{:}D_1..\mathrm{X}_n{:}v_{t_n}{:}D_n$ s.t.

(a) there is an $r \in R$ expanding $\mathrm{X}_0$ to $\mathrm{X}_1..\mathrm{X}_n$,

(b) $D_0 = ID(r, \langle v_{t_0}, v_{t_1}, .., v_{t_n}\rangle) \cup \bigcup_{i=1}^{n} D_i$,

(c) if $v_{v_i} \in (v_{t_j}\ \sigma)$ belongs to $D_j$ then $v_{v_i} \neq v_{t_k}$ $(k = 1, .., n)$ and $\neg\exists D_h$ $(h \neq j)$ s.t. $v_{v_i} \in (v_{t_h}\ \sigma') \in D_h$,[2]

(iii) X:$v_t$:$D \rightarrow$ a:$\emptyset$:$\emptyset$ s.t.

(a) there is an $r \in R$ expanding X to a,

(b) $D = ID(r, \langle v_t\rangle)$.

We define the projection $Cat(x{:}y{:}z) = x$ for every category in $N_F \cup T_F$ and extend this function in the natural way to strings of categories and sets of strings of categories. Note that the set

$$Cat(L(G_F)) = \{s \mid \exists w \in L(G_F) \text{ s.t. } Cat(w) = s\}$$

is context-free, since the set of context-free languages is closed under homomorphisms such as *Cat*. We show that the language $Cat(L(G_F)) = Gen_G(F)$.

We prove first that $Gen_G(F) \subseteq Cat(L(G_F))$. Let $c$ be an annotated c-structure of a string $s$ with f-structure $F$ in $G$. On the basis of $c$ and $F$ we construct a derivation tree of a string $s'$ in $G_F$ with $Cat(s') = s$ in two steps. In the first step we relabel each terminal node with label a by a:$\emptyset$, the root by S:$v_{v_0}$, each node introducing a set element with label X biuniquely by X:$v_{v_i}$, and each other node labelled X by X:$v_t$ where $t$ is a designator that is constructable from the function-assigning equations of the annotations along the path from the unique root or set element to that node. On the basis of this relabelled c-structure we construct a derivation tree of $s'$ in $G_F$ bottom-up. We relabel each terminal node with label a:$\emptyset$ by a:$\emptyset$:$\emptyset$ and each preterminal node with label X:$v_t$ by X:$v_t$:$D$ where $D$ is defined as in (iiib) with $r$ expanding X in $c$ to a. Suppose we have constructed the subtrees dominated by $\mathrm{X}_1{:}v_{t_1}{:}D_1..\mathrm{X}_n{:}v_{t_n}{:}D_n$, the corresponding subtrees in $c$ are derived with $r$ expanding $\mathrm{X}_0$ to $\mathrm{X}_1..\mathrm{X}_n$, and the mother node is relabelled by $\mathrm{X}_0{:}v_{t_0}$. We then relabel this mother node by $\mathrm{X}_0{:}v_{t_0}{:}D_0$ where $D_0$ is determined according to (iib). By induction on the depth of the subtrees it is then easy to verify that the instantiated description $D$ of a subtree dominated by X:$v_t$:$D$ is equivalent to the f-description of the corresponding annotated subtree in $c$. Thus, $F$ must be a minimal model of the instantiated description of the root label S:$v_{v_0}$:$D_F$, $\mathrm{S}_F$ derives S:$v_{v_0}$:$D_F$ in $G_F$ and $Cat(s') = s$.

We now show that $Cat(L(G_F)) \subseteq Gen_G(F)$. Let $c''$ be a derivation tree of $s'$ in $G_F$ with $Cat(s') = s$ and suppose that the root (with label $\mathrm{S}_F$) expands to S:$v_{v_0}$:$D_F$. We construct a new derivation tree $c'$ that results from $c''$ by eliminating the root. We then define a function $\phi'$ such that for each nonterminal node $\mu$ of $c'$: $\phi'(\mu) = v_t$ if $\mu$ is labelled by X:$v_t$:$D$ in $c'$. According to our rule construction it can easily be seen by induction on the depth of the subtrees that there must be an annotated c-structure $c$ of $G$ with the same underlying tree structure as $c'$ such that for each node $\mu$ labelled by $x{:}y{:}D$ in $c'$:
(i) $\mu$ is labelled by $x$ in $c$,
(ii) $D$ is identical with the description that results from $D_\mu$, the f-description of the sub-c-structure dominated by $\mu$ in $c$, by replacing each occurrence of an f-structure variable '$\phi(\nu)$' (usually abbreviated by $f_\nu$) in $D_\mu$ by $\phi'(\nu)$. Since $\phi(\mu) = \phi(\nu)$ follows for two f-structure designators if $\phi'(\mu) = \phi'(\nu)$, the f-description of the whole c-structure must be equivalent to $D_F$ and thus $\Delta_G(s, c, \phi, F)$ where $\phi = \phi' \circ \phi_V$ and $\phi_V$ is the unique function that maps each $v_t$ to the unit of $F$ that is denoted by $t$. QED

## 3 An Example

As a simple illustration, we produce the context-free grammar $G_F$ for the input (5) and the grammar in (2,3) above. The only designator variables that will yield useful rules are $v_{v_0}$ and $v_{(v_0\ \mathrm{SUBJ})}$, in the following abbreviated by $v$ and $v_\mathrm{S}$. Consider first the context-free rules that correspond to the rules that generate NP's. If we choose the sequence $I = \langle v_\mathrm{S}\rangle$, the instantiated description for the determiner rule in (3a) is (13).

(13) $\{(v_\mathrm{S}\ \mathrm{SPEC}) = \mathrm{INDEF}, (v_\mathrm{S}\ \mathrm{NUM}) = \mathrm{SG}\}$

[2] This condition captures LFG's special interpretation of membership statements. The proper treatment of LFG's semantic forms requires a similar condition.

Rule (14) is thus a production of $G_F$.

$$(14)\quad \text{DET}{:}v_{\text{S}}{:}\left\{\begin{array}{c}(v_{\text{S}}\ \text{SPEC}) = \text{INDEF}\\ (v_{\text{S}}\ \text{NUM}) = \text{SG}\end{array}\right\} \rightarrow \text{a}{:}\emptyset{:}\emptyset$$

Rule (15) is obtained from the N rule using the same sequence.

$$(15)\quad \text{N}{:}v_{\text{S}}{:}\left\{\begin{array}{c}(v_{\text{S}}\ \text{PRED}) = {'}\text{STUDENT}{'}\\ (v_{\text{S}}\ \text{NUM}) = \text{SG}\\ (v_{\text{S}}\ \text{SPEC})\end{array}\right\} \rightarrow \text{student}{:}\emptyset{:}\emptyset$$

For the NP rule and the sequence $\langle v_{\text{S}}, v_{\text{S}}, v_{\text{S}}\rangle$, both daughter annotations instantiate to the trivial description $v_{\text{S}} = v_{\text{S}}$, and this can combine with many daughter descriptions. Two of these are the basis for the rules (16) and (17). The daughter categories of rule (16) match the mother categories of rules (14) and (15), and the three rules together can derive the string a:$\emptyset$:$\emptyset$ student:$\emptyset$:$\emptyset$. Rule (17), on the other hand, is a legitimate rule but does not combine with any others to produce a terminal string. It is a useless, albeit harmless, production; if desired, it can be removed from the set of productions by standard algorithms for context-free grammars.

If we continue along in this manner, we find that the rules in (18,19,20) are the only other useful rules that belong to $G_F$.

The grammar $G_F$ also includes the following starting rule:

$$(21)\quad \text{S}_F \rightarrow \text{S}{:}v{:}\left\{\begin{array}{c}(v\ \text{SUBJ}) = v_{\text{S}}\\ (v_{\text{S}}\ \text{CASE}) = \text{NOM}\\ v = v\\ (v\ \text{TENSE})\\ v_{\text{S}} = v_{\text{S}}\\ (v_{\text{S}}\ \text{SPEC}) = \text{INDEF}\\ (v_{\text{S}}\ \text{NUM}) = \text{SG}\\ (v_{\text{S}}\ \text{PRED}) = {'}\text{STUDENT}{'}\\ (v_{\text{S}}\ \text{SPEC})\\ (v\ \text{PRED}) = {'}\text{FALL}\langle(\text{SUBJ})\rangle{'}\\ (v\ \text{TENSE}) = \text{PAST}\end{array}\right\}$$

This grammar provides one derivation for a single string, a:$\emptyset$:$\emptyset$ student:$\emptyset$:$\emptyset$ fell:$\emptyset$:$\emptyset$. Applying *Cat* to this string gives 'a student fell', the only sentence that this grammar associates with the input f-structure.

## 4 Consequences and Observations

Our main result offers a new way to conceptualize the problem of generation for LFG and other higher-order context-free-based grammatical formalisms. The proof of the theorem is constructive: it indicates precisely how to build the grammar $G_F$ whose language is the desired set of strings. Thus, the problem of LFG generation is divided into two phases, constructing the context-free grammar $G_F$, and then using a standard context-free generation algorithm to produce strings from it.

We can regard the first phase of LFG generation as specializing the original LFG grammar to one that only produces the given input f-structure. This specialization refines the context-free backbone of the original grammar, but our theorem indicates that the input f-structure provides enough information so that, in effect, the metavariables in the functional annotations can all be replaced by variables contained in a fixed finite set. Thus, in the LFG generation case the specialized grammar turns out to be in a less powerful formal class than the original. We can understand different aspects of generation as pertaining either to the way the grammar is constructed or to well-known properties of context-free grammars and context-free generation.

It follows as an immediate corollary, for example, that it is decidable whether the set $Gen_G(F)$ is empty, contains a finite number of strings, or contains an infinite number of strings. This can be determined by inspecting $G_F$ with standard context-free tools, once it has been constructed. If the language is infinite, we can make use of the context-free pumping lemma to identify a finite number of short strings from which all other strings can be produced by repetition of subderivations. Wedekind (1995) first established the decidability of LFG generation and proved a pumping lemma for the generated string set; our theorem provides alternative and very direct proofs of these previously known results.

We also have an explanation for another observation of Wedekind (1995). Kaplan and Bresnan (1982) showed that the Nonbranching Dominance Condition (sometimes called Offline Parsability) is a sufficient condition to guarantee decidability of the membership problem. Wedekind noted, however, that this condition is not necessary to determine whether a given f-structure corresponds to any strings. We now see more clearly why this is the case: if there is a context-free derivation for a given string that involves a nonbranching dominance cycle, we know (from the pumping lemma) that there is another derivation for that same string that has no such cycle. Thus, the generated language is the same whether or not derivations with nonbranching dominance cycles are allowed.

There is a practical consequence to the two phases of LFG generation. The grammar $G_F$ can be provided to a client as a finite representation of the set of perhaps infinitely many strings that correspond to the given f-structure, and the client can then control the process of enumerating individual strings. The client may choose simply to produce the shortest ones just by avoiding recursive category expansions. Or the client may apply the technology of stochastic context-free grammars to choose the most probable sentence from the set of possibilities. The client may also be interested in strings that meet further conditions that the shortest or most probable strings fail to satisfy; in this case the client may

(16) $\text{NP}{:}v_s{:}\left\{\begin{array}{c}v_s = v_s\\(v_s\ \textsc{spec}) = \textsc{indef}\\(v_s\ \textsc{num}) = \textsc{sg}\\(v_s\ \textsc{pred}) = {'}\textsc{student}{'}\\(v_s\ \textsc{spec})\end{array}\right\} \rightarrow \text{DET}{:}v_s{:}\left\{\begin{array}{c}(v_s\ \textsc{spec}) = \textsc{indef}\\(v_s\ \textsc{num}) = \textsc{sg}\end{array}\right\}\ \text{N}{:}v_s{:}\left\{\begin{array}{c}(v_s\ \textsc{pred}) = {'}\textsc{student}{'}\\(v_s\ \textsc{num}) = \textsc{sg}\\(v_s\ \textsc{spec})\end{array}\right\}$

(17) $\text{NP}{:}v_s{:}\{v_s = v_s, (v_s\ \textsc{num}) = \textsc{sg}\} \rightarrow \text{DET}{:}v_s{:}\{(v_s\ \textsc{num}) = \textsc{sg}\}\ \text{N}{:}v_s{:}\emptyset$

(18) $\text{V}{:}v{:}\left\{\begin{array}{c}(v\ \textsc{pred}) = {'}\textsc{fall}\langle(\textsc{subj})\rangle{'}\\(v\ \textsc{tense}) = \textsc{past}\end{array}\right\} \rightarrow \text{fell}{:}\emptyset{:}\emptyset$

(19) $\text{VP}{:}v{:}\left\{\begin{array}{c}v = v\\(v\ \textsc{pred}) = {'}\textsc{fall}\langle(\textsc{subj})\rangle{'}\\(v\ \textsc{tense}) = \textsc{past}\end{array}\right\} \rightarrow \text{V}{:}v{:}\left\{\begin{array}{c}(v\ \textsc{pred}) = {'}\textsc{fall}\langle(\textsc{subj})\rangle{'}\\(v\ \textsc{tense}) = \textsc{past}\end{array}\right\}$

(20) $\text{S}{:}v{:}\left\{\begin{array}{c}(v\ \textsc{subj}) = v_s\\(v_s\ \textsc{case}) = \textsc{nom}\\v = v\\(v\ \textsc{tense})\\v_s = v_s\\(v_s\ \textsc{spec}) = \textsc{indef}\\(v_s\ \textsc{num}) = \textsc{sg}\\(v_s\ \textsc{pred}) = {'}\textsc{student}{'}\\(v_s\ \textsc{spec})\\(v\ \textsc{pred}) = {'}\textsc{fall}\langle(\textsc{subj})\rangle{'}\\(v\ \textsc{tense}) = \textsc{past}\end{array}\right\} \rightarrow \text{NP}{:}v_s{:}\left\{\begin{array}{c}v_s = v_s\\(v_s\ \textsc{spec}) = \textsc{indef}\\(v_s\ \textsc{num}) = \textsc{sg}\\(v_s\ \textsc{pred}) = {'}\textsc{student}{'}\\(v_s\ \textsc{spec})\end{array}\right\}\ \text{VP}{:}v{:}\left\{\begin{array}{c}v = v\\(v\ \textsc{pred}) = {'}\textsc{fall}\langle(\textsc{subj})\rangle{'}\\(v\ \textsc{tense}) = \textsc{past}\end{array}\right\}$

apply the pumping lemma to systematically produce longer strings for examination.

Our recipe for constructing $G_F$ may produce many categories and expansion rules that cannot play a role in any derivation, either because they are inaccessible from the root symbol, they do not lead to a terminal string, or because they involve individual descriptions that $F$ does not satisfy. Having constructed the grammar, we can again apply standard context-free methods, this time to put the grammar in a more optimal form by removing useless categories and productions. We can view several different generation algorithms as strategies for avoiding the creation of useless categories in the first place.

The most obvious optimization, of course, is to incrementally evaluate all the instantiated descriptions and remove from consideration categories and rules involving descriptions for which $F$ is not a model. A second strategy is to construct the grammar in bottom-up fashion. We begin by comparing the terminal rules of the LFG grammar with the features of the input f-structure, and construct only the corresponding categories and rules that meet the criteria in (iii) above. We then construct rules that can derive the mother categories of those rules, and so on. With this strategy we insure that every category we construct can derive a terminal string, but we have no guarantee that every bottom-up sequence will reach the root symbol.

It is also appealing to construct the grammar by means of a top-down process. If we start with an agenda containing the root symbol, create rules only to expand categories on the agenda, and place categories on the agenda whenever they appear for the first time on the right side of a new rule, we get the effect of a top-down exploration of the grammar. We will only create categories and rules that are accessible from the root symbol, but we may still produce categories that derive no terminal string.

The top-down strategy may not provide effective guidance, however, if the set $D(F)$ contains many alternative descriptions of $F$. But suppose we can associate with every instantiated description $D$ a unique canonical description that has the same f-structure as its minimal model, and suppose that we then reformulate the grammar construction in terms of such canonical descriptions. This can sharply reduce the size of the grammar we produce according to any enumeration strategy, since it avoids rules and categories that express only uninformative variation. It can particularly benefit a top-down enumeration because the set $D(F)$ will have at most one canonical member. Presumably any practical generation scheme will define and operate on canonical descriptions of some sort, but our context-free result does not depend on whether or how such descriptions might be specified and manipulated.

Just as for context-free parsing, there are a number of mixed strategies that take top-down and bottom-up information into account at the same time. We can use a precomputed reachability table to guide the process of top-down exploration, for instance. Or we can simulate a left-corner enumeration of the search space, considering categories that are reachable from a current goal category and

match the left corner of a possible rule. In general, almost any of the traditional algorithms for processing context-free grammars can be reformulated as a strategy for avoiding the creation of useless categories and rules. Other enumeration strategies focus on the characteristics of the input f-structure. A head-driven strategy (e.g. van Noord 1993) identifies the lexical heads first, finds the rules that expand them, and then uses information associated with those heads, such as their grammatical function assignments, to pick other categories to expand.

Our proof depends on the assumption that the input $F$ is fully specified so that the set of possible instantiations is finite. Dymetman (1991), van Noord (1993), and Wedekind (1999) have shown that it is in general undecidable whether or not there are any strings associated with an f-structure that has units in addition to those in the input. Indeed, our proof of context-freeness does not go through if we allow new units to be hypothesized arbitrarily, beyond the ones that appear in $F$; if this is permitted, we cannot establish a finite bound on the number of possible categories. This is unfortunate, since there may be interesting practical situations in which it is convenient to leave unspecified the value of a particular feature. However, if there can be only a finite number of possible values for an underspecified feature, the context-free result can still be established. We create from $F$ a set of alternative structures $F_1..F_n$ by filling in all possible values of the unspecified features, and we produce the context-free grammar corresponding to each of them. Since a finite union of context-free languages is context-free, the set of strings generated from any of these structures remains in that class.

A final comment about the generation problem for other high-order grammatical formalisms. Our proof depends on several features of LFG: the context-free base, the piecewise correspondence of phrase structure and f-structure units, and the idempotency of the functional description language. PATR shares these properties, although the correspondence is implicit in the mechanism and not reified as a linguistically significant concept. So, our proof can be used to establish the context-free result for PATR. On the other hand, it is not clear whether the string set corresponding to an underlying HPSG structure is context-free. HPSG (Pollard and Sag 1994) does not make direct use of a context-free skeleton, and operations other than concatenation may be used to assemble a collection of substrings into an entire sentence. We cannot extend our proof to HPSG unless the effect of these mechanisms can be reduced to an equivalent characterization with a context-free base. However, grammars written for the ALE system's logic of typed feature structures (Carpenter and Penn 1994) do have a context-free component and therefore are amenable to the treatment we have outlined.

**Acknowledgments**

We are indebted to John Maxwell, Hadar Shemtov, Martin Kay, and Paula Newman for many fruitful and insightful discussions of the LFG generation problem, and for criticisms and suggestions that have helped to clarify many of the mathematical and computational issues.

## References

Carpenter, B. and G. Penn. 1994. ALE 2.0 User's Guide. Technical report, Carnegie Mellon University, Pittsburgh, PA.

Dymetman, M. 1991. Inherently Reversible Grammars, Logic Programming and Computability. In *Proceedings of the ACL Workshop: Reversible Grammar in Natural Language Processing*. Berkeley, CA, pages 20–30.

Kaplan, R. M. 1995. The Formal Architecture of Lexical-Functional Grammar. In M. Dalrymple, R. M. Kaplan, J. Maxwell, and A. Zaenen, editors, *Formal Issues in Lexical-Functional Grammar*. CSLI Publications, Stanford, CA, pages 7–27.

Kaplan, R. 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, R. M. and J. Maxwell. 1996. *LFG Grammar Writer's Workbench*. Technical report, Xerox Palo Alto Research Center. At http://ftp.parc.xerox.com/pub/lfg/lfgmanual.ps.

Pollard, C. and I. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago, IL.

Shieber, S., H. Uszkoreit, F. Pereira, J. Robinson, and M. Tyson. 1983. The Formalism and Implementation of PATR-II. In B. Grosz and M. Stickel, editors, *Research on Interactive Acquisition and Use of Knowledge*. SRI Final Report 1894. SRI International, Menlo Park, CA, pages 39–79.

van Noord, G. 1993. Reversibility in Natural Language Processing. Ph.D. thesis, Rijksuniversiteit Utrecht.

Wedekind, J. 1995. Some Remarks on the Decidability of the Generation Problem in LFG- and PATR-style Unification Grammars. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics*. Dublin, pages 45–52.

Wedekind, J. 1999. Semantic-driven Generation with LFG- and PATR-style Grammars. *Computational Linguistics*, 25(2): 277–281.