Closure Properties of Minimalist Derivation Tree Languages

Thomas Graf

Department of Linguistics
University of California, Los Angeles
tgraf@ucla.edu
http://tgraf.bol.ucla.edu

Abstract. Recently, the question has been raised whether the derivation tree languages of Minimalist grammars (MGs; [14, 16]) are closed under intersection with regular tree languages [4, 5]. Using a variation of a proof technique devised by Thatcher [17], I show that even though closure under intersection does not obtain, it holds for every MG and regular tree language that their intersection is identical to the derivation tree language of some MG *modulo* category labels. It immediately follows that the same closure property holds with respect to union, relative complement, and certain kinds of linear transductions. Moreover, enriching MGs with the ability to put regular constraints on the shape of their derivation trees does not increase the formalism's weak generative capacity. This makes it straightforward to implement numerous linguistically motivated constraints on the Move operation.

Keywords: Minimalist Grammars, Derivation Tree Languages, Closure Properties, Regular Tree Languages, Derivational Constraints

Introduction

Minimalist grammars (MGs) were introduced in [14] as a formalism inspired by Chomsky's Minimalist Program [1]. Over the years, MGs have been enriched with various tools from the syntactic literature (e.g. phases and persistent features [15]), most of which do not increase the framework's weak generative capacity. One recent extension was proposed in my own work ([4, 5]; I will refer to these papers in the third person): the addition of reference-set constraints, which introduce a notion of optimality to the system. Graf proposes to model these constraints by linear tree transductions mapping the set of derivation trees of an MG to its subset of optimal derivation trees. He concludes that while the specific constraints he implements do not increase the weak generative capacity of MGs, the result carries over to arbitrary reference-set constraints definable by linear tree transductions only if the class of derivation tree languages is closed under intersection with regular tree languages — an open problem.

In this paper, I show that even though Minimalist derivation tree languages are not closed under intersection with regular tree languages, it holds for every Minimalist derivation tree language and regular tree language that their

Published version appeared in
S. Pogodalla and J.-P. Prost (Eds.), LACL 2011, LNAI 6736, pp. 96–111, 2011

Page breaks are NOT identical!

intersection is a projection of a Minimalist derivation tree language. I call this property *p-closure* under intersection with regular tree languages. While this p-closure result is already sufficient for Graf's purposes, it turns out that Minimalist derivation tree languages enjoy several more p-closure properties that make them appear very similar to regular tree languages. The p-closure properties of MGs also entail that their weak generative capacity is unaffected by the addition of regular control, which proves useful in the implementation of syntactic constraints such as islandhood, phases and Relativized Minimality.

The paper is laid out as follows: The preliminaries section covers, besides the definition of MGs, mundane topics such as tree languages and tree automata and the new yet crucial notion of p-closure. I then proceed to define Minimalist derivation tree languages and establish some of their basic properties (Sec. 2). This is followed up by a detailed investigation of their p-closure properties in Sec. 3, the greatest part of which is devoted to showing p-closure under intersection with regular tree languages. In the last part of this paper, I define MGs with regular control as a formalism with the same weak generative capacity as standard MGs, and I sketch some potential linguistic applications.

1 Preliminaries and Notation

In this section, I briefly introduce tree languages, tree automata, Minimalist grammars, and the notion of p-closure, which will be of great importance in this paper. As usual, $\mathbb{N}$ denotes the set of non-negative integers. A *tree domain* is a finite subset D of $\mathbb{N}^*$ such that, for $w \in \mathbb{N}^*$ and $j \in \mathbb{N}$, $wj \in D$ implies both $w \in D$ and $wi \in D$ for all $i < j$. Every $n \in D$ is called a *node*. Given nodes $m, n \in D$, m *immediately dominates* n iff $n = mi$, $i \in \mathbb{N}$. In this case we also say m is the mother of n, or conversely, n is a daughter of m. The transitive closure of the immediate dominance relation is called *dominance*. A node that does not dominate any other nodes is a *leaf*, and the unique node that isn't dominated by any nodes is called the *root*.

Now let Σ be a *ranked* alphabet, i.e. every $\sigma \in \Sigma$ has a unique non-negative *rank* (*arity*); $\Sigma^{(n)}$ is the set of all n-ary symbols in Σ. A Σ-tree is a pair $T := \langle D, \ell \rangle$, where D is a tree domain and $\ell : D \to \Sigma$ is a function assigning each node n a *label* drawn from Σ such that $\ell(n) \in \Sigma^{(d)}$ iff n has d daughters. Usually the alphabet will not be indicated in writing when it is irrelevant or can be inferred from the context. Sometimes trees will be given in functional notation such that $f(t_1, \ldots, t_n)$ is the tree where the root node is labeled f and immediately dominates trees $t_1, \ldots, t_n$. I denote by T_Σ the set of all trees such that for $n \geq 0$, $f(t_1, \ldots, t_n)$ is in T_Σ iff $f \in \Sigma^{(n)}$ and $t_i \in T_\Sigma$, $1 \leq i \leq n$. A *tree language* is some subset of T_Σ. It is *regular* (a *recognizable set*) iff it is recognized by a deterministic bottom-up tree automaton.

Definition 1. *A* deterministic bottom-up tree automaton *is a* 4*-tuple* $A := \langle \Sigma, Q, F, \delta \rangle$*, where*

- Σ *is a ranked alphabet,*

- Q *is a finite set of states (i.e. of unary symbols* $q \notin \Sigma$*),*
- $F \subseteq Q$ *is the set of final states,*
- $\delta\colon \left(\bigcup_{n\geq 0} Q^n \times \Sigma^{(n)}\right) \to Q$ *is the transition function.*

In the remainder of this paper, I will refer to deterministic bottom-up tree automata simply as (tree) automata. The transition function of a tree automaton can be extended to entire trees in the usual manner. A tree T, then, is *recognized* (*accepted*) by A iff $\delta(T) \in F$, and the language recognized by A is $L(A) := \{T \mid \delta(T) \in F\}$. At several points in the paper, I also mention tree transducers. All the reader needs to know about them is that they are the tree equivalent of string transducers, i.e. they rewrite input trees as output trees.

Given a tree T, a *treelet* t of T is a continuous substructure of T, that is to say, there is no node that does not belong to t yet both dominates a node of t and is dominated by a node of t. In the special case where t contains either all the nodes of T dominated by some node of t or none of them, we call t a *subtree* of T. A *projection* of a tree language is its image under a function $\hat{f}$ that is the pointwise extension of a surjective map $f : \Sigma \to \Omega$ between alphabets to tree languages. Thus projections are a particular kind of relabeling.

Definition 2 (P-Closure). *Given a class of languages* $\mathcal{L}$ *and an operation* O*,* $\mathcal{L}$ *is* p-closed *under* O *iff the result of applying* O *to some* $L \in \mathcal{L}$ *is a projection of some* $L' \in \mathcal{L}$*.*

We now turn to the definition of MGs, which mostly follows the chain-based exposition of [16] except that I allow for multiple final categories. This small extension has no effect on expressivity, as will be explained after the MG apparatus has been introduced.

Definition 3. *A* Minimalist grammar *is a 6-tuple* $G := \langle \Sigma, \mathit{Feat}, F, \mathit{Types}, \mathit{Lex}, \mathit{Op}\rangle$*, where*

- $\Sigma \neq \emptyset$ *is the alphabet,*
- *Feat is the union of a non-empty set base of* basic features *(also called* category features*) and its prefixed variants* $\{=f \mid f \in \mathit{base}\}$*,* $\{+f \mid f \in \mathit{base}\}$*,* $\{-f \mid f \in \mathit{base}\}$ *of* selector*,* licensor*, and* licensee *features, respectively,*
- $F \subseteq \mathit{base}$ *is a set of final categories,*
- $\mathit{Types} := \{::, :\}$ *distinguishes* lexical *from* derived *expressions,*
- *the lexicon Lex is a finite subset of* $\Sigma^* \times \{::\} \times \mathit{Feat}^*$*,*
- *and Op is the set of generating functions to be defined below.*

A chain *is a triple in* $\Sigma^* \times \mathit{Types} \times \mathit{Feat}^*$*, and* C *denotes the set of all chains (whence* $\mathit{Lex} \subset C$*). Non-empty sequences of chains will be referred to as* expressions*, the set of which is called* E*. I will usually drop the tuple brackets of chains and lexical items, but not those of expressions (the exception being depictions of derivation trees and the definitions of merge and move below).*

The set Op of generating functions consists of the operations merge and move. The operation merge$\colon (E \times E) \to E$ *is the union of the following three functions, for* $s, t \in \Sigma^*$*,* $\cdot \in \mathit{Types}$*,* $f \in \mathit{base}$*,* $\gamma \in \mathit{Feat}^*$*,* $\delta \in \mathit{Feat}^+$*, and chains* $\alpha_1, \ldots, \alpha_k$*,* $\iota_1, \ldots, \iota_k$*,* $0 \leq k, l$*:*

$$\frac{s :: =f\gamma \qquad t \cdot f, \iota_1, \ldots, \iota_k}{st : \gamma, \iota_1, \ldots, \iota_k}\ \textit{merge1}$$

$$\frac{s : =f\gamma, \alpha_1, \ldots, \alpha_k \qquad t \cdot f, \iota_1, \ldots, \iota_l}{ts : \gamma, \alpha_1, \ldots, \alpha_k, \iota_1, \ldots, \iota_l}\ \textit{merge2}$$

$$\frac{s \cdot =f\gamma, \alpha_1, \ldots, \alpha_k \qquad t \cdot f\delta, \iota_1, \ldots, \iota_l}{s : \gamma, \alpha_1, \ldots, \alpha_k, t : \delta, \iota_1, \ldots, \iota_l}\ \textit{merge3}$$

As the domains of all three functions are disjoint, their union is a function, too. Given one of the configurations above, one also says that s selects t.

The operation move: $E \to E$ *is the union of the two functions below, with the notation as above and the further assumption that all chains satisfy the Shortest Move Constraint (SMC), according to which no two chains in the domain of move display the same licensee feature* $-f$ *as their first feature.*

$$\frac{s : +f\gamma, \alpha_1, \ldots, \alpha_{i-1}, t : -f, \alpha_{i+1}, \ldots, \alpha_k}{ts : \gamma, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \alpha_k}\ \textit{move1}$$

$$\frac{s : +f\gamma, \alpha_1, \ldots, \alpha_{i-1}, t : -f\delta, \alpha_{i+1}, \ldots, \alpha_k}{s : \gamma, \alpha_1, \ldots, \alpha_{i-1}, t : \delta, \alpha_{i+1}, \ldots, \alpha_k}\ \textit{move2}$$

The language $L(G)$ *generated by* G *is the string component of the subset of the closure of the lexicon under the generating functions that contains all and only those expressions consisting of a single chain whose feature component is a single final category:* $L(G) := \{\sigma \mid \langle \sigma \cdot c \rangle \in \textit{closure}(\textit{Lex}, \textit{Op}), \cdot \in \textit{Types}, c \in F\}$.

Example 1. Let G be an MG defined by $F := \{c\}$ and the lexicon *Lex* below:

a :: a b :: =a =a +k a c :: =a c
a :: a −k

Two derivation trees of G are depicted in Fig. 1. □

As noted before, I slightly relax the MG formalism by allowing multiple final categories instead of just one. This is an innocent move. If a relaxed MG has final categories $c_1, \ldots, c_n$, we can turn it into a canonical MG by restricting the set of final categories to some new category c and introducing n new lexical items of the form $\varepsilon :: =c_i\ c$, where $1 \leq i \leq n$ and ε designates the empty string. The two grammars have virtually identical derivation tree languages with the only difference being the merger of a phonetically null c-head as the last step of every derivation for the canonical variant.

As for their weak generative capacity, MGs were shown in [6, 11, 12] to generate multiple context-free string languages, whence they constitute a mildly context-sensitive grammar formalism in the sense of [7]. The set of derivation trees of an MG, however, is a regular tree language and there is an effective procedure for obtaining the derived trees from their derivation trees — this holds even of the strictly more powerful class of MGs with unbounded copying [9, 10]. This is my main reason for considering derivation trees rather than derived trees (besides the central role of derivation trees in Graf's work): in contrast to the latter, they provide a unified, finite-state perspective on both MG variants; however, derived trees are investigated by Kobele in this volume, and he, too, proves closure under intersection with regular tree languages.

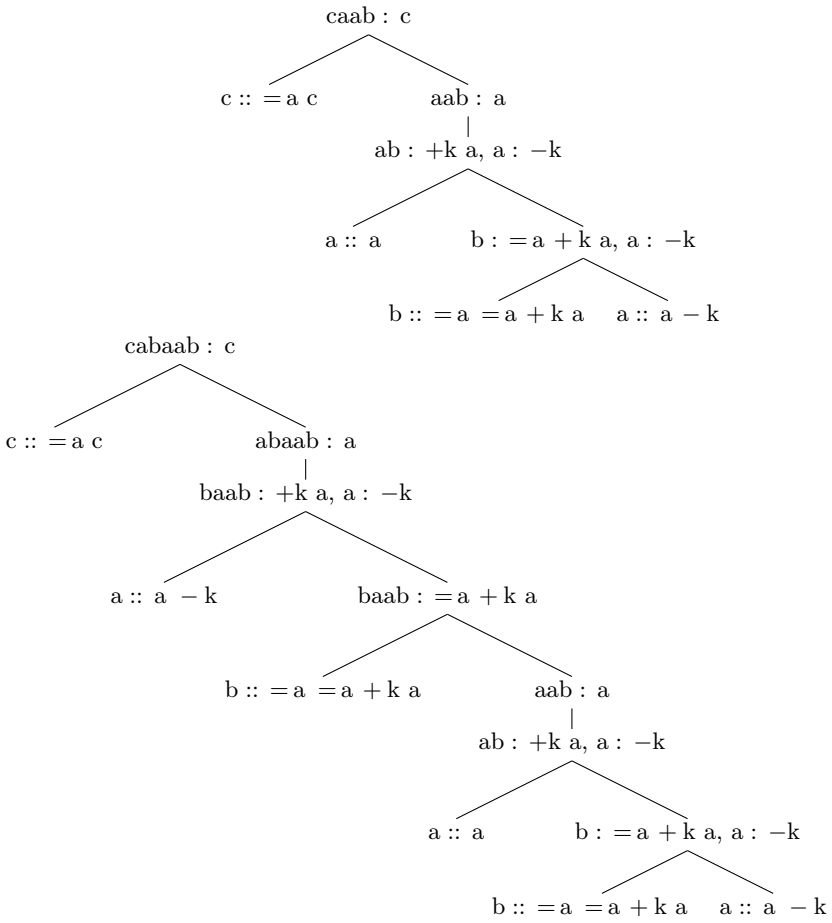

Fig. 1. Two derivation trees of the MG from example 1

2 Minimalist Derivation Tree Languages

The presentation of derivation trees in the MG literature varies somewhat with respect to what kind of labels are assigned to the interior nodes. The more common variant [cf. 16] is what I call string-annotated derivation trees, which were already encountered by the reader in the example at the end of the preliminaries section.

Definition 4 (String-annotated Derivation Tree Language). *Given an MG* $G := \langle \Sigma, Feat, F, Types, Lex, Op \rangle$*, its* string-annotated derivation tree language $\mathrm{sder}(G)$ *is the largest subset of* T_E *satisfying the following conditions:*

- *For every leaf node* n*,* $\ell(n) = \langle l \rangle$*,* $l \in Lex$*.*
- *For every binary branching node* n *immediately dominating* n' *and* n''*, it holds that* $merge(n', n'')$ *is defined and* $\ell(n) = merge(n', n'')$*.*

- *For every unary branching node n immediately dominating n', $move(n')$ is defined and $\ell(n) = move(n')$.*
- *For n the root node, $\ell(n) = \langle \sigma : c \rangle$, where $\sigma \in \Sigma^*$ and $c \in F$.*

String-annotated derivation tree languages aren't particularly interesting from the perspective of formal language theory as they are defined over an infinite alphabet ($L(G)$ is infinite in the general case, and so is E). Hence all work focusing on formal aspects of the derivation trees themselves [cf. 10] assume that the labels of the interior nodes indicate only which operations have taken place rather than the outputs of these operations at the respective stages of the derivations. This kind of derivation tree I refer to as Minimalist derivation tree.[1]

Definition 5 (Minimalist Derivation Tree Language). *Given an MG G, its* Minimalist derivation tree language $\mathrm{mder}(G)$ *is the set of trees obtained from* $\mathrm{sder}(G)$ *by the map μ relabeling all interior nodes by the corresponding operation:*

- $\mu(\langle l \rangle) = \langle l \rangle$, *where* $l \in Lex$
- $\mu(e(e_1, \ldots, e_n)) = op(\mu(e_1), \ldots, \mu(e_n))$, *where* $e, e_1, \ldots, e_n \in E$, $n \geq 1$, *and op is the unique operation in Op such that* $op(e_1, \ldots, e_n) = e$

Note that since the domains of all $op \in Op$ are pairwise disjoint, μ is indeed well-defined and a function. Also, whenever *merge* and *move* are used as labels, I will abbreviate them in the remainder of this paper by M and O, respectively.

It is fairly easy to see that Minimalist derivation tree languages are regular (consider the deterministic bottom-up tree automaton whose states correspond to the feature components of the labels of $\mathrm{sder}(G)$). In fact, they are a proper subset of the regular tree languages and are not closed under intersection with them.

Theorem 1. *The intersection of a Minimalist derivation tree language and a regular tree language may fail to be a Minimalist derivation tree language.*

Proof. Consider once again the Minimalist Grammar from example 1. Let $L_E \subseteq T_{Lex \cup \{M,O\}}$ contain all trees that have an even number of nodes, and only those. Then the result of intersecting the Minimalist derivation tree language of G with L_E is not a Minimalist derivation tree language. Among other things, it contains the Minimalist variant of the top tree in Fig. 1 on the facing page (i.e. with internal nodes replaced by M and O) but not the bottom one, yet they are built from the same lexical items and end in a final category, whence either both of them are in $closure(Lex, Op)$, or neither is. □

Corollary 1. *The image of a Minimalist derivation tree language under a linear transduction may fail to be a Minimalist derivation tree language, even if domain and co-domain of the transduction are identical.*

Proof. The intersection of two regular tree languages L and R is equivalent to the image of L under the diagonal of R, which is a linear transduction. □

[1] While it may not be in good style to have a technical term coalesce with a more colloquial one, the homophony is meant to highlight that I regard them as the canonical version of derivation trees for MGs.

3 P-Closure Properties

Theorem 1 and Cor. 1 notwithstanding, it does not take much ingenuity to realize that the counting condition in the example above can be enforced through the category and selection features of the feature calculus.

Example 2. Consider the lexical item $\mathrm{a} :: {=}\mathrm{a}\ {=}\mathrm{a} + \mathrm{k}\ \mathrm{a}$, and suppose that the derivation tree assembled so far contains an even number of nodes. The first selection feature causes the lexical item itself to be added to the derivation tree, plus the insertion of a Merge node. Thus the number of nodes in the derivation tree has increased by 2, which means that it is still even. Assume that at the next step a single lexical item is merged, increasing the count once more by 2. Then movement is triggered by the movement licensor feature, leading to the addition of only one node, so the tree now contains an odd number of nodes. At this point we only have to ensure that no lexical item of category c may be merged next in the derivation. The reason is that this would increase the counter only by 2, wherefore the number of nodes in the derivation tree would still be odd (and thus illicit), but nonetheless the derivation would be deemed well-formed, since we merged a lexical item of category c. In order to represent the arithmetic in the feature calculus, then, we have to replace $\mathrm{a} :: {=}\mathrm{a}\ {=}\mathrm{a} + \mathrm{k}\ \mathrm{a}$ by $\mathrm{a} :: {=}\mathrm{a}_\mathrm{e}\ {=}\mathrm{a}_\mathrm{o} + \mathrm{k}\ \mathrm{a}_\mathrm{o}$ and $\mathrm{c} :: {=}\mathrm{a}\ \mathrm{c}$ by $\mathrm{c} :: {=}\mathrm{a}_\mathrm{e}\ \mathrm{c}$. The fully refined grammar is given below.

$$
\begin{array}{lll}
\mathrm{a} :: \mathrm{a}_\mathrm{o} & \mathrm{b} :: {=}\mathrm{a}_\mathrm{o}\ {=}\mathrm{a}_\mathrm{o} + \mathrm{k}\ \mathrm{a}_\mathrm{e} & \mathrm{c} :: {=}\mathrm{a}_\mathrm{e}\ \mathrm{c} \\
\mathrm{a} :: \mathrm{a}_\mathrm{o}\ - \mathrm{k} & \mathrm{b} :: {=}\mathrm{a}_\mathrm{o}\ {=}\mathrm{a}_\mathrm{e} + \mathrm{k}\ \mathrm{a}_\mathrm{o} & \\
 & \mathrm{b} :: {=}\mathrm{a}_\mathrm{e}\ {=}\mathrm{a}_\mathrm{o} + \mathrm{k}\ \mathrm{a}_\mathrm{o} & \\
 & \mathrm{b} :: {=}\mathrm{a}_\mathrm{e}\ {=}\mathrm{a}_\mathrm{e} + \mathrm{k}\ \mathrm{a}_\mathrm{e} &
\end{array}
$$

□

The strategy in the example above can easily be generalized to intersection with arbitrary regular sets by a slight modification of the technique employed by Thatcher in [17]. Thatcher realized that we may view the states of a tree automaton as an alphabet which the automaton "adds" to the original node labels. Therefore, one can simplify any recognizable set R over alphabet Σ to a degree where it can be described as the derivation tree language of a context-free grammar (ignoring the distinction between terminals and non-terminals), even though the class of the latter is properly included in the class of the former. One does so by first subscripting the symbols in Σ with the states of the canonical automaton accepting R, and subsequently recasting the transition rules in terms of rewriting rules — a transition $\sigma(q_1, \ldots, q_n) \to q$ corresponds to the set $\{\sigma_q \to \tau_{1,q_1}, \ldots, \tau_{n,q_n} \mid \sigma(\tau_1, \ldots, \tau_n)$ is a subtree of some tree of R and each τ_i is assigned state q_i in some $T \in R$, $1 \leq i \leq n\}$. Thatcher's strategy, however, cannot be used if the alphabet of our trees is fixed, as is the case with Minimalist derivation trees. Internal nodes have to be labeled by M or O, and adding subscripts to these symbols takes us out of the class of Minimalist derivation trees. Crucially, though, the internal nodes of a derivation tree are tied to the leaf nodes in a very peculiar way: every internal node denotes an operation,

and this operation has to be triggered by a feature of some lexical item. So while we may not suffix the states of an automaton to the internal node labels of a Minimalist derivation tree, we can still make them explicit by incorporating them into the feature calculus.

The first step in sharpening this rough sketch is the introduction of *slices*.

Definition 6 (Slice). *Given a Minimalist derivation tree $T := \langle D, \ell \rangle$ and lexical item l occurring in T, the* slice of l *is the pair* $\text{slice}(l) := \langle S, \ell \rangle$, $S \subseteq D$, *defined as follows:*

- $l \in S$,
- *if node $n \in D$ immediately dominates a node $s \in S$, then $n \in S$ iff the operation denoted by $\ell(n)$ erased a selector or licensor feature on l.*

The unique $n \in S$ that isn't dominated by any $n' \in S$ is called the slice root *of l.*

Intuitively, $\text{slice}(l)$ marks the subpart of the derivation that l has control over by virtue of its selector and licensor features. The next two lemmas jointly establish that for any derivation tree T, the set $\{\text{slice}(l) \mid l \text{ a lexical item in } T\}$ is a partition of T.

Lemma 1. *For every Minimalist derivation tree $T := \langle D, \ell \rangle$ and lexical item l in T,* $\text{slice}(l) := \langle S, \ell \rangle$ *is a unary branching treelet.*

Proof. That $\text{slice}(l)$ is unary branching follows immediately from the definition. So we only have to show that there is no node $n \notin S$ that both dominates and is dominated by nodes in $\text{slice}(l)$. Since the selector and licensor features of a lexical item l cannot be manipulated by any $o \in Op$ after l has already been selected by another lexical item, all these features of l have to occur before its category feature (which is unique). It is also clear that every licensee feature has to follow the category feature (*move* must be triggered by a licensor feature on some lexical item that is higher than l in the derivation tree, and only the category feature allows l to be selected by another lexical item so that the derivation can reach this higher point). Thus it holds for every lexical item that its feature sequence is an element of $\{=f, +f \mid f \in \mathit{base}\}^* \times \mathit{base} \times \{-f \mid f \in \mathit{base}\}^*$, proving the claim above. □

Lemma 2. *Given a Minimalist derivation tree T, every node of T belongs to some slice.*

Proof. Trivial. □

With these basic facts established, we turn to the algorithm that upon being given an MG $G := \langle \Sigma, \mathit{Feat}_G, F_G, \mathit{Types}, \mathit{Lex}_G, \mathit{Op} \rangle$ and regular language R will compute an MG G' such that $\text{mder}(G) \cap R$ is a projection of $\text{mder}(G')$. We begin by constructing the canonical automaton A_R for R (note that A_R is deterministic). In the next step, we pick a tree $T \in R \cap \text{mder}(G)$ and suffix each node n of T with the state q that A_R assigns to n when recognizing T (since A_R is deterministic, q is unique). After the second step has been applied to all members

of $R \cap \mathrm{mder}(G)$, the result will be a set with $R \cap \mathrm{mder}(G)$ as its projection. So far, then, our procedure does not deviate at all from Thatcher's.

But now we have to move the state subscripts from the internal nodes into the lexical items. We do so again in two steps (before proceeding any further, though, the reader may want to take a look at the simplified example in Fig. 2 to get a better intuition for the procedure). For the first step, we look at each

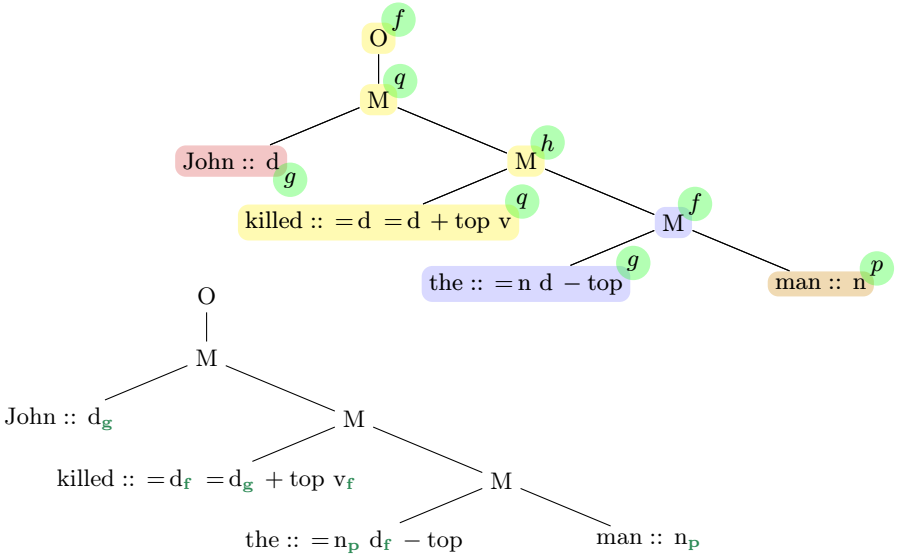

Fig. 2. Derivation tree of MG G with states of A_R (top), and corresponding derivation tree of G' with refined categories (bottom)

lexical item and add the subscripted state of its slice root to its category feature — keep in mind that a lexical item has exactly one such feature. In the second step, we have to refine the selection features accordingly. Given the definition of slices, it is easy to see that the $(n+1)^{\text{th}}$ node o of slice(l) (counting from l toward the slice root) denotes the operation that erased the n^{th} feature f of l, $n \geq 1$. If $\ell(o) = M$ (in which case f is a selector feature), determine the category c_q of the lexical item l' whose slice root is immediately dominated by o and replace f by $=c_q$. Repeat this procedure for all selector features of all lexical items in all trees.[2] Call the set of these lexical items with refined category and selector features $Lex_{G'}$ (which is still finite due to the restriction of A_R to finitely many states). The desired MG is $G' := \langle \Sigma, Feat_{G'}, F_{G'}, Types, Lex_{G'}, Op \rangle$, where

[2] Two things are worth mentioning here. First, there seems to be no way around restricting the selector features of lexical items, as is witnessed by example 2, where a lexical item of category a_o may not select two lexical items of the same category, and one of category a_e may not select two with differing categories. Second, it suffices for the construction to consider but a finite number of configurations, so a computational

- $Feat_{G'} := \{f_q, =f_q \mid f \in base_G,\ q \text{ a state of } A_R\} \cup \{-f, +f \mid f \in base_G\}$, and
- $F_{G'} := \{c_q \mid c \in F_G,\ q \text{ a final state of } A_R\}$

Lemma 1 and 2 jointly guarantee that the procedure above is well-defined. In order to prove its correctness, though, we first need one more property of slices.

Lemma 3. *Let $T := \langle D, \ell\rangle$ be a derivation tree, $\mathbb{S} := \{\text{slice}(l) \mid l = \ell(n)$ for some leaf $n \in D\}$, and $\mathbb{S} \upharpoonright \text{slice}(l) := \{\text{slice}(l') \in \mathbb{S} \mid l'$ was selected by $l\}$. Let $\overrightarrow{s} := \langle s_1, \ldots, s_n\rangle$ be a sequence of $s \in \mathbb{S}$ such that*

- $s_1 \in \mathbb{S}$
- *for all* $1 \leq i < n$, $s_{i+1} \in \mathbb{S} \upharpoonright s_i$
- $\mathbb{S} \upharpoonright s_n = \emptyset$

For every Minimalist derivation tree, there is at least one such $\overrightarrow{s}$, and for every choice of $\overrightarrow{s}$, $s_n := \langle S, \ell\rangle$ is a slice with $|S| = 1$.

Proof. The first half of the claim is trivial. As for the second one, if $\mathbb{S} \upharpoonright s_n$ is empty, the lexical item l such that $s_n := \text{slice}(l)$ has no selector features. But then it has no licensor features either, because these must precede the category feature, which is the only way l has left to enter the derivation. □

With this unsurprising yet important fact established, we finally turn to the correctness of the procedure, splitting the claim into two lemmas for the sake of readability. Note that I use π to denote the projection that strips away the state suffixes and thus turns $Lex_{G'}$ into Lex_G again.

Lemma 4. $\pi(\text{mder}(G')) \subseteq R \cap \text{mder}(G)$

Proof. Assume towards a contradiction $\pi(\text{mder}(G')) \not\subseteq R \cap \text{mder}(G)$. Then there has to be some tree $T \in \pi(\text{mder}(G'))$ such that $T \notin R \cap \text{mder}(G)$. But $\text{mder}(G)$ cannot be a proper subset of $\pi(\text{mder}(G'))$, so it has to be the case that $\text{mder}(G) \ni T \notin R$. Thus, when recognizing T, A_R assigns the root of T some non-final state q'. Suppose w.l.o.g. that the root node of T belongs to $\text{slice}(l)$ for some lexical item l of final category c_q. According to our procedure, c_q is a final category iff c is a final category of G and q is a final state of A_R. So if $T \in \pi(\text{mder}(G'))$, there has to be some $T' \in \text{mder}(G) \cap R$ such that in both trees the root node is also the root of $\text{slice}(l)$ (otherwise A_R never reached a final state in the slice root of l, whence l isn't of category c_q, a contradiction). Now since A_R is deterministic, the only way for it not to reach state q at the slice root of l in T is if the state it assigns to the slice root of some lexical item l' selected by l differs from the subscript of the corresponding selector feature of l. But the same reasoning applies to l' as well, so that we progress further down the tree until we encounter a lexical item that selects a lexical item l'' whose slice is of size 1 (by Lem. 3). So the automaton must have assigned the slice root of l'' a state different from the subscript of the category feature of l''. But A_R is bottom-up and deterministic, wherefore it always assigns the same state to l''. Contradiction. □

implementation of the procedure is still feasible. This follows from the finiteness of the lexicon and the index of the Nerode partition induced by the automaton.

Lemma 5. $\pi(\text{mder}(G')) \supseteq R \cap \text{mder}(G)$

Proof. Assume once again towards a contradiction $\pi(\text{mder}(G')) \not\supseteq R \cap \text{mder}(G)$. Then there has to be some tree $T \in R \cap \text{mder}(G)$ such that $T \notin \pi(\text{mder}(G'))$. Since the lexicons of G and G' are identical modulo state-subscripts, the only option is that A_R and G' disagree with respect to states, at which point the reasoning of the previous proof applies unaltered. □

Theorem 2. *The class of Minimalist derivation tree languages over Σ, Feat is p-closed under intersection with regular tree languages.*

Corollary 2. *The class of Minimalist derivation tree languages over Σ, Feat is p-closed under intersection and relative complement.*

Proof. Since every Minimalist derivation tree language is regular, p-closure under intersection follows immediately from Thm. 2. Given two Minimalist derivation tree languages L and M, $R := L - M$ is a regular language, so $L - M = L \cap R$ is a projection of some Minimalist derivation tree language. □

Note that p-closure under relative complement does not imply p-closure under complement with respect to the class of Minimalist derivation tree languages over Σ, *Feat*, as for each grammar over this signature there exists another grammar whose derivation tree language is a proper superset of the former's. However, when we restrict our attention to the class of all MG whose lexicon is a subset of some finite set *Lex* over Σ, *Feat*, there will be one Minimalist derivation tree language that subsumes all others and p-closure under relative complement implies p-closure under complement as desired (which in turns implies p-closure under union).

Corollary 3. *Let Lex be some finite subset of $\Sigma^* \times \{::\} \times Feat^*$. Then the class $\{\text{mder}(G) \mid G \text{ an MG with } Lex_G \subseteq Lex\}$ is p-closed under complement and union.*

P-closure also extends to linear tree transductions whose co-domain is a Minimalist derivation tree language. This is of immediate relevance to Graf's tree transducer model of reference-set computation, because most reference-set constraints are conceived of as filters, that is to say, they map each Minimalist derivation tree language into a subset of itself.

Corollary 4. *Given a linear transduction τ with some Minimalist derivation tree language L as its co-domain, it holds for every regular tree language R that its image under τ is a projection of some Minimalist derivation tree language.*[3]

Proof. Follows from Thm. 2 and the fact that the range of a linear transduction applied to a regular tree language is regular. □

[3] My thanks go to an anonymous reviewer for pointing out that the corollary as it was originally stated was overly restrictive.

In connection with Graf's transducer approach, we also observe that the procedure as it is currently defined may lead to a significant (albeit still linear) blow-up in the size of the lexicon. The implication for Graf's work is that a grammar with reference-set constraints may be notably more succinct than one without them, even if both define the same tree languages *modulo* state subscripts. But since it might be the case that the procedure given here can still be improved upon, this has to remain a conjecture for now.

Conjecture 1. Given a lexicon *Lex* and $n \geq 0$, let $Lex^{(n)} := \{l \in Lex \mid l \text{ has exactly } n \text{ selector features}\}$. Now if there is no $m > k$ such that $Lex_G^{(m)} \neq \emptyset$, then in the worst case

$$|Lex_{G'}| = \sum_{i=0}^{k} \left(|Lex_G^{(i)}| \cdot |Q|^{i+1} \right)$$

4 Minimalist Grammars with Regular Control

P-closure under intersection also opens up new ways of incorporating constraints into MGs. Constraints have proven difficult to study in MGs, and their effects on the machinery are somewhat unpredictable; for instance, MGs with the SMC and the so-called Specifier Island Constraint (SPIC) are weaker than MGs that feature only the SMC, whereas MGs that lack the SMC yet have the SPIC generate type-0 languages [3]. But adopting the perspective of model-theoretic syntax [13], we may view constraints as defining formal languages. Thanks to Thm. 2, then, MGs can be augmented by any finite number of constraints defining regular tree languages without increasing their weak generative capacity. In fact, even the strong generative capacity of MGs is mostly unaffected, as the procedure outlined above only relies on refining category features, which — in the derived tree — surface only on the head of the highest phrase.

Definition 7 (MGs with Regular Control). *A* Minimalist Grammar with Regular Control *(MG^{RC}) is a 7-tuple $G := \langle \Sigma, Feat, F, Types, Lex, Op, \mathcal{R} \rangle$, where*

- *Σ, Feat, F, Types, Lex, and Op are defined as usual,*
- *and $\mathcal{R}$ is a finite set of regular tree languages.*

The language generated by G is the set $L(G) := \{\sigma \mid \langle \sigma \cdot c \rangle \in closure(Lex, Op), \cdot \in Types, c \in F,$ and $\langle \sigma \cdot c \rangle$ is the label of the root of some tree $T \in \mathrm{sder}(G)$ such that $\mu(T) \in \mathrm{mder}(G) \cap \bigcap_{R \in \mathcal{R}} R\}$.

Theorem 3. $\mathrm{MG} \equiv \mathrm{MG}^{\mathrm{RC}}$

Given the prominence of constraints in the syntactic literature, it is hardly surprising that there are numerous applications for regular control. The most obvious one are intervention conditions on movement such as the one illustrated in (1) below.

(1) a. Who$_i$ did John say that Bill adores t_i?

b. ?/* Who$_i$ did John doubt whether Bill adores t_i?

Without further precautions, a MG that derives (1a) will also derive (1b) as movement is restricted only by the feature calculus, not the shape of the phonological strings. A regular language can easily militate against such locality violations. Recall that the states of an automaton recognizing a Minimalist derivation tree can be taken to represent the feature components of the string-annotated derivation trees. In order to block (1b), then, one may proceed as follows. Let p and q be the states that the standard automaton would assign to *whether* and *Bill hates* t_i, respectively (we take these states to literally be sequences of feature sequences). Now introduce a new state p^{wh} that the automaton assigns to *whether* instead of p such that $p^{wh} \times q \times M$ is undefined only if q contains no sequence containing a movement licensee features, in which case $p^{wh} \times q \times M = p \times q \times M$.

It is easy to see that this strategy can be extended to intervention conditions in general, most of which require the automaton to check the shape of entire phrases rather than a single word. Two well-known examples are the Complex NP Constraint, which blocks extraction from a CP that is the complement of an NP (or rather, DP in contemporary analyses), and the Subject Constraint, which rules out movement originating from inside a DP in subject position.

(2) * Who$_i$ did John reject the claim that the lobbyists bribed t_i?

(3) a. Where$_i$ is it likely that John went t_i?

b. * Where$_i$ is that John went t_i likely?

A combination of both types of movement constraint is the *that*-trace effect: in general, a wh-word can be extracted out of a CP whose complementizer is *that*, but not if it is the subject of the clause.

(4) a. What$_i$ did you say that John ate t_i?

b. * Who$_i$ did you say that t_i ate my burrito?

Here the automaton has to be sensitive to both the nature of the complementizer and the structural properties of the domain from which the wh-word was extracted. The sensitivity to *that* is analogous to the *whether*-example, while the distinction between subjects and objects mirrors the domain condition of the Subject Constraint.

Further examples of linguistic locality constraints that can be captured this way are the Coordinate Structure Constraint, the Left Branch Condition, and phases (introduced in [2] and implemented for MGs in [15]). Many of the principles formalized in [13] can also be adapted for MGs, although the change from derived trees to derivation trees will require some slight revisions in certain cases, in particular binding and control, which rely on a notion of c-command that might prove tricky to capture on a derivational level.

Through the use of constraints we can also reduce the number of movement steps in our grammars. In early Minimalism [1], satisfying feature dependencies between non-adjacent phrases invariably required movement, an assumption inherited by MGs. In such a setup, subject-verb agreement, say, is assumed to

be an effect of the subject moving into the specifier of the TP and checking its person features. But other instances of agreement, e.g. between determiners or adjectives on the one hand and nouns on the other, are rather cumbersome to handle this way. This brought about a major revision of the feature calculus in order to make feature checking apply a distance in certain cases [2]. As long as we do not allow unbounded nesting and crossing of the checking paths defined by this operation, regular constraints can yield the same effect by associating every lexical item with "pseudo-features" that encode properties not pertinent to movement. For instance, the Icelandic adjective *rauðan* 'red', which is masculine, singular, accusative, and strongly inflected, could be assigned the corresponding pseudo-features, which in turn also have to be present on the noun the adjective combines with. A more interesting case is long-distance subject-verb agreement in English expletive constructions, as the phenomenon is noticeably more difficult to capture by manual refinement of categories.

(5) a. * There seems to John to be several men in the garden.
b. There seem to John to be several men in the garden.

But the gerrymandering of the feature calculus need not stop here. We may also employ regular constraints to incorporate a restricted version of pied-piping. Pied-piping refers to the phenomenon that a constituent containing some element with a movement licensee feature seems to be stuck to it for the purposes of movement.

(6) a. [Which famous linguist]$_i$ did Robert write a book [about t_i]?
b. [About which famous linguist]$_i$ did Robert write a book t_i?

In the syntactic literature this is often analyzed as the movement licensee feature of the DP percolating upwards into the PP. Unfortunately, enriching MGs with such a feature percolation mechanism allows them to generate any recursively enumerable language [8]. But at least for the example above, only a very limited kind of feature percolation is required: it is sufficient to allow both *about* and *which* to carry a movement licensee feature as long as we ensure that the variant of *about* with such a feature must merge with a DP such that the determiner of said DP does not carry the same feature, but could in principle (i.e. there is an entry in the lexicon with the same phonological string and the same category as the determiner that also carries the relevant feature). It is easy to see that this constraint can be enforced by regular means.

Dynamic restrictions on the distribution of features also allows us to work around certain shortcomings of the SMC. The SMC — albeit essential for keeping Minimalist derivation trees within the confines of regular tree languages — comes with its fair share of linguistic inadequacies, in particular with respect to wh-movement. Since English allows for wh-phrases to stay *in situ*, every wh-phrase in an MG must come in two variants, one with a wh-licensee feature, and one without it. But given this duality, nothing prevents superiority violations like the one in (7b) (for the sake of simplicity, only wh-movement is indicated by traces).

(7) a. Who$_i$ t_i prefers what?
b. * What$_i$ does who prefer?

The ungrammatical (7b) can be derived because *who* need not carry a wh-licensee feature, in which case the MG will treat it like any other DP. Consequently, nothing prevents *what* from carrying a wh-licensee feature, so the movement step is licit. Instances of overgeneration like this can be blocked if one takes a hint from our implementation of pied-piping: the automaton has to verify that no node along the movement path could potentially carry the same feature. In (7b) above, this condition is violated because *who* could be a carrier of the wh-licensee feature, and so the automaton rejects the derivation tree. Note that a similar strategy can be used if two identical features have to be distinguished due to the SMC yet at the same we want to capture specific locality restrictions they impose on each other (e.g. *who* being assigned feature wh_1 and *what* feature wh_2 in a multiple wh-movement language). Admittedly the notion "node along the movement path" has to be carefully worked out and may turn out to be rather complex in grammars with massive remnant movement. Overall, though, it seems that this approach goes a long way towards an MG implementation of Relativized Minimality as envisioned in [15], with the added benefit that the restrictions imposed at the level of derivation trees also carry over to the strictly more powerful mechanism of MGs with copying [9].

Conclusion

I defined Minimalist derivation tree languages and showed that they are p-closed under intersection with regular tree languages, intersection, complement, relative complement, union, and linear tree transductions whose co-domain is a Minimalist derivation tree language. From these closure properties it follows immediately that enriching MGs with regular control does not increase their weak generative capacity. The result has numerous linguistic applications, in particular regarding locality conditions on movement and reference-set constraints [cf. 4, 5].

Acknowledgments For their motivational comments and helpful criticism, I am greatly indebted to Ed Stabler, Ed Keenan, Jens Michaelis, Uwe Mönnich, the three anonymous reviewers, and the attendees of the UCLA Mathematical Linguistics Circle. The research reported herein was supported by a DOC-fellowship of the Austrian Academy of Sciences.

Bibliography

[1] Chomsky, N.: The Minimalist Program. MIT Press, Cambridge, Mass. (1995)
[2] Chomsky, N.: Derivation by phase. In: Kenstowicz, M.J. (ed.) Ken Hale: A Life in Language, pp. 1–52. MIT Press, Cambridge, Mass. (2001)

[3] Gärtner, H.M., Michaelis, J.: Some remarks on locality conditions and minimalist grammars. In: Sauerland, U., Gärtner, H.M. (eds.) Interfaces + Recursion = Language? Chomsky's Minimalism and the View from Syntax-Semantics, pp. 161–196. Mouton de Gruyter, Berlin (2007)
[4] Graf, T.: Reference-set constraints as linear tree transductions via controlled optimality systems. In: Proceedings of the 15$^{\text{th}}$ Conference on Formal Grammar (2010), to appear
[5] Graf, T.: A tree transducer model of reference-set computation. UCLA Working Papers in Linguistics 15, Article 4 (2010)
[6] Harkema, H.: A characterization of minimalist languages. In: de Groote, P., Morrill, G., Retoré, C. (eds.) Logical Aspects of Computational Linguistics (LACL'01), LNAI, vol. 2099, pp. 193–211. Springer, Berlin (2001)
[7] Joshi, A.: Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions? In: Dowty, D., Karttunen, L., Zwicky, A. (eds.) Natural Language Parsing, pp. 206–250. Cambridge University Press, Cambridge (1985)
[8] Kobele, G.M.: Features moving madly: A formal perspective on feature percolation in the minimalist program. Research on Language and Computation 3(4), 391–410 (2005)
[9] Kobele, G.M.: Generating Copies: An Investigation into Structural Identity in Language and Grammar. Ph.D. thesis, UCLA (2006)
[10] Kobele, G.M., Retoré, C., Salvati, S.: An automata-theoretic approach to minimalism. In: Rogers, J., Kepser, S. (eds.) Model Theoretic Syntax at 10. pp. 71–80 (2007)
[11] Michaelis, J.: Derivational minimalism is mildly context-sensitive. LNAI 2014, 179–198 (1998)
[12] Michaelis, J.: Transforming linear context-free rewriting systems into minimalist grammars. LNAI 2099, 228–244 (2001)
[13] Rogers, J.: A Descriptive Approach to Language-Theoretic Complexity. CSLI, Stanford (1998)
[14] Stabler, E.P.: Derivational minimalism. In: Retoré, C. (ed.) Logical Aspects of Computational Linguistics (LACL'96), pp. 68–95. Springer, Berlin (1997)
[15] Stabler, E.P.: Computational perspectives on minimalism. In: Boeckx, C. (ed.) Oxford Handbook of Linguistic Minimalism, pp. 617–643. Oxford University Press, Oxford (2011)
[16] Stabler, E.P., Keenan, E.: Structural similarity. Theoretical Computer Science 293, 345–363 (2003)
[17] Thatcher, J.W.: Characterizing derivation trees for context-free grammars through a generalization of finite automata theory. Journal of Computer and System Sciences 1, 317–322 (1967)