Without Remnant Movement, MGs Are Context-Free

Gregory M. Kobele

University of Chicago
kobele@uchicago.edu

Abstract. Minimalist grammars offer a formal perspective on a popular linguistic theory, and are comparable in weak generative capacity to other mildly context sensitive formalism. Minimalist grammars allow for the straightforward definition of so-called remnant movement constructions, which have found use in many linguistic analyses. It has been conjectured that the ability to generate this kind of configuration is crucial to the super-context-free expressivity of minimalist grammars. This conjecture is here proven.

In the minimalist program of [2], the well-formedness conditions on movement-type dependencies of the previous GB Theory [1] are reimplemented derivationally, so as to render ill-formed movement chains impossible to assemble. For example, the c-command restriction on adjacent chain links is enforced by making movement always to the root of the current subtree–a position c-commanding any other. One advantage of this derivational reformulation of chain well-formedness conditions is that so-called 'remnant movement' configurations, as depicted on the left in figure 1, are easy to generate. Remnant movement occurs when, due to previous movement operations, a moving expression does not itself have a grammatical description. Here we imagine that the objects derivable by the grammar in figure 1 include the black triangle and the complex of white and black triangles, but *not* the white triangle to the exclusion of the black triangle. From an incremental bottom-up perspective, the structure on the left in figure 1 first involves moving the grammatically specifiable black triangle, but then the non-directly grammatically describable white triangle moves. This is to be contrasted with the superficially similar configuration on the right in figure 1, in which, again from an incremental bottom-up perspective, both

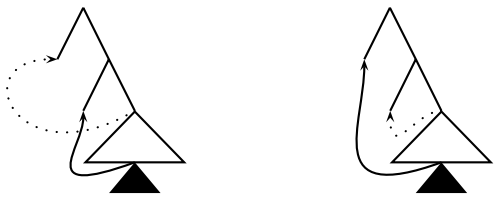

Fig. 1. Remnant Movement (left) vs Non-Remnant Movement (right)

C. Ebert, G. Jäger, and J. Michaelis (Eds.): MOL 10/11, LNCS 6149, pp. 160–173, 2010.

movement steps are of grammatically specifiable objects (the first step (here, the dotted line) involves movement of the complex of white and black triangles, and the second step (the solid line) involves movement of the black triangle). In particular, the dependencies generated by remnant movement are 'crossing', while those of the permissible type are nested (in the intuitive sense made evident in the figure).

The formalism of Minimalist Grammars (MGs) [20] was shown in [15] to be mildly context-sensitive (see also [10]). The MGs constructed in the proof use massive remnant movement to derive the non-context-free patterns, inviting the question as to whether this is necessary. Here we show that it is. MGs without remnant movement derive all and only the context-free languages. This result holds even when the SMC (a canonical constraint on movement, see [7]) is relaxed in such a way as to render the set of well-formed derivation trees non-regular. In this case, the standard proof [15] that MGs are mildly context-sensitive no longer goes through.

1 Mathematical Preliminaries

We assume familiarity with basic concepts of formal language theory. We write 2^A for the power set of a set A, and, for $f : A \to B$ a partial function, $dom(f)$ denotes the subset of A on which f is defined. Given a set Σ, Σ^* denotes the set of all finite sequences of elements from Σ, including the empty sequence ϵ. Σ^+ is the set of all finite sequences over Σ of length greater than 0. For $u, v \in \Sigma^*$, $u^\frown v$ is their concatenation. Often we will simply indicate concatenation via juxtaposition. A ranked alphabet is a set Σ together with a function **rank** : $\Sigma \to \mathbb{N}$ assigning to each 'function symbol' in Σ a natural number indicating the arity of the function it denotes. If Σ is a ranked alphabet, we write Σ_i for the set $\{\sigma \in \Sigma : \mathbf{rank}(\sigma) = i\}$. If $\sigma \in \Sigma_i$, we write $\sigma^{(i)}$ to indicate this fact. Let Σ be a ranked alphabet, the set of terms over Σ is written T_Σ, and is defined to be the smallest set containing each $\sigma \in \Sigma_0$, and for each $\sigma \in \Sigma_n$, and $t_1, \ldots, t_n \in T_\Sigma$, the term $\sigma(t_1, \ldots, t_n)$. For X any set, and Σ a ranked alphabet, $\Sigma \cup X$ is also a ranked alphabet, where $(\Sigma \cup X)_0 = \Sigma_0 \cup X$, and $(\Sigma \cup X)_i = \Sigma_i$ for all $i > 0$. We write $T_\Sigma(X)$ for $T_{\Sigma \cup X}$. A unary context over Σ is $C \in T_\Sigma(\{x\})$, such that x occurs exactly once in C. Given a unary context C and term T, we write $C[t]$ to denote the result of substituting t in for x in C ($x[t] = t$, $\sigma(t_1, \ldots, t_n)[t] = \sigma(t_1[t], \ldots, t_n[t])$). A bottom-up tree automaton is given by a quadruple $\langle Q, \Sigma, \to, Q_F \rangle$, where Q is a finite set of states, $Q_f \subseteq Q$ is the set of final states, Σ is a ranked alphabet, and $\to \subset_{fin} \Sigma \times Q^* \to Q$. A bottom-up tree automaton defines a relation $\Rightarrow$: $T_\Sigma(Q) \times T_\Sigma(Q)$. If C is a unary context over $\Sigma \cup Q$, and $\langle \sigma^{(n)}, q_1, \ldots, q_n \rangle \to q$, then $C[\sigma(q_1, \ldots, q_n)] \Rightarrow C[q]$. The tree language accepted by a bottom-up tree automaton A is defined as $L(A) = \{t \in T_\Sigma : \exists q \in Q_F.\ t \Rightarrow^* q\}$. A set of trees is regular iff it is the language accepted by some bottom-up tree automaton.

2 Minimalist Grammars

We use the notation of [22]. An MG over an alphabet Σ is a triple $G = \langle Lex, \mathbf{sel}, \mathbf{lic}\rangle$ where **sel** and **lic** are finite non-empty sets (of 'selection' and 'licensing' feature types), and for $\mathbb{F} = \{\texttt{=s}, \texttt{s} : s \in \mathbf{sel}\} \cup \{\texttt{+l}, \texttt{-l} : l \in \mathbf{lic}\}$, $Lex \subset_{fin} \Sigma^* \times \mathbb{F}^*$. Given binary function symbols $\Delta_2 := \{\mathbf{mrg1}, \mathbf{mrg2}, \mathbf{mrg3}\}$ and unary $\Delta_1 := \{\mathbf{mv1}, \mathbf{mv2}\}$, a derivation is a term in $der(G) = T_{\Delta_2 \cup \Delta_1 \cup Lex}$, where elements of Lex are treated as nullary symbols. An expression is a finite sequence $\phi_0, \phi_1, \ldots, \phi_n$ of pairs over $\Sigma^* \times \mathbb{F}^*$; the first component ϕ_0 represents the yield and features of the expression (qua tree) minus any moving parts, and the remaining components represent the yield and features of the moving parts of the expression. Thus an expression of the form $\phi_0 = \langle \sigma, \gamma\rangle$ represents a tree with no moving pieces; such an expression is called a *complete* expression of category γ. $Eval : der(G) \to 2^{(\Sigma^* \times \mathbb{F}^*)^+}$ is a partial function mapping derivations to the sets of expressions they are derivations of. Given $\ell \in Lex$, $Eval(\ell) = \{\ell\}$, and $Eval(\mathbf{mrg}_i(d_1, d_2))$ and $Eval(\mathbf{mv}_i(d))$ are defined as $\{merge_i(e_1, e_2) : e_j \in Eval(d_j)\}$ and $\{move_i(e) : e \in Eval(d)\}$ respectively, where the operations $merge_i$ and $move_i$ are defined below. In the following, $\sigma, \tau \in \Sigma^*$, $\gamma, \delta \in \mathbb{F}^*$, and $\phi_i, \psi_j \in \Sigma^* \times \mathbb{F}^*$.

$$\frac{\langle \sigma, \texttt{=c}\gamma\rangle \in Lex \qquad \langle \tau, \texttt{c}\rangle, \psi_1, \ldots, \psi_n}{\langle \sigma^\frown \tau, \gamma\rangle, \psi_1, \ldots, \psi_n}\ merge_1$$

$$\frac{\langle \sigma, \texttt{=c}\gamma\rangle, \phi_1, \ldots, \phi_m \qquad \langle \tau, \texttt{c}\rangle, \psi_1, \ldots, \psi_n}{\langle \tau^\frown \sigma, \gamma\rangle, \phi_1, \ldots, \phi_m, \psi_1, \ldots, \psi_n}\ merge_2$$

$$\frac{\langle \sigma, \texttt{=c}\gamma\rangle, \phi_1, \ldots, \phi_m \qquad \langle \tau, \texttt{c}\delta\rangle, \psi_1, \ldots, \psi_n}{\langle \sigma, \gamma\rangle, \phi_1, \ldots, \phi_m, \langle \tau, \delta\rangle, \psi_1, \ldots, \psi_n}\ merge_3$$

$$\frac{\langle \sigma, \texttt{+c}\gamma\rangle, \phi_1, \ldots, \phi_{i-1}, \langle \tau, \texttt{-c}\rangle, \phi_{i+1}, \ldots, \phi_m}{\langle \tau^\frown \sigma, \gamma\rangle, \phi_1, \ldots, \phi_{i-1}, \phi_{i+1}, \ldots, \phi_m}\ move_1$$

$$\frac{\langle \sigma, \texttt{+c}\gamma\rangle, \phi_1, \ldots, \phi_{i-1}, \langle \tau, \texttt{-c}\delta\rangle, \phi_{i+1}, \ldots, \phi_m}{\langle \sigma, \gamma\rangle, \phi_1, \ldots, \phi_{i-1}, \langle \tau, \delta\rangle, \phi_{i+1}, \ldots, \phi_m}\ move_2$$

The SMC is a restriction on the domains of $move_1$ and $move_2$ which render these relations functional.

$$\text{no } \phi_j = \langle \sigma_j, \gamma_j\rangle \text{ is such that } \gamma_j = \texttt{-c}\gamma'_j \text{ unless } j = i \qquad (\mathbf{SMC})$$

The (string) language generated at a category $\texttt{c}$ (for $c \in \mathbf{sel}$) by a MG G is defined to be the yields of the complete expressions of category $\texttt{c}$:[1] $L_{\texttt{c}}(G) := \{\sigma : \exists d \in der(G).\ \langle \sigma, \texttt{c}\rangle \in Eval(d)\}$.

[1] Implicit in [15] is the fact that for any $\texttt{c}$, $dom_{\texttt{c}}(Eval) = \{d : \exists \sigma.\ \langle \sigma, \texttt{c}\rangle \in Eval(d)\}$ is a regular tree language. This is explicitly shown in [13].

3 A Ban on Remnant Movement

In order to implement a ban on remnant movement, we want to implement a *temporaray* island status on moving expressions: nothing can move out of a moving expression until it has settled down ('please wait until the train has come to a complete stop before exiting'). Currently, an expression $e = \phi_0, \phi_1, \ldots, \phi_k$ has the form just given, where ϕ_0 is the 'head' of the expression, and the other ϕ_i are 'moving parts'. Importantly, although we view such an expression as a compressed representation of a tree, there is no hierarchical relation among the ϕ_i. In order to implement a ban against remnant movement, we need to indicate which of the moving parts are contained in which others. We represent this information by retaining some of the relative dominance relations in the represented tree: $e = \phi_0, T_1, \ldots, T_n$, where each tree T_i pairs a moving part with a (possibly empty) sequence of trees (the set of trees T is the smallest set X such that $X = (\Sigma^* \times \mathbb{F}^*) \times X^*$). We interpret such a structure as a moving part (the features of which are represented by ϕ_i) which itself may contain moving subparts $(T_1^i, \ldots, T_m^i)$. By allowing these moving subparts to become accessible for movement only after the features of ϕ_i have been exhausted, we rule out the crossing, remnant movement type dependencies. The revised cases of the operations *merge* and *move*, *PBC-merge* and *PBC-move*,[2] are given below. The function *PBC-Eval* interprets derivations $d \in der(G)$ in 'PBC-mode,' such that $PBC\text{-}Eval(\ell) = \{\ell\}$, $PBC\text{-}Eval(\mathbf{mv}_i(d)) = \{PBC\text{-}move_i(e) : e \in PBC\text{-}Eval(d)\}$, and $PBC\text{-}Eval(\mathbf{mrg}_i(d_1, d_2)) = \{PBC\text{-}merge_i(e_1, e_2) : e_j \in PBC\text{-}Eval(d_j)\}$.

In the below, σ, τ are strings, γ, δ are finite sequences of syntactic features, S_i, T_j are trees of the form $\langle\langle\sigma, \gamma\rangle, \langle S_1, \ldots, S_n\rangle\rangle$.

$$\frac{\langle \sigma, \texttt{=c}\gamma\rangle \in Lex \qquad \langle \tau, \texttt{c}\rangle, T_1, \ldots, T_n}{\langle \sigma^\frown\tau, \gamma\rangle, T_1, \ldots, T_n}\ PBC\text{-}merge_1$$

$$\frac{\langle \sigma, \texttt{=c}\gamma\rangle, S_1, \ldots, S_m \qquad \langle \tau, \texttt{c}\rangle, T_1, \ldots, T_n}{\langle \tau^\frown\sigma, \gamma\rangle, S_1, \ldots, S_m, T_1, \ldots, T_n}\ PBC\text{-}merge_2$$

$$\frac{\langle \sigma, \texttt{=c}\gamma\rangle, S_1, \ldots, S_m \qquad \langle \tau, \texttt{c}\delta\rangle, T_1, \ldots, T_n}{\langle \sigma, \gamma\rangle, S_1, \ldots, S_m, \langle\langle \tau, \delta\rangle, \langle T_1, \ldots, T_n\rangle\rangle}\ PBC\text{-}merge_3$$

$$\frac{\langle \sigma, \texttt{+c}\gamma\rangle, S_1, \ldots, S_{i-1}, \langle\langle \tau, \texttt{-c}\rangle, \langle T_1, \ldots, T_n\rangle\rangle, S_{i+1}, \ldots, S_m}{\langle \tau^\frown\sigma, \gamma\rangle, S_1, \ldots, S_{i-1}, T_1, \ldots, T_n, S_{i+1}, \ldots, S_m}\ PBC\text{-}move_1$$

$$\frac{\langle \sigma, \texttt{+c}\gamma\rangle, S_1, \ldots, S_{i-1}, \langle\langle \tau, \texttt{-c}\delta\rangle, \langle T_1, \ldots, T_n\rangle\rangle, S_{i+1}, \ldots, S_m}{\langle \sigma, \gamma\rangle, S_1, \ldots, S_{i-1}, \langle\langle \tau, \delta\rangle, \langle T_1, \ldots, T_n\rangle\rangle, S_{i+1}, \ldots, S_m}\ PBC\text{-}move_2$$

[2] The 'PBC' is named after the proper binding condition of [6], which filters out surface structures in which a trace linearly precedes its antecedent. If the antecedent of a trace left behind by a particular movement step is defined to be the element (trace or otherwise) in the target position of that movement, the present modification to the rules *merge* and *move* exactly implement the PBC in the minimalist grammar framework.

We will continue to require that these rules satisfy (a version of) the SMC.[3] Following [12], we define the SMC over *PBC-move* as follows:

$$\text{no } T_j = \langle\langle \sigma_j, \gamma_j\rangle, \langle T_1^j, \ldots, T_n^j\rangle\rangle \text{ is such that } \gamma_j = \texttt{-c}\gamma_j' \text{ unless } j = i \qquad \textbf{(PBC-SMC)}$$

The string language generated in PBC-mode at a category c is defined as usual: $L_{\texttt{c}}^{PBC}(G) := \{\sigma : \exists d \in der(G).\langle \sigma, \texttt{c}\rangle \in PBC\text{-}Eval(d)\}$.

Observe that the rule $PBC\text{-}merge_3$ introduces new tree structure, temporarily freezing the moving pieces within its second argument. The rules $PBC\text{-}move_1$ and $PBC\text{-}move_2$ enforce that only the root of a tree is accessible to movement operations, and that its daughter subtrees become accessible to movement only once the root has finished moving.

Note also that the set of well-formed derivation trees in PBC-mode (the set $dom(PBC\text{-}Eval)$) is not a regular tree language (this is due to the laxness of the PBC-SMC). To see this, consider the MG $G_1 = \langle Lex, \{x, y\}, \{A\}\rangle$, where Lex contains the four lexical items below.

a::=x x -A f::x
c::=y +A y e::=x y

Derivations of complete expressions of category y begin by starting with f, and repeatedly merging tokens of a. Then e is merged, and for each a, a c is merged, and a move step occurs. In particular, although the yields of these trees form the context-free language $\texttt{c}^n\texttt{ea}^n\texttt{f}$, the number of $\mathbf{mrg}_3$ nodes must be equal to the number of $\mathbf{mv}_1$ nodes. It is straightforward to show that no finite-state tree automaton can enforce this invariant.

Our main result is that minimalist grammars under the PBC mode of derivation (i.e. using the rules just given above) generate exactly the class of context-free languages.

4 MGs with Hypotheses

Because the elimination of remnant movement guarantees that, viewed from a bottom-up perspective, we will finish moving a containing expression before we need to deal with any of its subparts, we can re-represent expressions using 'slash-features', as familiar from GPSG [8]. Accordingly, we replace $(PBC\text{-})merge_3$,

[3] There are two natural interpretations of the SMC on expressions $e = \Phi, T_1, \ldots, T_n$. First, one might require that no two ϕ_i and ϕ_j, share the same first feature, regardless of how deeply embedded within trees they may be. This perspective views the tree structure as irrelevant for the statement of the SMC. Another reasonable option is to require only that no two ϕ_i and ϕ_j share the same first feature, where ϕ_i and ϕ_j are the roots of trees T_i and T_j respectively. This perspective views the tree structure of moving parts as relevant to the SMC, and allows for a kind of 'smuggling' [3], as described in [12]. The results of this paper are independent of which of these two interpretations of the SMC we adopt. We adopt the second, because it is more interesting (the derivation tree sets no longer constitute regular tree languages).

which introduces a to-be-moved expression, with a new (non-functional) operation, *assume*, which introduces a 'slash-feature', or hypothesis. A hypothesis takes the form of a pair of feature strings $\langle \delta, \gamma \rangle$. The interpretation of a hypothesis $\langle \delta, \gamma \rangle$, is such that δ records the originally postulated 'missing' feature sequence (and thus is unchanging over the lifetime of the hypothesis), whereas γ represents the remaining features of the hypothesis, which are checked off as the derivation progresses. $Move_1$, which re-integrates a moving part into the main expression, is replaced with another new operation, *discharge*. *Discharge* replaces 'used up' hypotheses with the expressions that they 'could have been.' These expressions may themselves contain hypothesized moving pieces. Derivations of minimalist grammars with hypothetical reasoning in this sense are terms $d \in$ *Hyp-der*(G) over a signature $\{\mathbf{mrg}_1^{(2)}, \mathbf{mrg}_2^{(2)}, \mathbf{assm}^{(1)}, \mathbf{mv}_2^{(1)}, \mathbf{dschrg}^{(2)}\} \cup Lex$, and *Hyp-Eval* partially maps such terms to expressions in the by now familiar manner.

In the below, σ, τ are strings over Σ, γ, δ, ζ are finite sequences of syntactic features, ϕ_i, ψ_j are pairs of the form $\langle \delta, \gamma \rangle$.

$$\frac{\langle \sigma, \texttt{=c}\gamma \rangle \in Lex \qquad \langle \tau, \texttt{c} \rangle, \psi_1, \ldots, \psi_n}{\langle \sigma^\frown \tau, \gamma \rangle, \psi_1, \ldots, \psi_n} \; merge_1$$

$$\frac{\langle \sigma, \texttt{=c}\gamma \rangle, \phi_1, \ldots, \phi_m \qquad \langle \tau, \texttt{c} \rangle, \psi_1, \ldots, \psi_n}{\langle \tau^\frown \sigma, \gamma \rangle, \phi_1, \ldots, \phi_m, \psi_1, \ldots, \psi_n} \; merge_2$$

$$\frac{\langle \sigma, \texttt{=c}\gamma \rangle, \phi_1, \ldots, \phi_m}{\langle \sigma, \gamma \rangle, \phi_1, \ldots, \phi_m, \langle \texttt{c}\delta, \delta \rangle} \; assume$$

$$\frac{\langle \sigma, \texttt{+c}\gamma \rangle, \phi_1, \ldots, \phi_{i-1}, \langle \delta, \texttt{-c} \rangle, \phi_{i+1}, \ldots, \phi_m \qquad \langle \tau, \delta \rangle, \psi_1, \ldots, \psi_n}{\langle \tau^\frown \sigma, \gamma \rangle, \phi_1, \ldots, \phi_{i-1}, \psi_1, \ldots, \psi_n, \phi_{i+1}, \ldots, \phi_m} \; discharge$$

$$\frac{\langle \sigma, \texttt{+c}\gamma \rangle, \phi_1, \ldots, \phi_{i-1}, \langle \zeta, \texttt{-c}\delta \rangle, \phi_{i+1}, \ldots, \phi_m}{\langle \sigma, \gamma \rangle, \phi_1, \ldots, \phi_{i-1}, \langle \zeta, \delta \rangle, \phi_{i+1}, \ldots, \phi_m} \; move_2$$

We subject the operations $move_2$ and *discharge* to a version of the SMC:

$$\text{no } \phi_j = \langle \zeta_j, \gamma_j \rangle \text{ is such that } \gamma_j = \texttt{-c}\gamma'_j \text{ unless } j = i \qquad \textbf{(Hyp-SMC)}$$

The language of a minimalist grammar G at category c using hypothetical reasoning is defined to be:

$$L_{\texttt{c}}^{Hyp}(G) := \{\sigma : \exists d \in \textit{Hyp-der}(G).\ \langle \sigma, \texttt{c} \rangle \in \textit{Hyp-Eval}(d)\}$$

The operation *discharge* constrains the kinds of assumptions introduced by *assume* which can be part of a well-formed derivation to be those which are of the form $\langle \texttt{c}\delta, \delta \rangle$, where there is some lexical item $\langle \sigma, \gamma\texttt{c}\delta \rangle$. As there are finitely many lexical items, there are thus only finitely many useful assumptions given a particular lexicon. It will be implicitly assumed in the remainder of this paper that *assume* is restricted so as to generate only useful assumptions. We henceforth index **assm** nodes with the features of the hypotheses introduced (writing thus $\mathbf{assm}_{\texttt{c}\gamma}$ for an *assume* operation introducing the hypothesis $\langle \texttt{c}\gamma, \gamma \rangle$).

Theorem 1. *For any G, and any $c \in \boldsymbol{sel}_G$, the set $dom_{\mathsf{c}}(Hyp\text{-}Eval) = \{d : \exists\sigma.\ \langle\sigma, \mathsf{c}\rangle \in Hyp\text{-}Eval(d)\}$ is a regular tree language.*

Proof. Construct a nondeterministic bottom-up tree automaton whose states are $(|\mathbf{lic}| + 1)$-tuples of pairs of suffixes of lexical feature sequences. The Hyp-SMC allows us to devote each component of such a sequence beyond the first to the (if it exists, unique) hypothesis beginning with a particular `-c` feature, and thus we assume to be given a fixed enumeration of **lic**. The remarks above guarantee that there are only a finite number of such states needed. Given an expression, $\phi_0, \phi_1, \ldots, \phi_n$, the state representing it has as its i^{th} component the pair $\langle\epsilon, \epsilon\rangle$ if there is no ϕ_j beginning with the i^{th} `-c` feature, and the unique ϕ_j beginning with the i^{th} `-c` feature otherwise. The 0^{th} component of a state is always of the form $\langle\epsilon, \gamma\rangle$, where γ is the feature sequence of ϕ_0. As we are interested in derivations of complete expressions of category `c`, the final state is $\langle\langle\epsilon, \mathsf{c}\rangle, \langle\epsilon, \epsilon\rangle, \ldots, \langle\epsilon, \epsilon\rangle\rangle$. The transitions of the automaton are defined so as to preserve this invariant: at a lexical item $\ell = \langle\sigma, \gamma\rangle$, the automaton enters the state $\langle\langle\epsilon, \gamma\rangle, \langle\epsilon, \epsilon\rangle, \ldots, \langle\epsilon, \epsilon\rangle\rangle$, and at an internal node $\sigma^{(n)}(q_1, \ldots, q_n)$, the automaton enters the state q just in case there are expressions $e_1, \ldots, e_n$ represented by states $q_1, \ldots, q_n$ which are mapped by the operation denoted by σ to an expression e represented by state q.

We use the facts that linear homomorphisms preserve recognizability and that the yield of a recognizable set of trees is context-free [4] in conjunction with theorem 1 to show that minimalist grammars using hypothetical reasoning define exactly the context-free languages.

Theorem 2. *For any G, and any $c \in \boldsymbol{sel}_G$, $L^{Hyp}_{\mathsf{c}}(G)$ is context-free.*

Proof. Let G and c be given. By theorem 1, $D = dom_{\mathsf{c}}(Hyp\text{-}Eval)$ is recognizable. Let $E = f[D]$, where f is the homomorphism defined as follows (f maps nullary symbols to themselves):

$$f(\sigma(e_1, \ldots, e_n)) = \begin{cases} \sigma(f(e_2), f(e_1)) & \text{if } \sigma \in \{\mathbf{mrg}_2, \mathbf{dschrg}\} \\ \sigma(f(e_1), \ldots, f(e_n)) & \text{otherwise} \end{cases}$$

Inspection of f reveals that it is merely putting sister subtrees in the order in which they are pronounced (à la *Hyp-Eval*) and thus, for any $d \in D$, $Hyp\text{-}Eval(d)$ contains $\langle\sigma, \mathsf{c}\rangle$ iff $yield(f(d)) = \sigma$. As f is linear, E is recognizable, and thus $yield(E) = L^{Hyp}_{\mathsf{c}}(G)$ is context-free.

5 Relating the PBC to Hypothetical Reasoning

To show that minimalist grammars in PBC mode are equivalent to minimalist grammars with hypothetical reasoning we will exhibit an Eval-preserving bijection between complete derivation trees of both formalisms.[4] The gist of the

[4] A complete derivation tree is just one which is the derivation of a complete expression. I will in the following use the term in conjunction with derivations in $der(G)$ to refer exclusively to expressions derived in PBC-mode.

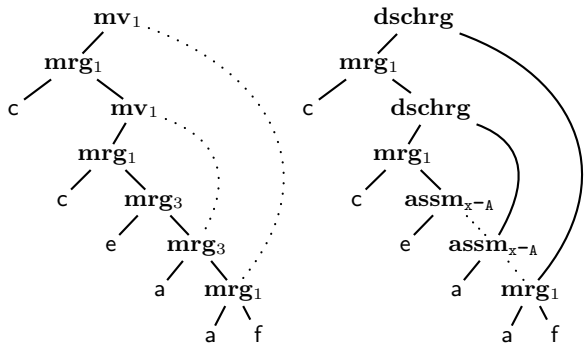

Fig. 2. Derivations in G_1 of *afcace*

transformation is best provided via an example. Consider the trees in figure 2, which are derivations over the MG G_1 in PBC mode and using hypothetical reasoning respectively of the string *afcace*.

The dotted lines in the derivation trees in figure 2 indicate the implicit dependencies between the unary operations and other expressions in the derivation. For example, **mv** nodes are connected via a dotted line to the subtree which 'moves'. Similarly, $\mathbf{assm}_\gamma$ nodes are connected via a dotted line to the expression which ultimately discharges the assumption they introduced. The subtree connected via dotted line to a $\mathbf{mv}_1$ node I will call the subtree targeted by that *move* operation, and is that subtree whose leftmost leaf introduces the `-c` feature checked by this *move* step, and which is the right child of a $\mathbf{mrg}_3$ node. Note that if the derivation is well-formed (i.e. is in the domain of *PBC-Eval*) there is a unique subtree targeted by every $\mathbf{mv}_1$ node. The right daughter of a **dschrg** node connected via dotted line to a $\mathbf{assm}_\gamma$ node is called the hypothesis discharged by that *discharge* operation, and is connected to the $\mathbf{assm}_\gamma$ node which introduces the hypothesis which is discharged at its parent node. Again, if the derivation is well-formed, the $\mathbf{assm}_\gamma$ node in question is the unique such. Note, however, that it is only in the case of complete derivation trees that to every $\mathbf{assm}_\gamma$ node there corresponds the hypothesis-discharging **discharge** node. The major difference between PBC and Hyp MG derivations is that expressions entering into multiple feature checking relationships during the course of the derivation are introduced into the derivation at the point their first feature checking relationship takes place in the case of PBC (and MG derivations more generally), and at the point their *last* feature checking relationship obtains in the case of Hyp MGs.

The relation between the two trees in figure 2, and more generally between PBC derivations and hypothetical derivations, is that the subtree connected via dotted line to $\mathbf{mv}_1$ becomes the second argument of **dschrg**, and the second argument of $\mathbf{mrg}_3$ becomes the subtree connected to **assm** via a dotted line. This is shown in figure 3.

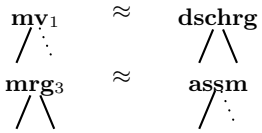

Fig. 3. Relating PBC derivations and hypothetical derivations

We define a relation $\textsc{Trans} \subset der(G) \times Hyp\text{-}der(G)$ in the following manner:

1. $\textsc{Trans}(\mathbf{mv}_1(d), \mathbf{dschrg}(d_1, d_2))$, where, for d' is the (unique) subtree targeted by this instance of $\mathbf{mv}_1$, $\textsc{trans}(d, d_1)$ and $\textsc{Trans}(d', d_2)$
2. $\textsc{Trans}(\mathbf{mrg}_3(d_1, d_2), \mathbf{assm}_\gamma(d))$, where $PBC\text{-}Eval(d_2) = \{\phi_0, \phi_1, \ldots, \phi_n\}$, $\phi_0 = \langle \sigma, \gamma \rangle$, and $\textsc{Trans}(d_1, d)$
3. $\textsc{Trans}(\sigma(d_1, \ldots, d_n), \sigma(d'_1, \ldots, d'_n))$, where $\textsc{Trans}(d_i, d'_i)$, for all $1 \leq i \leq n$

By inspection of the above case-wise definition, it is easy to see that

Theorem 3. Trans *is a function.*

The point of defining Trans as per the above is to use it to show that the structural 'equivalence' sketched in figure 3 preserves relevant aspects of weak generative capacity. Expressions denoted by derivations in both 'formalisms' have been represented here as sequences. However, only the type of the first element of such sequences (a pair $\langle \sigma, \gamma \rangle \in \Sigma^* \times \mathbb{F}^*$) is identical across formalisms (the other elements are trees with nodes pairs of the same type in the PBC MGs, but are pairs of feature sequences in Hyp MGs). Accordingly, the relation I will show Trans to preserve is the identity of the first element of the yield of the source and target derivation trees.

Theorem 4. *For $d \in der(G)$, such that $\{\phi_0, T_1, \ldots, T_n\} = PBC\text{-}eval(d)$, if $\psi_0, \psi_1, \ldots, \psi_k \in Hyp\text{-}Eval(\textsc{Trans}(d))$, then $\phi_0 = \psi_0$, $n = k$, and for $1 \leq i \leq n$, $T_i = \langle \langle \sigma_i, \gamma_i \rangle, T^i_1, \ldots, T^i_m \rangle$ and $\psi_i = \langle \zeta_i, \gamma_i \rangle$.*

Proof. By induction. For the base case, let d be a lexical item. $PBC\text{-}Eval(d)$ and $Hyp\text{-}Eval(d)$ are both equal to $\{d\}$, which by case 3 of the definition of Trans, is equal to $Hyp\text{-}Eval(\textsc{Trans}(d))$. Now let d_1 and d_2 be appropriately related to $\textsc{Trans}(d_1)$ and $\textsc{Trans}(d_1)$ respectively. There are five cases to consider ($\mathbf{mrg_i}, \mathbf{mv_j}$, for $1 \leq i \leq 3$ and $1 \leq j \leq 2$).

1. Let $d = \mathbf{mrg_1}(d_1, d_2)$. Then by case 3 of the definition of Trans, $\textsc{Trans}(d) = \mathbf{mrg_1}(\textsc{Trans}(d_1), \textsc{Trans}(d_2))$. $PBC\text{-}Eval(d)$ is defined if and only if both $PBC\text{-}Eval(d_1) = \{\langle \sigma, \texttt{=c}\gamma \rangle\}$ and $PBC\text{-}Eval(d_1) = \{\langle \tau, \texttt{c} \rangle, T_1, \ldots, T_n\}$, in which case it is $\{\langle \sigma^\frown \tau, \gamma \rangle, T_1, \ldots, T_n\}$. By the induction hypothesis, we conclude that $Hyp\text{-}Eval(\textsc{Trans}(d_1)) = \{\langle \sigma, \texttt{=c}\gamma \rangle\}$ and $Hyp\text{-}Eval(\textsc{Trans}(d_2)) = \{\langle \tau, \texttt{c} \rangle, \psi_1, \ldots, \psi_n\}$, which are in the domain of $merge_1$, and thus, by inspection of the definition of this latter, that d and $\textsc{Trans}(d)$ are appropriately related as well.

2. The case where $d = \mathbf{mrg_2}(d_1, d_2)$ is not interestingly different from the above.
3. Let $d = \mathbf{mrg_3}(d_1, d_2)$. Then by case 2 of the definition of TRANS, TRANS(d) = $\mathbf{assm}_\gamma$(TRANS(d_1)). As d_1 and TRANS(d_1) are appropriately related (by the induction hypothesis), and as both $merge_3$ and $assume_\gamma$ define the first component of their result to be the same as the first component of their leftmost argument minus the first feature, d and TRANS(d) are appropriately related as well.
4. Let $d = \mathbf{mv_1}(d_1)$, and let d_2 be the unique subtree targeted by this instance of $\mathbf{mv_1}$. For *PBC-Eval*(d) to be defined, *PBC-Eval*(d_1) must be equal to $\{\langle \sigma, \texttt{+c}\gamma\rangle, S_1, \ldots, S_{i-1}, \langle\langle \tau, \texttt{-c}\rangle, \langle T_1, \ldots, T_n\rangle\rangle, S_{i+1}, \ldots, S_m\}$. In this case, *PBC-Eval*(d_2) = $\{\langle \tau, \delta\texttt{-c}\rangle, T_1, \ldots, T_n\}$. By the induction hypothesis, *Hyp-Eval*(TRANS(d_1)) = $\{\langle \sigma, \texttt{+c}\gamma\rangle, \phi_1, \ldots, \langle \delta, \texttt{-c}\rangle, \phi_{i+1}, \ldots, \phi_m\}$, and in addition *Hyp-Eval*(TRANS(d_2)) = $\{\langle \tau, \delta\texttt{-c}\rangle, \psi_1, \ldots, \psi_n\}$. Thus, we can see that the *discharge* operation is defined on these arguments, and is equal to $\{\langle \tau^\frown\sigma, \gamma\rangle, \phi_1, \ldots, \phi_{i-1}, \psi_1, \ldots, \psi_n, \phi_{i+1}, \ldots, \phi_m\}$. Applying the operation $move_1$ to *PBC-Eval*(d_1) we obtain the unit set consisting of the single element $\langle \tau^\frown\sigma, \gamma\rangle, S_1, \ldots, S_{i-1}, T_1, \ldots, T_n, S_{i+1}, \ldots, S_m$, and thus establish that d is appropriately related to TRANS(d).
5. Finally, let $d = \mathbf{mv_2}(d_1)$. By case 3, TRANS(d) = $\mathbf{mv_2}$(TRANS(d_1)). If *PBC-Eval*(d) is defined, then there is a unique moving component T_i of *PBC-Eval*(d_1) which an appropriate first feature. By the induction hypothesis, there is a unique corresponding ψ_i in *Hyp-Eval*(TRANS(d_1)), allowing $move_2$(*Hyp-Eval*(TRANS(d_1))) to be defined, and us to see that d and TRANS(d) are appropriately related in this case too.

Note that whenever $d \in der(G)$ is complete, so too is TRANS(d) $\in$ *Hyp-der*(G).

Corollary 1. TRANS *preserves completeness.*

Furthermore, by inspecting the cases of the proof above, we see that the hypothesis introduced by a particular $\mathbf{assm}_\gamma$ node which is the translation of a $\mathbf{mrg}_3$ node, is discharged at the **dschrg** node which is the translation of the $\mathbf{mv}_1$ node which targets the right daughter of that $\mathbf{mrg}_3$ node.

From theorem 4 follows the following

Corollary 2. *For every G, and any feature sequence γ, $L^{PBC}_\gamma(G) \subseteq L^{Hyp}_\gamma(G)$.*

To prove the inclusion in the reverse direction, I will show that for every complete $d' \in$ *Hyp-der*(G) there is a $d \in der(G)$ such that TRANS(d) = d'. I define a function SNART which takes a pair consisting of a derivation tree $d' \in$ *Hyp-der*(G) and a set M of pairs of strings over $\{0,1\}^*$ and derivation trees in $der(G)$. We interpret a pair $\langle p, d\rangle \in M$ as stating that we are to insert tree d as a daughter of the node at address p. (Recall that in translating from Hyp MG derivation trees to PBC trees we need to 'lower' expressions introduced at a **dschrg** node into the position in which they were assumed.) Given a set of such pairs M, I denote by ${}_{(i)}M$ (for $i \in \{0,1\}$) the set $\{\langle p, d\rangle : \langle ip, d\rangle \in M\}$. I will use this notation to keep track of where the trees in M should be inserted into the translated structure. Basically, when an item $\langle \epsilon, d\rangle \in M$, it indicates that it should be inserted

as a daughter of the current root. I will use the notation $M(\epsilon)$ to denote the unique d such that $\langle \epsilon, d\rangle \in M$, if one exists.

1. for $d = \mathbf{dschrg}(d_1, d_2)$, and p the address in d of the $\mathbf{assm}_\gamma$ node whose hypothesis is discharged at this **dschrg** node, we define $\textsc{snart}(d, M) = \mathbf{mv_1}(\textsc{snart}(d_1, {}_{(0)}(M \cup \{\langle p, \textsc{snart}(d_2, {}_{(1)}M)\rangle\})))$
2. for $d = \mathbf{assm}_\gamma(d_1)$, $\textsc{snart}(d, M) = \mathbf{mrg_3}(\textsc{snart}(d_1, {}_{(0)}M), M(\epsilon))$
3. $\textsc{snart}(\sigma(d_1, \ldots, d_n), M) = \sigma(\textsc{snart}(d_1, {}_{(0)}M), \ldots, \textsc{snart}(d_n, {}_{(n-1)}M))$

Note that, although SNART is not defined on all trees in *Hyp-der*(G) (case 2 is undefined whenever there is no (unique) $\langle \epsilon, d\rangle \in M$), it *is* defined on all complete $d \in$ *Hyp-der*(G).

Theorem 5. *For all complete* $d \in$ *Hyp-der*(G), $\textsc{snart}(d, \emptyset) \in$ *der*(G).

Proof. Case 2 is the only potential problem (as is undefined whenever $M(\epsilon)$ is). However, in a complete derivation tree, every $\mathbf{assm}_\gamma$ node is dominated by a **dschrg** node, at which is discharged the hypothesis introduced by this former. Moreover, no **dschrg** node discharges the hypothesis of more than one $\mathbf{assm}_\gamma$ node. Thus, we are guaranteed in a complete derivation tree that at each occurrence of an $\mathbf{assm}_\gamma$ node $M(\epsilon)$ is defined.

That the range of SNART is contained in *der*(G) is verified by simple inspection of its definition.

Of course, we want not just that SNART map derivations in *Hyp-der*(G) to ones in *der*(G), but also that a derivation d in *der*(G) to which a complete derivation d' in *Hyp-der*(G) is mapped by SNART maps back to d' via TRANS. This will allow us to conclude the converse of corollary 2.

Theorem 6. *For all complete* $d \in$ *Hyp-der*(G), $d = \textsc{Trans}(\textsc{snart}(d, \emptyset))$.

Proof. In order to have a strong enough inductive hypothesis, we need to prove something stronger than what is stated in the theorem. Let $d \in$ *Hyp-der*(G), and M be a partial function with domain $\{0,1\}^*$ and range *der*(G), such that p is the address of an $\mathbf{assm}_\gamma$ node in d without a corresponding **dschrg** node iff there is some d' such that $M(p) = d'$. (In plain English, M tells us how to translate 'unbound' $\mathbf{assm}_\gamma$ nodes in d.) Then $d = \textsc{Trans}(\textsc{snart}(d, M))$. Note that the statement of the theorem is a special case, as for d complete there are no unbound $\mathbf{assm}_\gamma$ nodes, and thus M can be $\emptyset$.

For the base case, let d be a lexical item (and thus complete). Then by case 3 of the definition of SNART, $\textsc{snart}(d, \emptyset) = d$, and by case 3 of the definition of TRANS, $\textsc{Trans}(d) = \textsc{Trans}(\textsc{snart}(d)) = d$. Now let d_1, d_2, be as per the above such that for appropriate M_1, M_2, $\textsc{Trans}(\textsc{snart}(d_1, M_1)) = d_1$ and $\textsc{Trans}(\textsc{snart}(d_2, M_2)) = d_2$. There are again five cases to consider.

1. Let $d = \mathbf{mrg}_1(d_1, d_2)$, and M an assignment of trees in *der*(G) to unbound $\mathbf{assm}_\gamma$ nodes in d. Then $\textsc{Trans}(\textsc{snart}(d, M))$ is, by case 3 of the definition of SNART, $\textsc{Trans}(\mathbf{mrg}_1(\textsc{snart}(d_1, {}_{(0)}M), \textsc{snart}(d_2, {}_{(1)}M)))$. By

case 3 of the definition of TRANS, this is easily seen to be identical to $\mathbf{mrg}_1(\text{TRANS}(\text{SNART}(d_1, {}_{(0)}M)), \text{TRANS}(\text{SNART}(d_2, {}_{(1)}M)))$. As, for any i, ${}_{(i)}M$ is an assignment of trees to unbound $\mathbf{assm}_\gamma$ nodes in d_i, the inductive hypothesis applies, and thus $\text{TRANS}(\text{SNART}(d, M)) = d$, as desired.

2. The case where $d = \mathbf{mrg_2}(d_1, d_2)$ is not interestingly different from the above.
3. Let $d = \mathbf{assm}_\gamma(d_1)$, and let M assign trees to its unbound $\mathbf{assm}_\gamma$ nodes (in particular, $M(\epsilon)$ is defined). Then by case 2 of the definition of SNART, $\text{TRANS}(\text{SNART}(d, M)) = \text{TRANS}(\mathbf{mrg}_3(\text{SNART}(d_1, {}_{(0)}M), M(\epsilon)))$. Now, according to case 2 of the definition of TRANS, this is seen to be identical to $\mathbf{assm}_\gamma(\text{TRANS}(\text{SNART}(d_1, {}_{(0)}M)))$, which according to our inductive hypothesis is simply $\mathbf{assm}_\gamma(d_1) = d$.
4. Let $d = \mathbf{dschrg}(d_1, d_2)$, and let M assign trees in $der(G)$ to all and only unbound $\mathbf{assm}_\gamma$ nodes in d. By case 1 of the definition of SNART, we have that $\text{TRANS}(\text{SNART}(d, M))$ is equal to $\text{TRANS}(\mathbf{mv}_1(\text{SNART}(d_1, {}_{(0)}(M \cup \{\langle 0p, \text{SNART}(d_2, {}_{(1)}M)\rangle\}))))$, where $0p$ is the address of the $\mathbf{assm}_\gamma$ node in d bound by the **dschrg** node at its root. Next, we apply the first case of the definition of TRANS. This gives us $\mathbf{dschrg}(\text{TRANS}(\text{SNART}(d_1, {}_{(0)}(M \cup \{\langle 0p, \text{SNART}(d_2, {}_{(1)}M)\rangle\}))), \text{TRANS}(d'))$, where d' is the unique subtree targeted by the $\mathbf{mv}_1$ node at the root of the translated expression. This is the right daughter of the $\mathbf{mrg}_3$ node which the $\mathbf{assm}_\gamma$ node at position p in d_1 translates as, namely, $\text{SNART}(d_2, {}_{(1)}M)$. As ${}_{(0)}M$ assigns trees to all unbound $\mathbf{assm}_\gamma$ nodes in d_1 except for the one at location p, ${}_{(0)}(M \cup \{\langle 0p, \text{SNART}(d_2, {}_{(1)}M)\rangle\})$ assigns trees to *all* of d_1's unbound $\mathbf{assm}_\gamma$ nodes. Therefore, the induction hypothesis applies, and $\text{TRANS}(\text{SNART}(d, M))$ is seen to be identical to $\mathbf{dschrg}(d_1, d_2)$.
5. Finally, let $d = \mathbf{mv_2}(d_1)$, and M as assignment of trees to unbound $\mathbf{assm}_\gamma$ nodes in d. By case 3, $\text{SNART}(d, M) = \mathbf{mv_2}(\text{TRANS}(\text{SNART}(d_1, M)))$, which, by the induction hypothesis, is equal to d.

Theorem 4 allowed us to conclude that for every $d \in der(G)$ deriving a complete expression, there was a complete $d' \in \textit{Hyp-der}(G)$ deriving the same complete expression (whence corollary 2). From theorem 6 we are able to conclude the reverse as well.

Corollary 3. *For every G, and any feature sequence γ, $L_\gamma^{PBC}(G) = L_\gamma^{Hyp}(G)$.*

As Hyp MGs were shown in theorem 2 to be context free, we conclude that MGs subject to the proper binding constraint are as well.

6 Conclusion

We have demonstrated that movement by itself is not enough to describe non-context free languages; the super-CFness of the MG formalism is essentially tied

to remnant movement. This result confirms the intuition of several (cf. [15,21]), and seems related to the results of [18] in the context of the formalism of the GB theory presented therein.

Stabler [21] conjectures that:

> Grammars in $\mathcal{MG}$ can define languages with more than 2 counting dependencies only when some sentences in those languages are derived with remnant movements.

As we have shown here that MGs without remnant movement can only define context-free languages, we have proven Stabler's conjecture. However, we can in fact show a strengthened version of this conjecture to be true. Beyond the mere existence of remnant movement, where at item moves from which has already been moved another, we can identify hierarchies of such movement, depending on whether the item moved out of the 'remnant mover' is itself a remnant, and if so, whether the item moved out of that item is a remnant, and so on. We could place an upper bound of k on the so-defined degree of remnant movement we were willing to allow by, using the tree-structured representation of moving subpieces from our definition of PBC-MGs, allowing the move operations to target `-c` features of up to depth k in the tree. In this case, however, we could simply enrich the complexity of our hypotheses in the corresponding Hyp-MGs by a finite amount, which would not change their generative capacity. Thus, in order to derive non-context free languages, MGs must allow for movement of remnants of remnants of remnants. . . , in other words, a MG can define languages with more than two counting dependencies only when there is no bound k such that every sentence in the language is assigned a structure with remnant degree less than k.

Given that MGs can analyze non-CF patterns only in terms of unbounded remnant movement, one question these results make accessible is which such patterns in human languages are naturally so analyzed? Perhaps the most famous of the supra CF constructions in natural language is given by the relation between embedded verb clusters and their arguments in Swiss German [19]. [14] have provided an elegant analysis of verbal clusters in Germanic and Hungarian using remnant movement.[5] Patterns of copying in natural language [5,17,11] on the other hand, do not seem particularly naturally treated in terms of unbounded remnant movement. [11] shows how the addition of 'copy movement' (non-linear string manipulation operations) to the MG formalism allows for a natural treatment of these patterns, one that is orthogonal to the question of whether our grammars for natural language should use bounded or unbounded remnant movement.

[5] Not every linguist working in this tradition agrees that verb clusters are best treated in terms of remnant movement. [9] argues that remnant movement approaches to verb clustering are inferior to one using head movement. Adding head movement to MGs without remnant movement allows the generation of non-context free languages [16].

References

1. Chomsky, N.: Lectures on Government and Binding. Foris, Dordrecht (1981)
2. Chomsky, N.: The Minimalist Program. MIT Press, Cambridge (1995)
3. Collins, C.: A smuggling approach to the passive in English. Syntax 8(2), 81–120 (2005)
4. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications (2002), `http://www.grappa.univ-lille3.fr/tata`
5. Culy, C.: The complexity of the vocabulary of Bambara. Linguistics and Philosophy 8(3), 345–352 (1985)
6. Fiengo, R.: On trace theory. Linguistic Inquiry 8(1), 35–61 (1977)
7. Gärtner, H.M., Michaelis, J.: Some remarks on locality conditions and minimalist grammars. In: Sauerland, U., Gärtner, H.M. (eds.) Interfaces + Recursion = Language?, Studies in Generative Grammar, vol. 89, pp. 161–195. Mouton de Gruyter, Berlin (2007)
8. Gazdar, G., Klein, E., Pullum, G., Sag, I.: Generalized Phrase Structure Grammar. Harvard University Press, Cambridge (1985)
9. Haider, H.: V-clustering and clause union - causes and effects. In: Seuren, P., Kempen, G. (eds.) Verb Constructions in German and Dutch, pp. 91–126. John Benjamins, Amsterdam (2003)
10. Harkema, H.: Parsing Minimalist Languages. Ph.D. thesis, University of California, Los Angeles (2001)
11. Kobele, G.M.: Generating Copies: An investigation into structural identity in language and grammar. Ph.D. thesis, University of California, Los Angeles (2006)
12. Kobele, G.M.: A formal foundation for A and A-bar movement in the minimalist program. In: Kracht, M., Penn, G., Stabler, E.P. (eds.) Mathematics of Language, vol. 10. UCLA (2007)
13. Kobele, G.M., Retoré, C., Salvati, S.: An automata theoretic approach to minimalism. In: Rogers, J., Kepser, S. (eds.) Proceedings of the Workshop Model-Theoretic Syntax at 10; ESSLLI 2007, Dublin (2007)
14. Koopman, H., Szabolcsi, A.: Verbal Complexes. MIT Press, Cambridge (2000)
15. Michaelis, J.: On Formal Properties of Minimalist Grammars. Ph.D. thesis, Universität Potsdam (2001)
16. Michaelis, J.: Notes on the complexity of complex heads in a minimalist grammar. In: Proceedings of the Sixth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6), Venezia (2002)
17. Michaelis, J., Kracht, M.: Semilinearity as a syntactic invariant. In: Retoré, C. (ed.) LACL 1996. LNCS (LNAI), vol. 1328, pp. 37–40. Springer, Heidelberg (1997)
18. Rogers, J.: A Descriptive Approach to Language-Theoretic Complexity. CSLI Publications, Stanford (1998)
19. Shieber, S.M.: Evidence against the context-freeness of natural language. Linguistics and Philosophy 8, 333–343 (1985)
20. Stabler, E.P.: Derivational minimalism. In: Retoré, C. (ed.) LACL 1996. LNCS (LNAI), vol. 1328, pp. 68–95. Springer, Heidelberg (1997)
21. Stabler, E.P.: Remnant movement and complexity. In: Bouma, G., Hinrichs, E., Kruijff, G.J.M., Oehrle, R. (eds.) Constraints and Resources in Natural Language Syntax and Semantics, ch. 16, pp. 299–326. CSLI Publications, Stanford (1999)
22. Stabler, E.P., Keenan, E.L.: Structural similarity within and among languages. Theoretical Computer Science 293, 345–363 (2003)