Minimalist Tree Languages Are Closed Under Intersection with Recognizable Tree Languages

Gregory M. Kobele

University of Chicago
kobele@uchicago.edu

Abstract. Minimalist grammars are a mildly context-sensitive grammar framework within which analyses in mainstream chomskyian syntax can be faithfully represented. Here it is shown that both the derivation tree languages and derived tree languages of minimalist grammars are closed under intersection with regular tree languages. This allows us to conclude that taking into account the possibility of 'semantic crashes' in the standard approach to interpreting minimalist structures does not alter the strong generative capacity of the formalism. In addition, the addition to minimalist grammars of complexity filters is easily shown using a similar proof method to not change the class of derived tree languages.

Minimalist grammars (in the sense of [1]) are a formalization of mainstream chomskyian syntax. In this paper I will show that both derived and derivation tree languages of minimalist grammars are closed under intersection with regular tree languages. The technique used in the proofs of this fact is similar to that of [2], where non-terminals of context-free derivation trees were paired with states of an automaton. While the closure of the derived tree languages under regular intersection can be seen to follow from that of the derivation tree languages (by virtue of the monadic second order relation between the two), the proof method extends immediately to cases of linguistic interest where the connection is not as obvious, as in the case of 'complexity filters' in the sense of [3]. The closure of derived tree languages under regular intersection guarantees that the kind of semantic interpretation performed in the minimalist literature [4], which makes use of only a finite domain of types, cannot in virtue of partiality (semantic 'crashes') lead to sets of semantically well-formed trees which could not be directly derived by some minimalist grammar.

The remainder of the paper is organized as follows. The next section introduces minimalist grammars, as well as some relevant notation. Section 2 contains the proofs of closure under intersection with regular tree languages of both derivation and derived tree languages of minimalist grammars. Consequences and extensions of linguistic relevance are discussed in section 3. Finally, section 4 concludes.

1 Formal Preliminaries

Given a finite set A, A^* denotes the set of all finite sequences of elements over A. The symbol ϵ denotes the empty sequence. A ranked alphabet is a finite set

S. Pogodalla and J.-P. Prost (Eds.): LACL 2011, LNAI 6736, pp. 129–144, 2011.

F together with a function $\mathbf{rank} : F \to \mathbb{N}$ mapping each symbol in F to a natural number indicating its arity. Given $f \in F$ with arity $n = \mathbf{rank}(f)$, I will sometimes write $f^{(n)}$ to denote f while indicating that it has arity n. The set $T(F)$ of terms over a ranked alphabet F is the smallest subset of F^* containing all $f^{(0)} \in F$, and such that whenever it contains $t_1, \ldots, t_n$, it contains $f^{(n)}t_1 \cdots t_n$ for each $f^{(n)} \in F$. In lieu of writing $f^{(n)}t_1 \cdots t_n$, I will insert parentheses and commas for readability, writing in its stead $f(t_1, \ldots, t_n)$. Regular subsets of terms over an alphabet F can be given in terms of bottom-up tree automata, which are tuples $A = \langle Q, (\delta_f)_{f \in F} \rangle$, where each δ_f is a $\mathbf{rank}(f)$-ary function over Q. An automaton $A = \langle Q, (\delta_f)_{f \in F} \rangle$ induces a function $A : T(F) \to Q$ in the following manner: $A(f^{(n)}(t_1, \ldots, t_n)) = \delta_f(A(t_1), \ldots, A(t_n))$. For each state $q \in Q$, the set of terms $A^{-1}(q) = \{t \in T_F : A(t) = q\}$ mapped by A to the state q is a regular term language. It will be convenient in the following to treat A as operating over $T(F \cup Q)$, where elements of Q are treated as nullary symbols. In this case, $A(f^{(n)}(s_1, \ldots, s_n)) = \delta_f(q_1, \ldots, q_n)$, where $q_i = s_i$ if $s_i \in Q$, and $A(s_i)$ otherwise.

A partial function f from A to B is a total function from A to $B \cup \{\star\}$. If $f : A \to B$ is a partial function, we say it is undefined at $a \in A$ if $f(a) = \star$. The everywhere undefined function is denoted $\emptyset$. Given partial functions $f, g : A \to B$, their union $f \oplus g$ is defined iff there is no $a \in A$ such that both f and g are defined at a and $f(a) \neq g(a)$. In this case $(f \oplus g)(a) = \mathbf{if}\ f(a) = \star\ \mathbf{then}\ g(a)\ \mathbf{else}\ f(a)$. Given partial $f : A \to B$ and $a \in A$, $(f_{/a})(b) = \mathbf{if}\ b = a\ \mathbf{then}\ \star\ \mathbf{else}\ f(b)$. Given a subset $B \subset A^*$ with the property that $aw, au \in B$ implies that $w = u$, B can be viewed as the partial function f_B from A to A^* such that $f_B(a) = aw$ if $aw \in B$ and is undefined otherwise. In particular, given $aw \in A^*$, $\{aw\} : A \to A^*$ is the partial function defined only at a.

1.1 Minimalist Grammars

A minimalist grammar is given by a four-tuple $G = \langle \Sigma, \mathbf{sel}, \mathbf{lic}, Lex \rangle$ where Σ is a finite set, **sel** and **lic** are finite sets of *selection* and *licensing* features respectively which determine a set $\mathbb{F} := \{=x, x, +y, -y : x \in \mathbf{sel}, y \in \mathbf{lic}\}$ of *features*, $Lex \subset \Sigma \times \mathbb{F}^*$ is a finite set of lexical items. Features of the form $=x$ are *selector* features, those of the form $+y$ are *licensor* features, and those of the form x ($-y$) are *selectee* (*licensee*) features. Treating elements of Σ as nullary symbols, we define a ranked alphabet $S := \Sigma \cup \{\mathtt{t}^{(0)}, \bullet^{(2)}\}$. A term $t \in T(S)$ is *headed* by its right-most leaf. Conversely, a term $t' \in T(S)$ is a *maximal projection* (of its head) in t iff either $t' = t$ or there is a unary context $C[x]$, and some $t'' \in T(S)$ such that $t = C[\bullet(t', t'')]$ – in other words, t' is the left-daughter of some node. The expressions $L(G)$ generated by a minimalist grammar G is the smallest subset of $(T(S) \times \mathbb{F}^+)^+$ which 1) contains Lex, and 2) is closed under the operations presented in inference rule format below. In the rules below, $m, n \leq |\mathbf{lic}|$, $1 \leq i, j \leq m, n$, $\phi_i, \psi_j \in \Sigma \times \mathbb{F}^+$, $\gamma, \delta \in \mathbb{F}^+$, and

$s_1, s_2 \in T(S)$.[1] In addition, we require in $move_1$ and $move_2$ that ϕ_i is the only pair whose first feature is $-c$.[2]

$$\frac{\langle s_1, {=}c\gamma\rangle, \phi_1, \ldots, \phi_m \qquad \langle s_2, c\rangle, \psi_1, \ldots, \psi_n}{\langle \bullet(s_2, s_1), \gamma\rangle, \phi_1, \ldots, \phi_m, \psi_1, \ldots, \psi_n}\ merge_1$$

$$\frac{\langle s_1, {=}c\gamma\rangle, \phi_1, \ldots, \phi_m \qquad \langle s_2, c\delta\rangle, \psi_1, \ldots, \psi_n}{\langle \bullet(\mathtt{t}, s_1), \gamma\rangle, \phi_1, \ldots, \phi_m, \langle s_2, \delta\rangle, \psi_1, \ldots, \psi_n}\ merge_2$$

$$\frac{\langle s_1, {+}c\gamma\rangle, \phi_1, \ldots, \phi_{i-1}, \langle s_2, {-}c\rangle, \phi_{i+1}, \ldots, \phi_m}{\langle \bullet(s_2, s_1), \gamma\rangle, \phi_1, \ldots, \phi_{i-1}, \phi_{i+1}, \ldots, \phi_m}\ move_1$$

$$\frac{\langle s_1, {+}c\gamma\rangle, \phi_1, \ldots, \phi_{i-1}, \langle s_2, {-}c\delta\rangle, \phi_{i+1}, \ldots, \phi_m}{\langle \bullet(\mathtt{t}, s_1), \gamma\rangle, \phi_1, \ldots, \phi_{i-1}, \langle s_2, \delta\rangle, \phi_{i+1}, \ldots, \phi_m}\ move_2$$

An element of $L(G)$ is *complete* iff it is of the form $\langle s, c\rangle$. An element of $T(S)$ is a derived (or *surface*) tree iff it is the first component of a complete expression. The derived tree language of G at selectee feature c is defined to be the set of tree components of complete expressions with feature c, $L_c(G) := \{s : \langle s, c\rangle \in L(G)\}$.

The string language of G at selectee feature c (written $S_c(G)$) is defined to be the image of $L_c(G)$ under the mapping $\mathsf{y} : T(S) \to (S_0 - \{\mathtt{t}\})^*$ which maps a tree to the string consisting of its leaves in left-to-right order, ignoring traces; $\mathsf{y}(\mathtt{t}) = \epsilon$, $\mathsf{y}(s^{(0)}) = s$, and $\mathsf{y}(\bullet(t_1, t_2)) = \mathsf{y}(t_1)^\frown\mathsf{y}(t_2)$.

The following obvious proposition shows that every maximal projection is headed by some leaf.

Proposition 1. *Let G and $x \in \mathbb{F}$ be arbitrary. Then for every $t \in L_x(G)$ there is a unique maximal decomposition* LexProj(t) *of t into lexicalized maximal projections of the form* $\mathrm{TCon}_s^n[t_1, \ldots, t_n]$*, where* $\mathtt{t} \neq s^{(0)} \in S$*, $t_1, \ldots, t_n \in T(S)$, and* TCon_s^n *is the n-ary context over $T(S)$ defined as follows:*

1. $\mathrm{TCon}_s^0 := s$
2. $\mathrm{TCon}_s^{n+1} := \bullet(x_{n+1}, \mathrm{TCon}_s^n)$

Proof. This follows from the observation that every occurance of a trace ($\mathtt{t}$) in the surface tree of a complete expression is already itself a maximal projection (by inspection of the operations), and thus that every maximal projection is 'lexicalized'. □

Treating elements of Lex as nullary symbols, we define a ranked alphabet $U := Lex \cup \{\mathbf{r}^{(2)}, \mathbf{v}^{(1)}\}$. The set of derivation (or *underlying*) terms over G is $D(G) := T(U)$. Given $d \in D(G)$, we write $\mathsf{ev}(d)$ to denote the expression which is the evaluation of d in $L(G)$, if it exists ($\mathsf{ev} : D(G) \to L(G)$ is the partial injective mapping $\mathsf{ev}(\ell) = \ell$, $\mathsf{ev}(\mathbf{v}(d)) = move(\mathsf{ev}(d))$, and $\mathsf{ev}(\mathbf{r}(d_1, d_2)) = merge(\mathsf{ev}(d_1), \mathsf{ev}(d_2))$.)

[1] In the following rules, all move and merge operations are 'to the left'. This simplification made for expository purposes does not affect the results obtained in this paper.

[2] This constraint is called the *Shortest Move Constraint* (SMC).

We identify the set of well-formed, or *convergent*, derivations with the language (at a particular state) of a bottom up tree automaton $A_G = \langle Q, (\delta_f)_{f \in F} \rangle$, which is defined next. Given Lex we define $\mathsf{suf}(Lex) = \{\eta : \exists \sigma \in \Sigma^*, \gamma \in \mathbb{F}^*.\ \langle \sigma, \gamma\eta \rangle \in Lex\}$ to be the set of suffixes of lexical feature sequences. The states of our automaton are pairs $\langle \eta, f \rangle$, where $\eta \in \mathsf{suf}(Lex)$ and $f : \{\text{-}y : y \in \mathbf{lic}\} \to \mathsf{suf}(Lex)$ is a partial function.[3] The functions $(\delta_f)_{f \in F}$ are defined as per the following.

- For each lexical item $\ell = \langle \sigma, \eta \rangle \in Lex$, $\delta_\ell = \langle \eta, \emptyset \rangle$.
- Given state $s = \langle \text{+}y\eta, f \rangle$ with $f(\text{-}y) = \text{-}y\gamma$, $\delta_{\mathbf{v}}$ is defined at s iff either
  1. $\gamma = \epsilon$, in which case $\delta_{\mathbf{v}}(s) = \langle \eta, f_{/\text{-}y} \rangle$
  2. $\gamma = \text{-}z\eta$ and f is undefined at $\text{-}z$, in which case $\delta_{\mathbf{v}}(s) = \langle \eta, f_{/\text{-}y} \oplus \{\gamma\} \rangle$
- Given states $s = \langle \text{=}x\eta, f \rangle$ and $s' = \langle x\eta', f' \rangle$ with $f \oplus f'$ defined, $\delta_{\mathbf{r}}$ is defined at the pair s, s' iff either
  1. $\eta' = \epsilon$, in which case $\delta_{\mathbf{r}}(s, s') = \langle \eta, f \oplus f' \rangle$
  2. $\eta' = \text{-}y\gamma$ and $f \oplus f'$ is undefined at $\text{-}y$, in which case $\delta_{\mathbf{r}}(s, s') = \langle \eta, (f \oplus f') \oplus \{\eta'\} \rangle$

A derivation $d \in D(G)$ is *saturated* iff there is some state $\langle x\eta, f \rangle$ beginning with a selectee feature such that $A_G(d) = \langle x\eta, f \rangle$. The convergent derivations at selectee feature x are defined to be those in $D_x(G) := A_G^{-1}(\langle x, \emptyset \rangle) = \{t \in T(U) : A_G(t) = \langle x, \emptyset \rangle\}$. It was shown in [5] that for each x, there is a finite copying top-down tree transducer with regular look-ahead which maps $D_x(G)$ to $L_x(G)$. This is a simple consequence of the facts that the inference rules above can be put into the format of a multiple regular tree grammar, that we can restrict our attention to a finite set of 'categories' (as given by $\mathsf{suf}(Lex)^{|\mathbf{lic}|+1}$ [6]), and that a multiple regular tree grammar can be presented in terms of a finite copying top-down tree transducer with regular look-ahead acting on a regular set [7].

The following facts about convergent derivations will prove useful in the next section.

Proposition 2. *For any G, and any selectee feature x, the following are true:*

1. *if $t \in D_x(G)$ then every leaf of t is of the form $\langle \sigma\ ,\ \eta c \gamma \rangle$ for $\eta \in \{\text{=}x, \text{+}y : x \in \textbf{\textit{sel}} \,\&\, y \in \textbf{\textit{lic}}\}^*$ and $\gamma \in \{\text{-}y : y \in \textbf{\textit{lic}}\}^*$*
2. *if $t \in D_x(G)$ then every leaf ℓ of t occurs in the n-ary context $\textsc{Con}_\ell(\eta)$, where n is the number of selector features ℓ contains, and η is the initial sequence of selector and licensor features of ℓ, and where $\textsc{Con}_\ell(\cdot)$ is defined as follows:*

   (a) $\textsc{Con}_\ell(\epsilon) = \ell$
   (b) $\textsc{Con}_\ell(\eta\ \text{+}y) = \boldsymbol{v}(\textsc{Con}_\ell(\eta))$
   (c) $\textsc{Con}_\ell(\eta\ \text{=}x) = \boldsymbol{r}(\textsc{Con}_\ell(\eta), x_i)$, *where η contains i selector features*

[3] The SMC condition on the domains of the movement operations allows us to disregard expressions with more than one component beginning with the same feature.

Proof. 1 is proven by inspection of the definition of the transition functions δ_f for the automaton A_G; a state $\langle \texttt{+}z\eta, f\rangle$ where $f(\texttt{-}z) = x\gamma$, with $x \notin \{\texttt{-}y : y \in \mathbf{lic}\}$ is not in the domain of any transition function, nor is a state $\langle \texttt{-}z\eta, f\rangle$.

2 is proven by the observations that $A_G(\mathbf{r}(\langle \eta, f\rangle, \langle \gamma, g\rangle))$ is defined only if η begins with some selector feature and γ with some selectee feature, and that $A_G(\mathbf{v}(\langle \eta, f\rangle))$ is defined only if η begins with some licensor feature. □

Proposition 2 justifies us in restricting our attention to just those lexica which contain lexical items of the form $\langle \sigma\ ,\ \eta c\gamma\rangle$, where η is a string of selector and licensor features, and γ a string of licensee features. In the following, for ℓ as above, we write $\textsc{Con}_\ell$ as an abbreviation for $\textsc{Con}_\ell(\eta)$.

The equivalent of part two of proposition 2 for surface trees can be proven.

Proposition 3. *Let G and selectee feature x be arbitrary. Then for every $t \in L_x(G)$, each non-trace leaf s in t occurs in a context $\textsc{TCon}_s^{|\eta|}$, for some lexical item $\langle s\ ,\ \eta c\gamma\rangle \in Lex$.*

Proof. Let G, x, t, and s be as in the statement of the proposition, and let $d \in D_x(G)$ be a derivation of t. As the operations of a minimalist grammar are linear, non-deleting, and introduce only $\texttt{t}$ and $\bullet$ syncategorimatically, we can identify for each leaf of t the lexical item in d from which it came. Let s be derived from an occurrence of the lexical item $\ell = \langle s\ ,\ \eta c\gamma\rangle$ in d. This item occurs by proposition 2 in the context $\textsc{Con}_\ell$, and so there are $d_1, \ldots, d_{|\eta|}$ such that $d' = \textsc{Con}_\ell[d_1, \ldots, d_{|\eta|}]$ is the occurrence of (the projection of ℓ) in d. By inspection of the operations, $\mathsf{ev}(d') = \langle r, c\gamma\rangle, \phi_1, \ldots, \phi_k$ is seen to be such that r instantiates the context $\textsc{TCon}_s^{|\eta|}$. As no further operations can modify r internally, r occurs in t. □

Proposition 3 implies that once we know the identity and numerosity of the lexical items $\ell_1, \ldots, \ell_n$ in a complete derivation, the derived tree is gotten by putting their respective $\textsc{TCon}_\ell$ together, along with a certain number of syncategorematic traces (equal to the total number of licensee features across the ℓ_i).

2 Languages

The basic idea of the construction in both proofs is that each attractor feature of a minimalist lexical item can require that the element whose feature it checks have a certain property. Furthermore, each attractee feature can indicate that its lexical item has a certain property. This is done by annotating a feature with a representation of a particular property. (Then $\texttt{=}x^p$ indicates that it is looking for an x with property p, and $\texttt{-}y^p$ indicates that it is a $\texttt{-}y$ which has property p.) In the cases we will be interested in, P will be the state set of a finite state automaton.

Given a finite set P of properties, we define a function $\langle\cdot\rangle_P$ which maps a feature type $f \in \mathbf{sel} \cup \mathbf{lic}$ to the set $\{f^p : p \in P\}$ of its P-variants. Abusing

notation, we write $\langle\cdot\rangle_P$ for its extension to features ($\langle *x\rangle_P := \{*x^p : p \in P\}$), to sequences ($\langle aw\rangle_P := \langle a\rangle_P \cdot \langle w\rangle_P$), and to sets ($\langle X\rangle_P := \bigcup_{x\in X}\langle x\rangle_P$). The set of P-variants of a lexical item ℓ is the set of lexical items whose feature sequences are P-variants of the feature sequence of ℓ ($\langle\langle\sigma\ ,\ \gamma\rangle\rangle_P := \{\langle\sigma\ ,\ \eta\rangle : \eta \in \langle\gamma\rangle_P\}$), and the P-variant of a grammar $G = \langle\Sigma, \mathbf{sel}, \mathbf{lic}, Lex\rangle$ is the grammar whose feature types, and lexicon are P-variants of G's ($\langle G\rangle_P := \langle\Sigma, \langle\mathbf{sel}\rangle_P, \langle\mathbf{lic}\rangle_P, \langle Lex\rangle_P\rangle$). The operation $\langle\cdot\rangle_P$ can be extended in the obvious way to derivation trees, mapping leaves (lexical items) to their sets of P-variants, and acting pointwise on non-leaf nodes ($\langle f(t_1, \ldots, t_n)\rangle_P := \{f(s_1, \ldots, s_n) : s_i \in \langle t_i\rangle_P \text{ for } 1 \leq i \leq n\}$).

Proposition 4. *For any G, any P, $x \in \mathbb{F}$ and $q \in P$, if $d \in D_{x^q}(\langle G\rangle_P)$, then $\langle d\rangle_P^{-1} \in D_x(G)$. If $|P| = 1$, then $d \in D_x(G)$ implies $\langle d\rangle_P \in D_{x^q}(\langle G\rangle_P)$.*

Proof. Let A_G and $A_{\langle G\rangle_P}$ be the automata over derivations of G and $\langle G\rangle_P$ described in the previous section. Viewed as a function from derivations to states, we show that $A_G = \langle\cdot\rangle_P^{-1} \circ A_{\langle G\rangle_P} \circ \langle\cdot\rangle_P$. First, we have that $\mathsf{suf}(Lex) = \langle\mathsf{suf}(\langle Lex\rangle_P)\rangle_P^{-1}$, and thus that the states of A_G and $A_{\langle G\rangle_P}$ are in a one-many correspondence (via $\langle\cdot\rangle_P$). By the definitions of $\delta_{\mathbf{v}}$ and $\delta_{\mathbf{r}}$, the transitions at non-leaf nodes of both machines partially commute with $\langle\cdot\rangle_P$, in the sense that if $\delta(s_1, \ldots, s_n)$ is defined (in $A_{\langle G\rangle_P}$), then $\delta(\langle s_1\rangle_P^{-1}, \ldots, \langle s_n\rangle_P^{-1})$ is defined (in A_G) and is equal to $\langle\delta(s_1, \ldots, s_n)\rangle_P^{-1}$. Finally, for every $\ell' \in \langle Lex\rangle_P$, $\langle\delta_{\ell'}\rangle_P^{-1} = \delta_{\langle\ell'\rangle_P^{-1}}$.

If $|P| = 1$, then the states are in a bijective correspondence, the transitions of both machines at non-leaf nodes commute with $\langle\cdot\rangle_P$, and $\langle\delta_\ell\rangle_P = \delta_{\langle\ell\rangle_P}$. □

Proposition 5. *For any G, any P, and any $x \in \mathbb{F}$, it holds for all $q \in P$ that $L_x(G) = L_{x^q}(\langle G\rangle_P)$.*

Proof. In the forward direction, let q and x be arbitrary, and let $t \in L_x(G)$, with derivation $d \in D_x(G)$. By proposition 4, $\langle d\rangle_{\{q\}} \in D_{x^q}(\langle G\rangle_P)$. As features can be renamed arbitrarily (as long as features that are supposed to match still do) without affecting the identity of the tree derived, $\langle d\rangle_{\{q\}}$ evaluates to $t \in L_{x^q}(\langle G\rangle_P)$.

In the reverse direction, let q and x be arbitrary and $t \in L_{x^q}(\langle G\rangle_P)$ with derivation $d \in D_{x^q}(\langle G\rangle_P)$. By proposition 4, $\langle d\rangle_P^{-1} \in D_x(G)$. By the reasoning just above, evaluating $\langle d\rangle_P^{-1}$ yields $t \in L_x(G)$. □

While propositions 4 and 5 show that marking up a lexicon does not change the derived trees, it *does* have the effect of multiplying derivations for any given derived object. By selectively removing lexical items from $\langle Lex\rangle_P$, we can *ipso facto* impose meanings on the property symbols.

Example 6. To require that every derivation include lexical item $\ell \in Lex$, we take $P = \{0, 1\}$, and consider the largest subset X of $\langle Lex\rangle_P$ which satisfies the following conditions:

1. if $\ell' \in X$ and $\ell' \in \langle\ell\rangle_P$ then the property associated with its category feature z is 1

2. if $\ell' \in X$ and $\ell' \notin \langle \ell \rangle_P$, then the property associated with its category feature z is the maximum of the properties associated with its selector features $=x_i$.

Then every well-formed derivation tree headed by a lexical item with a category feature with property 1 contains some ℓ' with $\ell' \in \langle \ell \rangle_P$, and conversely, those headed by lexical items with some category feature z^0 do not contain an ℓ.

This example can actually be viewed as a special case of a more general construction, which restricts our attention to just those derivations that satisfy some property defined by a regular tree automaton.

Proposition 7. *Let G be a minimalist grammar, x a feature, and L a regular subset of $T(U)$. There is a minimalist grammar G_L, a feature y, and a set P such that $\langle D_y(G_L) \rangle_P^{-1} = D_x(G) \cap L$.*

Proof. Let G, x, and L be arbitrary as in the statement of the proposition. Let $A = \langle Q, (\delta)_{f \in U} \rangle$ be a deterministic bottom-up tree automaton such that for some $r \in Q$, $A^{-1}(r) = L$. The feature y in the statement of the proposition will be identified with x^r. We will be interested in a particular subset $Lex_A \subseteq \langle Lex \rangle_Q$, in which the annotations on the features of lexical items reflect the behaviour of A on the well-formed derivation trees they occur in.

First, we define a variant of the Con function from proposition 2, a mapping $V : \langle Lex \rangle_Q \to T(U \cup Q)$ from annotated lexical items to terms over U and Q, in the following manner. Writing V_ℓ for $V(\ell)$, we define $V_\ell := \textsc{Con}_{\langle \ell \rangle_Q^{-1}}[q_1, \ldots, q_n]$, where $\ell = \langle \sigma \ , =x_1^{q_1} \eta_1 \cdots =x_n^{q_n} \eta_n c \gamma \rangle$, and for $1 \leq i \leq n$ $\eta_i \in \{+y : y \in \langle \mathbf{lic} \rangle_Q\}^*$. Intuitively, V_ℓ calls $\textsc{Con}_\ell$, and fills in the variables where an argument would be merged with the state that the automaton must be in when reading the subtree occuring in that position.

Now, let Lex_A be the smallest subset of $\langle Lex \rangle_Q$, such that it contains a lexical item $\ell = \langle \sigma \ , *x_1^{a_1} \cdots *x_n^{a_n} z^a -y_1^{b_1} \cdots -y_m^{b_m} \rangle \in Lex_Q$ iff $a = A(V_\ell)$. Intuitively, we are removing all the lexical items from $\langle Lex \rangle_Q$ whose feature annotations on selector and category features do not accurately reflect the behaviour of A on the contexts in which they occur. We prove that for $d \in D(G_L)$ a saturated derivation tree with first feature z^q, $A(\langle d \rangle_Q^{-1}) = q$ by induction on the heights of saturated derivation trees. For the base case, let $\ell = \langle \sigma \ , z -y_1 \cdots -y_m \rangle$, and let $A(\ell) = q$. Then Lex_A contains an item $\ell = \langle \sigma \ , z^a -y_1^{b_1} \cdots -y_m^{b_m} \rangle \in Lex_Q$ iff $a = A(V_\ell) = A(\langle \ell \rangle_Q^{-1}) = q$. Now assume that the proposition holds of saturated derivation trees up to height $n-1$, and let saturated $d \in D(G_L)$ have height n. Then $d = \textsc{Con}_{\langle \ell \rangle_Q^{-1}}[d_1, \ldots, d_k]$ for some $\ell = \langle \sigma \ , *x_1^{r_1} \cdots *x_j^{r_j} z^q \gamma \rangle \in Lex_A$, and saturated derivation trees $d_1, \ldots, d_k$. By the inductive hypothesis, $A(\langle d_i \rangle_Q^{-1}) = q_i$, where the first feature of d_i is $z_i^{q_i}$, for $1 \leq i \leq k$. Then $A(\textsc{Con}_{\langle \ell \rangle_Q^{-1}}[d_1, \ldots, d_k]) = A(\textsc{Con}_{\langle \ell \rangle_Q^{-1}}[q_1, \ldots, q_n]) = A(V_\ell) = q$.

Now we are in a position to prove the original proposition. Define $G_L := \langle \Sigma, \langle \mathbf{sel} \rangle_Q, \langle \mathbf{lic} \rangle_Q, Lex_A \rangle$, and let $t \in D_{x^r}(G_L)$. By proposition 4, $\langle t \rangle_Q^{-1} \in D_x(G)$. As we saw in the previous paragraph, $A(\langle t \rangle_Q^{-1}) = r$, and thus $\langle t \rangle_Q^{-1} \in L$. This establishes the forward containment ($\subseteq$). For the reverse direction ($\supseteq$), let $t \in$

$D_x(G) \cap L$, and let $t' \in \langle t \rangle_Q$ be such that every selector feature on every leaf in t is annotated with the state that A is in when it has read the corresponding argument to that feature, and every selectee feature on every leaf in t is annotated with the state that A is in when it has read the entire subtree of t which is the instantiation of the context of that leaf. Clearly, $t' \in D(G_L)$. As $t \in L$, $t' \in D_{x^r}(G_L)$, which concludes the proof. □

It is worth remarking about the constructed G_L in the proof of proposition 7 that its lexical items exhibit only certain dependencies amongst their features. In particular, if some lexical item $\ell \in Lex_A$ has as its i^{th} feature a licensee feature $\texttt{-}y^p$, then Lex_A contains every ℓ' like ℓ except that instead of $\texttt{-}y^p$, its i^{th} feature is $\texttt{-}y^q$, for some $q \in Q$. The same holds true of $\texttt{+}y$ features, but *not* in general for $\texttt{=}x$ and x features.

From a high-level perspective, proposition 7 shows that we may 'push' down into the features of the leaves the state-annotations on non-terminals in a derivation tree obtained from Thatcher's [2] construction of a context-free grammar from a bottom-up finite state tree automaton.

2.1 Closure under Intersection with Regular Sets

In the previous section we noted that when incorporating regular restrictions on derivations, we only needed to enforce dependencies among selection features. Intuitively, all of the information that an automaton traversing a derivation tree might care about is present already when two expressions are merged together, and so to 'keep track' of this information we only need to encode it on the features relevant for merger.

By placing constraints on the dependencies between licensing features, we allow information to be communicated about aspects of the derived objects, such as whether a particular movement is of an expression that will remain in its current position, or of one which will continue on (and leave behind a trace).

Proposition 8. *Let G be a minimalist grammar, and $L \subseteq T(S)$ a regular tree language. Then for any x there is a minimalist grammar G^L and a feature y such that $L_y(G^L) = L_x(G) \cap L$.*

Proof. Let G and L and x be arbitrary, and let $A = \langle Q, (\delta_s)_{s \in S} \rangle$ be a bottom-up finite tree automaton such that for some $r \in Q$, $A^{-1}(r) = L$. As before, we will identify the y in the statement of the theorem with x^r. Define the state $t = A(\texttt{t})$ to be the state A is in when it has scanned a trace. As before, we will define a subset $Lex^A \subseteq \langle Lex \rangle_Q$ to be our lexicon. First we define a 'surface counterpart' of the V mapping from proposition 7, a function $R : T(S) \times Q^* \to T(S \cup Q)$ from derived tree – state sequence pairs to terms over S and Q, in the following manner. Writing $R_t(w)$ for $R(t, w)$, we define $R_t(w) := \textsc{Tcon}_t^{|w|}[w_1, \ldots, w_{|w|}]$, where $\textsc{Tcon}_t^n$ is as in the statement of proposition 1.

Now we define $Lex^A \subseteq \langle Lex \rangle_Q$ to be the smallest set such that it contains a lexical item $\langle \sigma \, , \, \texttt{*}x_1^{a_1} \cdots \texttt{*}x_n^{a_n} z^t \texttt{-}y_1^t \cdots \texttt{-}y_m^a \rangle \in \langle \ell \rangle_Q$ iff $a \in A(R_\sigma(a_1 \cdots a_n))$. The selector and licensor feature annotations are intended to indicate the state the

automaton is in when it scans the surface item in that position, and the selectee and licensee feature annotations provide information about what surface term appears in each of the positions associated with these features. Note that all but the last selectee and licensee features are annotated with a 'trace' – this is because an expression leaves behind a trace in all but its final position.

Now let $X \subseteq L(G^L)$ be the subset of $L(G^L)$ containing all and only expressions $\phi_0, \ldots, \phi_k$ which meet the conditions that

1. for $1 \leq i \leq k$, the last feature in each ϕ_i is annotated with the state that A is in when it scans the tree component of ϕ_i, all other features of each ϕ_i are annotated with t (the state A is in when it scans a trace)
2. $\phi_0 = \langle s, *x_1^{a_1} \cdots *x_j^{a_j} z^{b_0} - y_1^{b_1} \cdots - y_m^{b_m} \rangle$ is such that $A(R_s(a_1 \cdots a_j)) = b_m$ and $b_0, \ldots, b_{m-1} = t$

We show that X contains Lex^A and is closed under $merge_i$ and $move_i$, for $1 \leq i \leq 2$, and is therefore equal to $L(G^L)$.

- Let $\ell = \langle \sigma \ , *x_1^{a_1} \cdots *x_j^{a_j} z^{b_0} - y_1^{b_1} \cdots - y_m^{b_m} \rangle \in Lex^A$. Then by definition, $b_0 \ldots, b_{m-1} = t$, and $b_m = A(R_\sigma(a_1 \cdots a_j))$.
- Let $\phi_0, \ldots, \phi_k \in X$ be in the domain of the $move_1$ operation. Then $\phi_0 = \langle s, +x_1^{a_1} * x_2^{a_2} \cdots *x_j^{a_j} z^{b_0} - y_1^{b_1} \cdots - y_m^{b_m} \rangle$, and for some i, $\phi_i = \langle r, -x_1^{a_1} \rangle$, where by hypothesis $a_1 = A(r)$, and $b_m = A(R_s(a_1 \cdots a_j))$. Then $move_1(\phi_0, \ldots, \phi_k)$ is the expression $\phi'_0, \phi_1, \ldots, \phi_{i-1}, \phi_{i+1}, \ldots, \phi_k$, where ϕ'_0 has as its first component the term $\bullet(r, s)$, and second component the tail of the feature sequence of ϕ_0. By the definition of R, we see that $A(R_{\bullet(r,s)}(a_2 \cdots a_j)) = A(R_s(A(r)\ a_2 \cdots a_j)) = A(R_s(a_1\ a_2 \cdots a_j)) = b_m$.
- Let $\phi_0, \ldots, \phi_k \in X$ be in the domain of the $move_2$ operation. Then $\phi_0 = \langle s, +x_1^{a_1} * x_2^{a_2} \cdots *x_j^{a_j} z^{b_0} - y_1^{b_1} \cdots - y_m^{b_m} \rangle$, and $\phi_i = \langle r, -x_1^{a_1} - z_1^{c_1} \cdots - z_h^{c_h} \rangle$ for some i, where by hypothesis $a_1 = t$, $c_1, \ldots, c_{h-1} = t$, $c_h = A(r)$, and $b_m = A(R_s(a_1 \cdots a_j))$. Then the result of applying $move_2$ to $\phi_0, \ldots, \phi_k$ is the expression $\phi'_0, \phi_1, \ldots, \phi'_i, \ldots, \phi_k$, where ϕ'_0 has as its first component the term $\bullet(\mathtt{t}, s)$, and second component the tail of the feature sequence of ϕ_0, and $\phi'_i = \langle r, -z_1^{c_1} \cdots - z_h^{c_h} \rangle$. By the definition of R, we see that $A(R_{\bullet(\mathtt{t},s)}(a_2 \cdots a_j)) = A(R_s(t\ a_2 \cdots a_j)) = A(R_s(a_1\ a_2 \cdots a_j)) = b_m$.
- the proof for $merge_1$ is similar to that of $move_1$ and is omitted.
- the proof for $merge_2$ is similar to that of $move_2$ and is omitted.

Now let $G^L := \langle \Sigma, \langle \mathbf{sel} \rangle_Q, \langle \mathbf{lic} \rangle_Q, Lex^A \rangle$. In the forward direction, let $t \in L_{x^r}(G^L)$. By proposition 5 we have that $t \in L_x(G)$. As $t \in L_{x^r}(G^L)$, it holds that $\langle t, x^r \rangle \in L(G^L)$, which has properties 1 and 2 above, whence $t \in A^{-1}(r) = L$. In the reverse direction, let $t \in L_x(G) \cap L$ with derivation $d \in D_x(G)$. For each non-trace leaf s in t identify the lexical item ℓ_s in d from which it came (as per proposition 3), and associate s with its annotated surface context $R_s(A(t_1), \ldots, A(t_n))$, for $t_1, \ldots, t_n \in T(S)$ such that $\textsc{TCon}_s^n[t_1, \ldots, t_n] \in \textsc{LexProj}(t)$. Define $\ell'_s \in \langle Lex \rangle_Q$ to be like ℓ_s but where the selector and licensor features are annotated from left to right with $A(t_1)$ throught $A(t_n)$, the last feature with $A(R_s(A(t_1), \ldots, A(t_n)))$, and all remaining feautres with t. Clearly,

$\ell'_s \in Lex^A$, and the d' obtained by replacing all leaves in d in this manner is a derivation in $D(G^L)$. That it is a complete derivation of category x^r follows from its construction, whence $\mathsf{ev}(d') = \langle t, x^r\rangle$, and $t \in L_{x^r}(G^L)$, as desired. □

As an application of proposition 8, we show the known result that the string languages of minimalist grammars are closed under intersection with regular string languages. We do this by generating a finite state tree automaton recognizing all trees with yields in the given regular string language, which can then by 8 be 'intersected' with an arbitrary minimalist grammar.

Example 9. Let $M = \langle Q^M, q_0^M, \zeta, q_f^M\rangle$ be a regular *string* automaton, with state set Q^M, initial and final states q_0^M and q_f^M, respectively, and transition function $\zeta : Q^M \times \Sigma \to Q^M$. The set of trees over $T(S)$ whose yield is in $L(M)$ is given by the regular *nondeterministic* tree automaton $A_M = \langle Q^A, Q_f^A, (\delta_f)_{f\in\Sigma}\rangle$ (where Q_f^A is the set of final states) defined as follows:

1. $Q^A = \left[Q^M \to Q^M\right]$
2. $Q_f^A = \{q \in Q^A : q(q_0^M) = q_f^M\}$
3. For $\mathtt{t}^{(0)}$, $\delta_{\mathtt{t}}(q) = q$
4. For $\sigma \in \Sigma_0$, $\delta_\sigma(q) = \lambda x.\zeta(q(x), \sigma)$
5. For $\bullet^{(2)}$, $\delta_\bullet(q_1, q_2) = q_1 \circ q_2$

Define $L := \bigcup_{q\in Q_f^A} A_M^{-1}(q)$. Then for G a minimalist grammar and x a selectee feature, there is a feature y such that, $S_y(G^L)) = S_x(G) \cap L(M)$.

Example 9 can be viewed as a particular instance of a much more general fact.[4]

Proposition 10. ***(Courcelle [8])***

1. *The inverse image of a (MSO) definable set of structures under a (MSO) definable transduction is (MSO) definable*
2. *The composition of two (MSO) definable transductions is (MSO) definable.*

In particular, if $\phi : T(S) \to \Delta^*$ is an MSO definable spell-out mapping, associating trees with strings over Δ, and $L \subseteq \Delta^*$ is a regular string language of 'what might have been just said', then $\phi^{-1}[L] = \{t \in T(S) : \phi(t) \in L\}$ is by proposition 10 a regular tree language, and thus for any minimalist grammar G, we can construct another minimalist grammar $G^{\phi^{-1}[L]}$ which derives exactly those trees in $\phi^{-1}[L]$ which are derived by G. As an example, mirror-style head

[4] In fact, proposition 8 follows from propositions 7 and 10 as a corollary, as was pointed out by an anonymous reviewer. Let L be a regular tree language, and ϕ_L a MSO formula defining exactly L. Let G be an arbitrary minimalist grammar, and let Δ be the MSO definable transduction taking exactly $D_x(G)$ to $L_x(G)$. Then by proposition 10, $S = \Delta^{-1}[L]$ is a regular (i.e. MSO-definable) subset of $D_x(G)$ which contains a derivation tree d iff d is the derivation of some tree $t \in L$, and thus by proposition 7, G_S exists, and is such that for some y, $L_y(G_S) = L_x(G) \cap L$.

movement, as implemented by [9], can be treated in terms of a non-standard (but MSO definable) spell-out mapping from minimalist trees into strings.

More generally, given a minimalist grammar G, any sequence $\phi_1, \ldots, \phi_n$ of MSO-definable mappings $\phi_i : A_i \to A_{i+1}$, where $A_0 = L(G)$ and $A_{n+1} = \Delta^*$ can be 'undone', in the sense that for any MSO-definable subset $L \subseteq \Delta^*$, $(\phi_n \circ \cdots \circ \phi_1)^{-1}(L)$ is a regular subset of $L(G)$, and there is therefore a minimalist grammar $G^{(\phi_n \circ \cdots \circ \phi_1)^{-1}(L)}$ which generates exactly this regular subset.

Not only does this yield a guarantee of mild-context sensitivity, as the string languages generable from finitely many MSO transductions applied successively to a regular tree language is exactly the set of multiple context free languages (see [10]), this also gives a sort of 'parsing as intersection' result [11] for any extension of the basic framework of minimalist grammars by MSO means.

3 Applications

In this section I discuss three applications of propositions 7 and 8 above.[5] I begin by discussing conditions under which semantic type mismatches (and the consequent deviance of an utterance) can be incorporated into the grammar. Next I introduce Koopman's 'complexity filters', and show that they do not increase the tree generating power of minimalist grammars. Finally, I offer a formalization of distributed morphology, and show that it is weakly equivalent to minimalist grammars.

3.1 Semantics

Semantics is often couched in a typed system. In this kind of framework, a structure might be unable to be assigned a meaning due to a type mismatch, and thus a semantic interpretation function might be partial. In such a case the meaningful structures generated by a grammar might be a proper subset of the structures it generates. Does the possibility of semantic type mismatch render the problem of finding meaningful parses for strings computationally more difficult? This question is especially salient in the context of the extended standard theory [13] and its decendents, where semantic interpretation is of the derived structure, and not of the structure of the derivation.

Propositions 7 and 8 tell us that, if we can represent the property of being semantically well-formed in terms of a bottom-up tree automaton, then we can transform a minimalist grammar generating possibly semantically uninterpretable surface trees into one which directly generates only the subset of the first which are semantically interpretable. This will hold true whether we interpret surface structures (as do [4]) or derivation trees (as is done in [14]).

While it is not in general true that given a set of typed constants, there are only finitely many types derivable from them, the textbook of [4] (which is the *locus classicus* for semantics in the chomskyian vein) eschews more than application

[5] Another application which I do not discuss is to work on economy constraints in minimalist grammars [12].

$(\alpha\beta \oplus \alpha \Rightarrow \beta)$ and (sometimes) 'predicate modification' $(\alpha t \oplus \alpha t \Rightarrow \alpha t)$, closing any finite set of types under which results in a finite set of types.

3.2 Complexity Filters

[15] analyze verbal complexes in Dutch, German and Hungarian in a syntactic framework much like the one presented here. They propose to treat these languages as similar for the purposes of this construction, with the primary locus of cross-linguistic variation being that some configurations are filtered out in certain languages. In general, the approach of [15], following [16], is to postulate a great number of silent terminal nodes, and that syntactic terminals get moved about repeatedly in a great number of ways. While this sort of approach allows for the 'discovery' of similar structures underlying very different surface strings (and in different languages), it tends to overgenerate. 'Complexity filters', as proposed by [15] (and expanded upon in [17,3]), are a way of reigning in this overgeneration. Koopman proposes that each lexical item ℓ may impose requirements on the *size* of the surface expressions $t_1, \ldots, t_n$ which may occur in $\text{TC}\textsc{on}_\ell[t_1, \ldots, t_n]$. The size of a tree t_i is defined in terms of the length of and labels along the path from the most deeply embedded phonetically overt terminal in t_i to its root. As a concrete example, [3] proposes that the Germanic prenominal genitive is structurally identical to the English Saxon genitive. The well-known restrictions on the prenominal genitive in German (that, for example, coordinated structures cannot appear) are not due to a different syntax from the superficially similar English construction, but to a lexical idiosyncracy of the German *'s* counterpart, which prohibits DPs with more than a small amount of pronounced structure (predominantly proper names and pronouns) to appear in its specifier. This is as shown in figure 1 (from [17]).

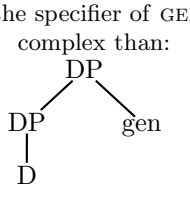

Fig. 1. Complexity filter associated with the German prenominal genitive

Importantly, all of the postulated complexity filters can be stated in terms of MSO logic over trees;[6] expressing that the length from the root to the most deeply embedded overt node is at most n is a simple universally quantified sentence to the effect that all nodes which are overt have depth less than or

[6] This is somewhat overkill, when compared to the attested complexity filters. The important thing is that the linguist appealing to complexity filters may propose *any* recognizable constraint, secure in the knowledge that he is not increasing the strong generative capacity of minimalist grammars.

equal to n. Conversely, *requiring* that a tree have a pronounced node deeper than some depth n is the negation of the previous statement.

To see that koopmanian complexity filters are admissible in the minimalist grammar formalism (in the sense that any MG with complexity filters may be reformulated as one with the same strong generative capacity without such), it is important to note first that these filters are not quite filters on the surface tree. As is indicated in figure 1, a complexity filter applies a regular filter to the surface trees in a particular geometric relationship to *a particular lexical item*, not just any head with a particular phonological form.

Toward a formalization, let $L_1, \ldots, L_n$ be the n complexity filters used in a minimalist grammar G. We will take each L_i $1 \leq i \leq n$ to define a regular tree language over $T(S)$, and will allow each selector and licensor feature of every lexical item to be associated with one of these filters. During a derivation, we check that the filters (if any) associated with a particular feature are respected. Below are presented two of the additional generating functions needed to take complexity filters associated with features into account.

$$\frac{\langle s_1, {=}c^L\gamma\rangle, \phi_1, \ldots, \phi_m \qquad \langle s_2, c\rangle, \psi_1, \ldots, \psi_n \qquad s_2 \in L}{\langle \bullet(s_2, s_1), \gamma\rangle, \phi_1, \ldots, \phi_m, \psi_1, \ldots, \psi_n} \; merge_1^{Filter}$$

$$\frac{\langle s_1, {+}c^L\gamma\rangle, \phi_1, \ldots, \phi_{i-1}, \langle s_2, {-}c\delta\rangle, \phi_{i+1}, \ldots, \phi_m \qquad \mathtt{t} \in L}{\langle \bullet(\mathtt{t}, s_1), \gamma\rangle, \phi_1, \ldots, \phi_{i-1}, \langle s_2, \delta\rangle, \phi_{i+1}, \ldots, \phi_m} \; move_2^{Filter}$$

To recast this in 'standard' minimalist grammars, we will take a new lexicon $Lex^{Filter} \subseteq \langle Lex\rangle_Q$, where $Q := Q_1 \times \cdots \times Q_n \cup \{\mathbf{1}\}$ is the state set of the disjoint union of the product automaton $A_1 \otimes \cdots \otimes A_n$ of the automata A_i such that $A_i^{-1}(r_i) = L_i$, $1 \leq i \leq n$, and the 'unit' automaton $A_{\mathbf{1}}$ with one state recognizing all of $T(S)$. We require that each A_i is *total*, in that every δ_f^i is defined on every input. We want the feature annotations to respect the transitions of A, and so we are interested in the smallest subset X of $Lex^A \subseteq \langle Lex\rangle_Q$ meeting the following conditions, which we will call Lex^{Filter}:

1. if $\ell \in Lex$, then X contains every $\ell' \in Lex^A$ such that for n the number of selector and licensor features of ℓ, and for $1 \leq i \leq n$,
   - if the i^{th} feature of ℓ is associated with complexity filter L_j, then the i^{th} feature of ℓ' is annotated with some state $q \in Q_1 \times \cdots \times Q_{j-1} \times \{r_j\} \times Q_{j+1} \times \cdots \times Q_k$
   - if the i^{th} feature of ℓ is not associated with any complexity filter, then the i^{th} feature of ℓ' is annotated with some state $q \in Q - \{\mathbf{1}\}$
2. for each $\ell' \in X$ with no licensee features, X contains the lexical item ℓ'', which is identical to ℓ' except that its selectee feature is annotated with the state $\mathbf{1}$.

Crucially, the state annotation $\mathbf{1}$ does not appear on any selector or licensor feature. Note also that because we are drawing lexical items from Lex^A, the last feature of a lexical item is annotated with a state which accurately reflects its structure.

Proposition 11. *For G a minimalist grammar with complexity filters, and G^{Filter} the minimalist grammar with lexicon Lex^{Filter}, for every selectee feature x it holds that $L_x(G) = L_{x^1}(G^{Filter})$.*

The proof is similar to that of proposition 8 and is suppressed.

3.3 Distributed Morphology

Distributed Morphology is a theory of the relation between a surface tree and its yield developed in [18]. There are two particularly salient aspects of this theory that are relevant here. First, the surface tree may be deformed prior to computing its yield. These deformations take the following form (from [18]):

morphological merger: a head X may be lowered to the head Y of its complement, forming the complex $\bullet(X, Y)$
fusion: a node $\bullet(X, Y)$, where X, Y are heads, may be replaced by a new node Z
fission: a head Z may be replaced by the node $\bullet(X, Y)$
impoverishment: A head X may be replaced by a head Y
morpheme addition: A term $t \in L$ may be replaced by $\bullet(t, X)$

Clearly, each of the above operations is a regular tree transduction, for any particular selection of heads X, Y, Z, and regular tree language L.

The second salient aspect of the theory is that 'lexical insertion takes place *during* spell-out'. This means (essentially) that the lexicon is a function from Σ to feature sequences (i.e. each lexical item is associated with a unique name); we might as well identify Σ with an initial subset of the natural numbers.[7] The surface tree is then a hierarchical arrangement of natural numbers. Each natural number can be realized as one of a finite number of strings over Δ^*, depending on which of a finite number of regular contexts it occurs in. Thus, each lexical item is associated with an MSO-definable transduction that 'realizes' it.

A distributed morphology grammar consists of a minimalist grammar G, along with a finite sequence of transductions $\phi_1, \ldots, \phi_n$, where ϕ_i is either a lexical insertion transduction, or a transduction of one of the five types given above. A string $s \in \Delta^*$ belongs to $L_x(G, \boldsymbol{\phi})$ (the language of a grammar $G, \phi_1, \ldots, \phi_n$ at category x) iff s is the yield of $\phi_n \circ \cdots \circ \phi_1(t)$, for some $t \in L_x(G)$. As $L_x(G)$ is the image of a regular set ($D_x(G)$) under a (direction preserving) MSO-definable transduction [19], we have that $L_x(G, \boldsymbol{\phi})$ is a multiple context-free language for any G, $\boldsymbol{\phi}$, and x [20]. Moreover, as by proposition 10 we have that for any $s \in \Delta^*$, $(\phi_n \circ \cdots \circ \phi_1)^{-1}(s)$ is a regular tree language, we can directly define a minimalist grammar defining exactly the set $L_x(G) \cap (\phi_n \circ \cdots \circ \phi_1)^{-1}(s)$ by proposition 8, making in principle available a parsing-as-intersection view of recognition and parsing in distributed morphology.

[7] In 'reality', one associates with each lexical item a (non-recursive) attribute value matrix, and with a subset of the lexical items (the contentful lexical items) a unique name (DOG, CAT, etc). The operation of fusion is the unification of AVMs, fission is splitting one AVM into two, and impoverishment is the removal of certain attribute-value pairs from an AVM. This is not particularly important for the present purposes.

4 Conclusion

We have shown that both derived and derivation tree languages of minimalist grammars are closed under intersection with recognizable sets of trees. This is non-trivial, as neither the derivation tree languages nor the derived tree languages constitute a well-known family of languages (the derivation tree languages are a proper subset of the recognizable sets, and the derived tree languages are a proper subset of the multiple regular tree languages).

These closure results are immediately translatable into results of direct linguistic interest. We can view the linguistic results in terms of the admissibility of a certain kind of grammar factorization. This is clearest in the case of the example of semantic types. We have shown there to be no impact on strong generative capacity whether we treat semantic type information syntactically, giving us a direct characterization of the semantically well-typed grammatical sentences, or semantically, characterizing the semantically well-typed grammatical sentences in terms of a filter. Whereas it is common consensus that we need to perform the work of semantic type-checking *somewhere* in the natural language pipeline, the complexity filters of Koopman are more controversial. By showing them to be merely notational devices, we have demonstrated that any rational debate about their adoption into minimalist grammars must revolve around something other than syntax, or semantics (perhaps learning).

Courcelle's results give us access to a weak generative capacity preseving array of extensions to a basic theory, and in conjunction with the closure under intersection results, access to an in-principle parsing method. As however the size of an automaton recognizing a tree language is not bounded by any elementary function in the size of the smallest MSO formula defining that language, and the size of the minimalist grammar recognizing the intersection language is directly related the number of states of the automaton, the size of the grammar recognizing the intersection is potentially much larger than the original grammar.[8]

References

1. Stabler, E.P.: Derivational minimalism. In: Retoré, C. (ed.) LACL 1996. LNCS (LNAI), vol. 1328, pp. 68–95. Springer, Heidelberg (1997)
2. Thatcher, J.W.: Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. Journal of Computer and System Sciences 1(4), 317–322 (1967)
3. Koopman, H.: On restricting recursion and the form of syntactic representations. In: Speas, P., Roeper, T. (eds.) Recursion. Oxford University Press, Oxford (to appear)
4. Heim, I., Kratzer, A.: Semantics in Generative Grammar. Blackwell Publishers, Malden (1998)
5. 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)

[8] See [21] for some relevant discussion.

6. Michaelis, J.: On Formal Properties of Minimalist Grammars. PhD thesis, Universität Potsdam (2001)
7. Raoult, J.C.: Rational tree relations. Bulletin of the Belgian Mathematical Society 4, 149–176 (1997)
8. Courcelle, B.: Monadic second-order definable graph transductions: a survey. Theoretical Computer Science 126, 53–75 (1994)
9. Kobele, G.M.: Formalizing mirror theory. Grammars 5(3), 177–221 (2002)
10. Engelfriet, J.: Context-free graph grammars. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages. Beyond Words, vol. 3, pp. 125–213. Springer, Heidelberg (1997)
11. Lang, B.: Recognition can be harder than parsing. Computational Intelligence 10(4), 486–494 (1994)
12. Graf, T.: A tree transducer model of reference-set computation. UCLA Working Papers in Linguistics 15, 1–53 (2010)
13. Chomsky, N.: Aspects of the Theory of Syntax. MIT Press, Cambridge (1965)
14. Kobele, G.M.: Generating Copies: An investigation into structural identity in language and grammar. PhD thesis, University of California, Los Angeles (2006)
15. Koopman, H., Szabolcsi, A.: Verbal Complexes. MIT Press, Cambridge (2000)
16. Kayne, R.: The Antisymmetry of Syntax. MIT Press, Cambridge (1994)
17. Koopman, H.: Derivations and complexity filters. In: Alexiadou, A., Anagnostopoulou, E., Barbiers, S., Gärtner, H.M. (eds.) Dimensions of Movement: From features to remnants. Linguistik Aktuell/Linguistics Today, vol. 48, pp. 151–188. John Benjamins, Amsterdam (2002)
18. Halle, M., Marantz, A.: Distributed morphology and the pieces of inflection. In: Hale, K., Keyser, S.J. (eds.) The View from Building 20, pp. 111–176. MIT Press, Cambridge (1993)
19. Mönnich, U.: Minimalist syntax, multiple regular tree grammars and direction preserving tree transductions. In: Rogers, J., Kepser, S. (eds.) Proceedings of the Workshop Model-Theoretic Syntax at 10, ESSLLI 2007, Dublin (2007)
20. Engelfriet, J., Heyker, L.: The string generating power of context-free hypergraph grammars. Journal of Computer and System Sciences 43, 328–360 (1991)
21. Morawietz, F., Cornell, T.: The MSO logic-automaton connection in linguistics. In: Lecomte, A., Lamarche, F., Perrier, G. (eds.) LACL 1997. LNCS (LNAI), vol. 1582, pp. 112–131. Springer, Heidelberg (1999)