On the Universal Generation Problem for Unification Grammars

Jürgen Wedekind*
University of Copenhagen

The universal generation problem for unification grammars is the problem of determining whether a given grammar derives any terminal string with a given feature structure. It is known that the problem is decidable for LFG and PATR grammars if only acyclic feature structures are taken into consideration. In this brief note, we show that the problem is undecidable for cyclic structures. This holds even for grammars that are off-line parsable.

The universal generation problem for unification grammars is the problem of determining for an arbitrary grammar G and an arbitrary feature structure F whether there exists at least one sentence that G derives with F. If F is acyclic, Wedekind and Kaplan (2012) have shown that the problem is decidable for LFG (Kaplan and Bresnan 1982) and PATR (Shieber et al. 1983) grammars. They prove that the set of strings that a grammar relates to an acyclic feature structure can be described by a context-free grammar. Decidability of the problem then follows because the emptiness problem is decidable for context-free languages. For cyclic feature structures they demonstrated by example that the set of strings that a grammar relates to an input might not be context-free, but they did not further investigate the formal properties of the languages that are in general related to cyclic structures.

In this brief note, we show the undecidability of the universal generation problem by reduction from the undecidable emptiness problem for the intersection of two context-free languages. We provide a proof for LFG- or PATR-style grammars that associate feature structures with trees derived in accordance with a context-free grammar. Our result also applies to other systems such as HPSG (Pollard and Sag 1994) whose formal devices are powerful enough to simulate, albeit indirectly, the effect of context-free derivation.

To state the universal generation problem more formally, recall that a unification grammar G defines a binary derivation relation Δ_G between terminal strings and feature structures, as given in (1).

(1) $\Delta_G(s, F)$ iff G derives terminal string s with feature structure F

The universal generation problem is then the problem of deciding for an arbitrary unification grammar G and an arbitrary feature structure F whether $\{s \mid \Delta_G(s, F)\}$ is empty or not.

* Center for Language Technology, University of Copenhagen, Njalsgade 140, 2300 Copenhagen S, Denmark. E-mail: jwedekind@hum.ku.dk.

Submission received: 29 October 2013; accepted for publication: 27 January 2014.

doi:10.1162/COLI_a_00191

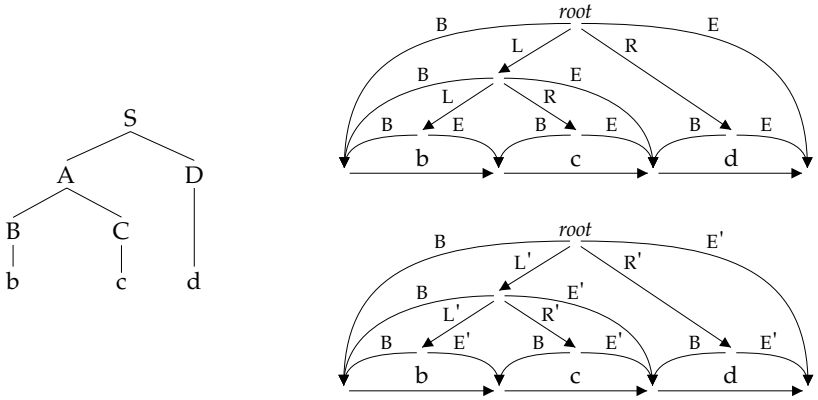


Figure 1
A sample c-structure and the f-structures associated with it by type 1 (top) and type 2 (bottom) string grammar derivations.

For the reduction of the emptiness problem for the intersection of two context-free languages, we can, without loss of generality, assume that the context-free languages are ϵ-free. These languages can be described by grammars in Chomsky normal form, that is, by context-free grammars $G = (N, T, S, P)$ with nonterminal vocabulary N, terminal vocabulary T, and start symbol S where every rule in P is of the form $A \rightarrow BC$ with $B, C \in N$, or $A \rightarrow a$ with $a \in T$.

For the proof we first define for each context-free grammar G in Chomsky normal form two LFG grammars that both derive $L(G)$ and that associate with each derivable terminal string feature structures (f-structures) that provide slightly different encodings of the derivable string.

Let $G = (N, T, S, P)$ be a context-free grammar in Chomsky normal form. A **type 1 string grammar** $String_1(G)$ for G is an LFG grammar (N, T, S, P') whose rule set P' includes for each rule $A \rightarrow BC$ in P a rule of the form (2a) and for each rule $A \rightarrow a$ in P a rule of the form (2b).

(2) a. $A \rightarrow$ B ($\uparrow$ L) $= \downarrow$, ($\uparrow$ B) $= (\downarrow$ B) C ($\uparrow$ R) $= \downarrow$, ($\uparrow$ E) $= (\downarrow$ E), ($\uparrow$ L E) $= (\downarrow$ B)

b. $A \rightarrow$ a ($\uparrow$ B a) $= (\uparrow$ E)

A **type 2 string grammar** $String_2(G)$ for G is an LFG grammar (N, T, S, P') whose rule set P' includes a rule of the form (3a) for each $A \rightarrow BC$ in P and a rule of the form (3b) for each $A \rightarrow a$ in P.

(3) a. $A \rightarrow$ B ($\uparrow$ L') $= \downarrow$, ($\uparrow$ B) $= (\downarrow$ B) C ($\uparrow$ R') $= \downarrow$, ($\uparrow$ E') $= (\downarrow$ E'), ($\uparrow$ L' E') $= (\downarrow$ B)

b. $A \rightarrow$ a ($\uparrow$ B a) $= (\uparrow$ E')

Figure 1 illustrates a c-structure and the f-structures associated with it by type 1 and type 2 string grammar derivations.[1] The attributes L, R, B, and E are mnemonic

1 Note that the terminal symbols also occur as attributes in the annotations of the terminal rules. This "abuse" of the terminal symbols is not essential to our argument (a set of new attributes that is in one-to-one correspondence with the set of terminals would also suffice), but it makes the encoding of the terminal strings in the f-structures more perspicuous.

for 'left', 'right', 'begin', and 'end', respectively. For later reference, we also depicted the constant *root* that we uniformly use to instantiate the ↑ of a derivation that refers to the c-structure root; *root* then labels the f-structure element to which it refers in the minimal model of the f-description. (In Kaplan and Bresnan's [1982] terminology, *root* corresponds to the f-structure variable associated with the c-structure root, usually notated by f_1.)

Both types of string grammars have in common that they have G as their context-free skeleton and that for every string in $L(G)$, the f-structure for each string grammar encodes both the string itself and also the branching structure of a derivation in G that leads to that terminal string. The f-structures derived by the two types of grammars vary only slightly in the labels that they use to encode those properties. An f-structure of a type 2 grammar derivation for a given string shares the 'begin' attribute (B) with the f-structure of a corresponding type 1 grammar derivation, but it has distinct 'left', 'right', and 'end' attributes (L', R', E').

Because the derived f-descriptions can never be unsatisfiable (the string grammars do not contain atomic values), the f-structure constraints of the string grammars do not actually filter the language of the context-free grammar. Thus G and its string grammars must have the same language $L(G) = L(String_1(G)) = L(String_2(G))$. By induction on the depth of the derivation trees it is also easy to see that the minimal solution of the f-description of a derivation of a terminal string s is acyclic and single-rooted, and satisfies $(root\ \mathrm{B}\ s') = (root\ \mathrm{E})$ and $(root\ \mathrm{B}\ s') = (root\ \mathrm{E}')$, respectively, if and only if $s' = s$. That is, these grammars both encode their terminal strings in their respective $(root\ \mathrm{B})$ to $(root\ \mathrm{E})/(root\ \mathrm{E}')$ paths.

Before going into the details of the undecidability proof, we first give an outline of the proof idea. For the reduction, we have to construct for two arbitrary ϵ-free context-free languages L_1 and L_2 an LFG grammar G and an input structure F such that the set of terminal strings that G derives with F is empty if and only if the intersection of L_1 and L_2 is empty. Because every ϵ-free context-free language is derivable by a context-free grammar in Chomsky normal form, we can make the LFG grammar G by combining the productions of $String_1(G_1)$ and $String_2(G_2)$, for two arbitrary context-free grammars $G_1 = (N_1, T_1, \mathrm{S}_1, P_1)$ and $G_2 = (N_2, T_2, \mathrm{S}_2, P_2)$ in Chomsky normal form. To avoid undesired interactions between the rules of the two string grammars, we assume that the sets of nonterminals of G_1 and G_2 are disjoint (this is without loss of generality because nonterminals can always be renamed).

We observed already that the string grammars $String_1(G_1)$ and $String_2(G_2)$ associate with any c-structure derivation of a terminal string s_1 in G_1 and any c-structure derivation of a terminal string s_2 in G_2 f-structures that encode s_1 and s_2 as their respective $(root\ \mathrm{B})$ values. By construction of the string grammars, the only paths that the two f-structures share are the paths $(root\ \mathrm{B}\ s')$ where s' is a common prefix of s_1 and s_2. Thus, if we define G to consist of the rules of $String_1(G_1)$ and $String_2(G_2)$, and a start rule that expands S to $\mathrm{S}_1\mathrm{S}_2$ and forces the f-structures for s_1 and s_2 to unify, their $(root\ \mathrm{E})$ and $(root\ \mathrm{E}')$ paths become reentrant $((root\ \mathrm{E}) = (root\ \mathrm{E}'))$ if and only if s_1 and s_2 are identical. G then assigns to a terminal string an f-structure with reentrant $(root\ \mathrm{E})$ and $(root\ \mathrm{E}')$ paths if and only if it has the form $s's'$ and s' is in $L(G_1) \cap L(G_2)$.

If all we do is unification on the top level the f-structures for the strings s_1s_2 would still record information on the structure of their derivation. Thus distinct strings in $\{s's' \mid s' \in L(G_1) \cap L(G_2)\}$ would get assigned distinct f-structures. However, the proof requires that there be a single f-structure that is assigned to all strings $s's'$ with

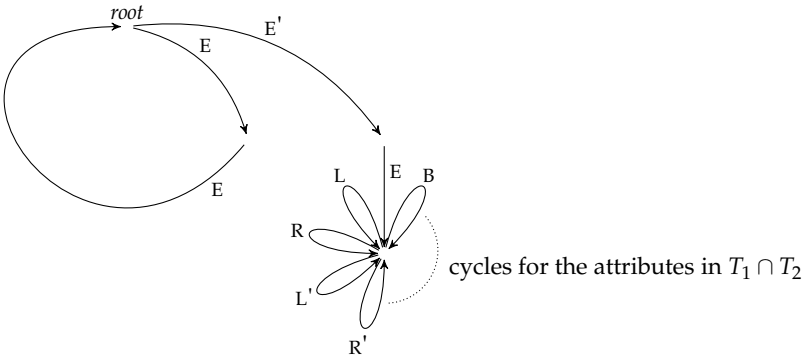
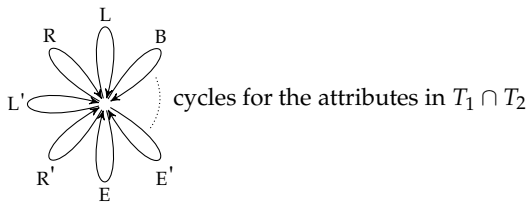


Figure 2
The functional contribution of the S rule to a derivation in G.

s' in $L(G_1) \cap L(G_2)$. We achieve that by annotating the start rule so that the unified f-structures derived by the string grammars are folded up into one and the same cyclic f-structure F. This f-structure consists of a single element (node) and $|T_1 \cap T_2| + 7$ cycles of length 1, each one labeled with one of the attributes in $\{\text{B}, \text{L}, \text{R}, \text{L}', \text{R}', \text{E}, \text{E}'\} \cup (T_1 \cap T_2)$. F thus has the following form.[2]

(4)



F must contain cycles for all terminals in $T_1 \cap T_2$, so that it imposes no constraints on the strings that may appear in $L(G_1) \cap L(G_2)$.

The folding into F is accomplished by annotations of G's start rule whose contribution to a derivation in G is depicted in Figure 2. As earlier, we include *root* for later use. Obviously, if $(\mathit{root}\ \text{E}) = (\mathit{root}\ \text{E}')$ holds in the unified f-structures of the string grammars, then the unification of the string grammar f-structures and the structure in Figure 2 yields F. Otherwise, their unification results in a structure that only properly subsumes F. This is because neither $(\mathit{root}\ \text{E}\ \text{E})$ nor $(\mathit{root}\ \text{E}'\ \text{E})$ exists in the unified f-structures of the two string grammars, and therefore their values in the structure in Figure 2 are not merged when the structures are unified. Thus G derives with F exactly the set of strings $s's'$ with s' in $L(G_1) \cap L(G_2)$. Hence, this set is empty if and only if $L(G_1) \cap L(G_2)$ is empty.

We now give a rigorous statement and proof of our undecidability theorem.

Theorem
For an arbitrary LFG grammar G and an arbitrary f-structure F it is undecidable whether $\{s \mid \Delta_G(s, F)\} = \emptyset$.

2 This f-structure may look peculiar in that it does not contain atomic feature values. However, this is not relevant to the proof. To make the f-structure look more "natural," we can, for example, expand G by an annotation $(\uparrow\ \text{D}) = \text{V}$ at the start rule and F by a feature D with value V.

Proof
Let $G_1 = (N_1, T_1, S_1, P_1)$ and $G_2 = (N_2, T_2, S_2, P_2)$ be two arbitrary context-free grammars in Chomsky normal form. Without loss of generality, we can assume that $N_1 \cap N_2 = \emptyset$. On the basis of $String_1(G_1)$ and $String_2(G_2)$ we construct an LFG grammar $G = (N, T, S, P)$ with $N = N_1 \cup N_2 \cup \{S\}$, $S \notin N_1 \cup N_2$, and $T = T_1 \cup T_2$. The rule set P consists of the rules of $String_1(G_1)$ and $String_2(G_2)$ and the following start rule.

$$
\begin{array}{lccl}
S \rightarrow & S_1 & S_2 & \\
 & \uparrow = \downarrow & \uparrow = \downarrow & \\
 & (\uparrow \mathrm{E}\,\mathrm{E}) = \uparrow & (\uparrow \mathrm{E'}\,\mathrm{E}) = (\uparrow \mathrm{E'}\,\mathrm{E}\,\mathrm{B}) & \\
 & & (\uparrow \mathrm{E'}\,\mathrm{E}) = (\uparrow \mathrm{E'}\,\mathrm{E}\,\mathrm{L}) & \\
 & & (\uparrow \mathrm{E'}\,\mathrm{E}) = (\uparrow \mathrm{E'}\,\mathrm{E}\,\mathrm{R}) & \\
 & & (\uparrow \mathrm{E'}\,\mathrm{E}) = (\uparrow \mathrm{E'}\,\mathrm{E}\,\mathrm{L'}) & \\
 & & (\uparrow \mathrm{E'}\,\mathrm{E}) = (\uparrow \mathrm{E'}\,\mathrm{E}\,\mathrm{R'}) & \\
 & & \vdots & \left.\vphantom{\vdots}\right\} \text{annotations } (\uparrow \mathrm{E'}\,\mathrm{E}) = (\uparrow \mathrm{E'}\,\mathrm{E}\,x) \\
 & & \vdots & \left.\vphantom{\vdots}\right\} \text{for all } x \text{ in } T_1 \cap T_2 \\
 & & \vdots &
\end{array}
$$

The functional contribution of this start rule to a derivation in G is depicted in Figure 2. The $(\uparrow \mathrm{E}\,\mathrm{E}) = \uparrow$ annotation at S_1 introduces the left cycle and the annotations at S_2 account for the rest. Now let F be the f-structure in (4) and consider an arbitrary derivation of a terminal string s with f-description FD in G. By construction of G, s must have the form $s_1 s_2$, with s_1 derived from S_1 and s_2 derived from S_2. We claim $s_1 = s_2$ iff F is the f-structure for FD. Note first that also G does not contain atomic values. Thus, FD cannot be unsatisfiable and must have an f-structure.
If $s_1 = s_2$, then $FD \vdash (root\ \mathrm{E}) = (root\ \mathrm{E'})$, since $(root\ \mathrm{B}\ s_1) = (root\ \mathrm{E})$ and $(root\ \mathrm{B}\ s_2) = (root\ \mathrm{E'})$ follow from FD. From $(root\ \mathrm{E}) = (root\ \mathrm{E'})$ and the instantiated annotations of the S rule, we get $(root\ x) = root$, for all $x \in \{\mathrm{B}, \mathrm{L}, \mathrm{R}, \mathrm{L'}, \mathrm{R'}\} \cup (T_1 \cap T_2)$. With these equations we can then derive from $(root\ \mathrm{B}\ s_1) = (root\ \mathrm{E})$ and $(root\ \mathrm{B}\ s_2) = (root\ \mathrm{E'})$ equations $root = (root\ x)$, for $x \in \{\mathrm{E}, \mathrm{E'}\}$. Thus F must be the f-structure that we obtain from a minimal model of FD.
Now suppose $s_1 \neq s_2$. Let FD_1 and FD_2 be the f-descriptions of the string grammars, $FD'_1 = FD_1 \cup \{root = n_1\}$, $FD'_2 = FD_2 \cup \{root = n_2\}$, with n_1 and n_2 instantiating $\downarrow$ in the annotations at S_1 and S_2, and $FD' = FD'_1 \cup FD'_2$. By construction of G, the only terms shared by the deductive closures of FD'_1 and FD'_2 are the common subterms of $(root\ \mathrm{B}\ s_1)$ and $(root\ \mathrm{B}\ s_2)$. Thus $FD' \nvdash (root\ \mathrm{E}) = (root\ \mathrm{E'})$, because otherwise $FD'_1 \vdash (root\ \mathrm{B}\ s') = (root\ \mathrm{E})$ and $FD'_2 \vdash (root\ \mathrm{B}\ s') = (root\ \mathrm{E'})$ would imply $s' = s_1$ and $s' = s_2$, as we saw earlier. Because obviously $(root\ \mathrm{E}\,\mathrm{E})$ and $(root\ \mathrm{E'}\,\mathrm{E})$ do not occur in any equation derivable from FD', $(root\ \mathrm{E}) = (root\ \mathrm{E'})$ cannot follow from FD either, and F cannot be the f-structure for FD.
Thus $\{s \mid \Delta_G(s, F)\} = \{s's' \mid s' \in L(G_1) \cap L(G_2)\}$ and hence $\{s \mid \Delta_G(s, F)\} = \emptyset$ if and only if $L(G_1) \cap L(G_2) = \emptyset$. Since the emptiness problem for the intersection of context-free languages is in general undecidable, the generation problem must be undecidable too. ■

As a consequence of this theorem we know that there does not exist a general generation algorithm, at least if cyclic input structures are considered as legitimate inputs.

We note that the grammars constructed in this proof are off-line parsable (cf., e.g., Kaplan and Bresnan 1982; Johnson 1988; Jaeger, Francez, and Wintner 2005). Off-line parsability is sufficient to guarantee the decidability of the recognition/parsing problem even for cyclic f-structures. But Wedekind and Kaplan (2012) have shown that off-line parsability is not necessary to guarantee that generation from acyclic structures

is decidable, and the grammars in this proof demonstrate that it is not sufficient for cyclic structures.

Off-line parsability typically bounds the size of the c-structures of a string by a function of the length of that string. This works for parsing because the size of the f-structure is bounded by the size of the c-structure, but it is insufficient for generation because it does not constrain the structural correspondence between the c- and f-structure (see also Dymetman 1991). A single constraint that guarantees decidability for both parsing and generation must not only bound the size of the f-structures for a terminal string by the length of the string, but it must also ensure, as we have learned from the proof herein, that the determination of the terminal strings for an f-structure can be achieved with finite control.

Acknowledgments

The author wishes to thank Ron Kaplan for his insightful comments and helpful suggestions during the preparation of this paper, and the four anonymous reviewers for their valuable feedback on an earlier draft.

References

Dymetman, Marc. 1991. Inherently reversible grammars, logic programming and computability. In *Proceedings of the ACL Workshop: Reversible Grammar in Natural Language Processing*, pages 20–30, Berkeley, CA.

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.

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

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

Shieber, Stuart M., Hans Uszkoreit, Fernando C. N. Pereira, Jane Robinson, and Mabry Tyson. 1983. The formalism and implementation of PATR-II. In Barbara J. Grosz and Mark E. Stickel, editors, *Research on Interactive Acquisition and Use of Knowledge*. SRI Final Report 1894. SRI International, Menlo Park, CA, pages 39–79.

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