Movement-Generalized Minimalist Grammars

Thomas Graf

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

Abstract. A general framework is presented that allows for Minimalist grammars to use arbitrary movement operations under the proviso that they are all definable by monadic second-order formulas over derivation trees. Lowering, sidewards movement, and clustering, among others, are the result of instantiating the parameters of this framework in a certain way. Even though weak generative capacity is not increased, strong generative capacity may change depending on the available movement types. Notably, TAG-style tree adjunction can be emulated by a special type of lowering movement.

Keywords: Minimalist Grammars, Movement, Monadic Second-Order Logic, Tree Languages, Transductions, Tree Adjunction Grammar

Introduction

Ever since Joshi's conjecture that natural language is mildly context-sensitive [8], a lot of research has been devoted to characterizing this class in various ways. One of them pertains to multiple context-free languages (MCFLs; [18]) and states that they coincide with the string yield of the class of tree languages that are the image of regular tree languages under tree-to-tree transductions definable in monadic second-order logic (MSO; see [14] and the literature cited there). This result meshes well with recent approaches that decompose Minimalist grammars (MGs) — which have the same weak generative capacity as MCFLs — into an MSO-definable (= regular) tree language L and a transduction from L to the intended phrase structure trees ([12, 14] and references therein).

From a linguistic perspective, an MG's set of well-formed derivation trees provides the most natural encoding of this underlying tree language L, and [12] demonstrated that this is indeed a workable solution. In [4], however, it is shown that recognizing Minimalist derivation trees does not require the full power of MSO. In a sense, then, MGs still have some wiggle room insofar as one can increase the complexity of their derivation trees and still stay inside the confines of MSO-definability that limit the formalism to MCFLs. One way to exploit this gap is by adding MSO-definable constraints to MGs. Even though this greatly increases their linguistic usefulness, weak and strong generative capacity remain

Published version appeared in
D. Béchet and A. J. Dikovsky (Eds.), LACL 2012, LNCS 7351, pp. 58–73, 2012

Page breaks are NOT identical!

the same [3, 11]. I explore another option in this paper: allowing for derivationally more complex variants of Move, yielding Movement-Generalized MGs (MGMGs).

My endeavour starts with another insight of [4], namely that the distribution of Move nodes in a derivation tree can be regulated by a few simple constraints stated in terms of proper dominance. To create new movement types, for instance sidewards movement [7, 15], one merely has to replace proper dominance by some other binary relation R. As long as R is MSO-definable, the derivation tree language will still be regular and weak generative capacity does not increase. Some parameters of Move, though, must be expressed directly in the mapping from derivation trees to derived trees. Fortunately, all of them are MSO-definable and thus pose no risk of taking us out of the class of MCFLs. The result is a general, mildly context-sensitive framework that accommodates almost all aspects of Move: directionality (raising, lowering, sideward), size of the moved constituent (head, phrase, pied-piped phrase), overt versus covert, and linearization (left or right specifier).

The paper is laid out as follows. After a few technical preliminaries, I define standard MGs in Sec. 2, focusing foremost on the constraints that ensure the well-formedness of Minimalist derivation trees. I then proceed with generalizing Move; Sec. 3.2 stays at the level of derivation trees, whereas Sec. 3.3 and 3.4 are devoted to transduction parameters. The final definition of MGMGs is given in Sec. 3.5. In the last section, I analyze the relationship between TAGs and MGs with lowering, conjecturing that their derived tree languages are identical given certain assumptions.

1 Preliminaries and Notation

Let Σ and Γ be alphabets. A *directed graph* with labeled nodes and edges over (Σ, Γ) is a triple $G(\Sigma, \Gamma) := \langle V, E, \ell \rangle$, with V a finite set of nodes, $E \subseteq V \times \Gamma \times V$ the set of labeled edges, and $\ell : V \to \Sigma$ the node labeling function. An edge $\langle u, \gamma, v \rangle$ is an edge from u to v with label γ; it is an outgoing edge of u and an incoming edge of v. In this case, u is called a *mother of* v, or equivalently, v is a *daughter of* u. A *path* from u to v is a (possibly empty) sequence of nodes $u_0 \cdots u_n$ such that $u = u_0$, $v = u_n$ and u_i is a mother of u_{i+1} for all $0 \leq i \leq n$. A path is a *cycle* iff $u_0 = u_n$ and $n \geq 1$. A graph is *cycle-free* iff it contains no cycles. A node with no incoming edges is a *root*, a node with no outgoing edges a *leaf*. A graph is *rooted* iff it has exactly one root.

Let Σ be a *ranked* alphabet, i.e. every $\sigma \in \Sigma$ has a unique non-negative *rank*; $\Sigma^{(n)}$ is the set of all n-ary symbols in Σ. A *Σ-term graph* is a cycle-free rooted graph $G(\Sigma, \Gamma)$ such that Σ is a ranked alphabet, $\Gamma := \{i \mid 1 \leq i \leq n \text{ and } n \text{ the largest integer such that } \Sigma^{(n)} \neq \emptyset\}$ and every node with label $\sigma \in \Sigma$ of rank i has i outgoing edges with pairwise distinct labels. The integers on the outgoing edges of a node are interpreted as linear order. A *Σ-tree* is a Σ-term graph in which every node except the root has exactly one incoming edge. Let $\Pi^n := \{\square_i \mid 0 < i \leq n\}$ be a set of distinguished nullary symbols called ports.

A (Σ, n)-*context* is a $(\Sigma \cup \Pi^n)$-tree such that all ports have pairwise distinct indices. Given a (Σ, n)-context C and a sequence $s := t_1, \ldots, t_n$ of (Σ, m)-contexts, $m \in \mathbb{N}$, the *n-fold tree concatenation* of C and s replaces each $\square_i$ in C (if it exists) by t_i.

My definition of MSO transductions follows [1] very closely. I assume that the reader is already familiar with monadic second-order logic (MSO) and write $\mathrm{MSO}(\Sigma, \Gamma)$ to denote the MSO language of (Σ, Γ)-graphs. A *finite-copying MSO graph transducer* from (Σ_1, Γ_1) to (Σ_2, Γ_2) is a triple $\mathrm{MSO}_{\mathrm{gr}} := \langle C, \Psi, \Theta \rangle$, where C is a finite set of copy names, $\Psi := \{\psi_{\sigma,c}(x) \in \mathrm{MSO}(\Sigma_1, \Gamma_1) \mid \sigma \in \Sigma_2, c \in C\}$ a set of node formulas, and $\Theta := \{\theta_{\gamma,c,c'}(x, y) \in \mathrm{MSO}(\Sigma_1, \Gamma_1) \mid \gamma \in \Gamma_2, c, c' \in C\}$ a set of edge formulas.

The *graph transduction* τ *defined by* $\mathrm{MSO}_{\mathrm{gr}}$ is as follows. For every graph $G(\Sigma_1, \Gamma_1)$, its image under τ is $G'(\Sigma_2, \Gamma_2)$ such that

- $V_{G'} := \{\langle c, u \rangle \mid c \in C, u \in V_G,$ and $G, u \models \psi_{\sigma,c}(x)$ for exactly one $\sigma \in \Sigma_2\}$,
- $E_{G'} := \{\langle \langle c, u \rangle, \gamma, \langle c', u' \rangle \rangle \mid \langle c, u \rangle, \langle c', u' \rangle \in V_{G'}, \gamma \in \Gamma_2$ and $G, u, u' \models \theta_{\gamma,c,c'}(x, y)\}$,
- $\ell_{G'} := \{\langle \langle c, u \rangle, \sigma \rangle \mid \langle c, u \rangle \in V_{G'}, \sigma \in \Sigma_2,$ and $G, u \models \psi_{\sigma,c}(x)\}$.

An *MSO term graph transducer* is a graph transducer from trees to term graphs. An *MSO tree transducer* is a graph transducer from trees to trees. Unless a transducer is explicitly designated to be finite-copying, C is assumed to be a singleton and thus superfluous.

2 Minimalist Grammars

The material covered in this paper presupposes a high level of familiarity with MGs. Unfortunately, space restrictions force me to proceed at a brisk pace, so that readers unacquainted with the formalism must be referred to [21] for a gentle introduction.

While MGs are usually defined in terms of the derived trees they generate [21] or in the chain-based format of [22], it makes more sense for our purposes to define them via Minimalist derivation tree languages (MDTLs). To this end, I adopt the approach taken by [4], which builds on the notion of *slices* (introduced in [4] and [11]). The slice of a lexical item (LI) l consists of l itself and those interior nodes which denote an operation checking a licensor or selector feature of l. Intuitively, then, the slice of l is the derivation tree equivalent of the phrase projected by l in the derived tree (cf. Fig. 1). Since every node in a well-formed derivation tree belongs to exactly one slice, MDTLs can be regarded as the result of combining a finite number of slices in all possible ways such that all conditions imposed by the feature calculus are obeyed. Consequently, every MG is fully specified by some finite set of slices. Slices can be obtained from LIs via a simple recursive procedure.

Definition 1. *Let* Base *be a non-empty, finite set of* feature names. *Furthermore,* Op $:= \{merge, move\}$ *and* Polarity $:= \{+, -\}$ *are the sets of*

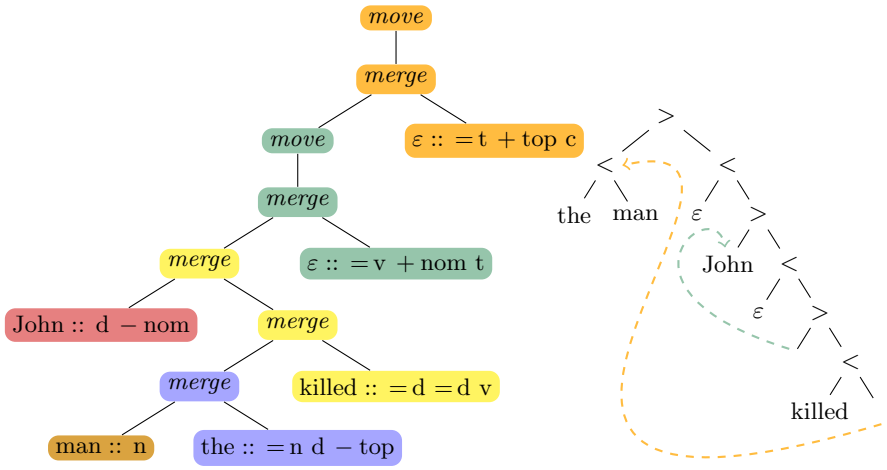

Fig. 1. Left: derivation tree of *The man, John killed*, with slices indicated by color; Right: corresponding derived tree, dashed arrows indicate movement

operations *and* polarities*, respectively. A* feature system *is a non-empty set* $Feat \subseteq \textsc{Base} \times \textsc{Op} \times \textsc{Polarity}$.

Note that this is merely a different notation for the familiar system of *category features* $f := \langle f, merge, - \rangle$, *selector features* $=f := \langle f, merge, + \rangle$, *licensee features* $-f := \langle f, move, - \rangle$, and *licensor features* $+f := \langle f, move, + \rangle$. In cases where only the name, operation, or polarity of f is of interest, $\nu(f)$, $\omega(f)$ and $\pi(f)$ will be used, respectively.

Definition 2. *Given a string alphabet* Σ *and feature system Feat, a* $(\Sigma, Feat)$-lexicon *is a finite subset of* $\Sigma \times \{::\} \times Feat^*$.

Definition 3. *Let Lex be a* $(\Sigma, Feat)$*-lexicon,* $Lex_\star := \{\sigma :: f_1 \cdots f_n \star \mid \sigma :: f_1 \cdots f_n \in Lex\}$*, and* Ω *the ranked alphabet* $\{l^{(0)} \mid l \in Lex\} \cup \{move^{(1)}, merge^{(2)}\}$*. Then the* slice lexicon of *Lex is* $\text{slice}(Lex) := \{\zeta(l) \mid l \in Lex_\star\}$*, where* $\zeta : Lex_\star \to T_\Omega$ *is given by*

$$\zeta(\sigma :: f_1 \cdots f_i \star f_{i+1} \cdots f_n) := \begin{cases} \sigma :: f_1 \cdots f_n \\ \qquad \text{if } f_1 \cdots f_i = \varepsilon \\ \zeta(\sigma :: f_1 \cdots f_{i-1} \star f_i \cdots f_n) \\ \qquad \text{if } \pi(f_i) = - \\ move(\zeta(\sigma :: f_1 \cdots f_{i-1} \star f_i \cdots f_n)) \\ \qquad \text{if } \tau(f_i) = move \text{ and } \pi(f_i) = + \\ merge(\square_i, \zeta(\sigma :: f_1 \cdots f_{i-1} \star f_i \cdots f_n)) \\ \qquad \text{if } \tau(f_i) = merge \text{ and } \pi(f_i) = + \end{cases}$$

I follow [4] in stipulating that slices are right branching, but this is merely a matter of convenience — linear order is irrelevant in derivation trees.

Despite its totality, ζ yields the intended result only for LIs of the form $\gamma c \delta$, where γ is a (possibly empty) string of selector and licensor features, c a category feature, and δ a (possibly empty) string of licensee features. As was shown in both [3] and [11], all LIs occurring in a well-formed derivation satisfy this condition. In anticipation of subsequent modifications of the formalism, though, I explicitly require this feature order.

F-Order Every LI is an element of $\Sigma \times \{::\} \times \{f \mid \pi(f) = +\}^* \times \{c \mid \omega(f) = \mathit{merge}, \pi(f) = -\} \times \{f \mid \omega(f) = \mathit{move}, \pi(f) = -\}^*$.

Given a slice lexicon slice(Lex), let $|\gamma|$ be the maximum of positive polarity features on a single LI. The closure of slice(Lex) under $|\gamma|$-fold tree concatenation is the *free slice language* FSL(slice(Lex)). Obviously this is not a well-formed MDTL (for instance, some trees still contain ports). Hence certain constraints must be enforced, which in turn requires additional terminology.

The *slice root* of LI $l := \sigma :: f_1 \cdots f_n$ is the unique node of $\zeta(l)$ reflexively dominating every node in $\zeta(l)$. An interior node of $\zeta(l)$ is *associated to feature* f_i *on* l iff it is the root of $\zeta(\sigma :: f_1 \cdots f_i \star f_{i+1} \cdots f_n)$. Two features f and g *match* iff they have identical names and operations but opposite feature polarities. An interior node m matches a feature g iff the feature m is associated to matches g. For every $t \in$ FSL(slice(Lex)) and LI l in t with string $-f_1 \cdots - f_n$ of licensee features, the *occurrences* of l in t are defined as follows:

- $occ_0(l)$ is the mother of the slice root of l in t (if it exists).
- $occ_i(l)$ is the unique node m of t labeled *move* such that m matches $-f_i$, properly dominates occ_{i-1}, and there is no node n in t that matches $-f_i$, properly dominates occ_{i-1}, and is properly dominated by m.

I also refer to occ_0(l) as the *zero occurrence of* l, while all other occurrences of l are *positive occurrences*. The i^{th} positive occurrence of l is simply the first Move node m one encounters when moving upwards through the derivation tree such that m is capable of checking the i^{th} licensee feature f_i of l after all preceding licensee features have already been checked. Given a well-formed derivation, m will always coincide with the node that actually checks f_i. In other words, the notion of occurrences provides a tree-geometric encoding of some parts of the Minimalist feature calculus pertaining to Move.

The remainder of the feature calculus can also be expressed tree-geometrically. For every $t \in$ FSL(slice(Lex)), node m of t, and LI l with licensee features $-f_1 \cdots - f_n$, $n \geq 0$:

Final For $F \subseteq$ Base a distinguished set of *final categories*, if the slice root of l is the root of t, then the category feature c of l is a final category, i.e. $\nu(c) \in F$.

Merge If m is associated to selector feature $= f$, then its left daughter is the slice root of an LI with category feature f.

Move There exist distinct nodes $m_1, \ldots, m_n$ such that m_i (and no other node of t) is the i^{th} positive occurrence of l, $1 \leq i \leq n$.

SMC If m is labeled *move*, there is exactly one LI for which m is a positive occurrence.

As expressed in Lemma 1 of [4], L is an MDTL iff it is the biggest subset of FSL(slice(Lex)) satisfying the four constraints above. Let us step back for a second and reflect on why this is the case. The first two constraints are easy to fathom. MGs require that the head of a derived tree must belong to some final category (usually c), and **Final** expresses this requirement in derivational parlance. Similarly, **Merge** ensures that an LI only selects LIs with matching category features. **Move** captures one half of MG's resource sensitivity with respect to Move: every licensee feature must be checked. For if an LI l has fewer occurrences than licensee features, some feature must remain unchecked because no Move node can check more than one of l's features. The other half of resource sensitivity is enforced by **SMC**: every Move node is responsible for checking exactly one licensee feature in the entire derivation. On the one hand, this rules out derivations with more licensor features than licensee features. On the other hand, it makes Move deterministic and rules out SMC violations. In a standard MG, the SMC guarantees for every step of a derivation that if $-f_i$ is the first licensee feature of an LI that still needs to be checked, it is not the first currently unchecked licensee feature of any other LI. If the SMC is violated, then the corresponding derivation tree must contain some node m properly dominating two LIs l and l' that both have $-f_i$ as their first unchecked feature. But then the lowest matching Move node reflexively dominating m will be an occurrence for both l and l', which is ruled out by **SMC**.

As all the constraints above can be stated in first-order logic with predicates for equality, proper dominance and right sister [4], we can view them as regular tree languages (i.e. as the sets of trees satisfying the respective constraints) and enforce them using regular control as described in [3]. The result is the desired MDTL — a regular set of derivation trees, each of which can be converted into the corresponding derived tree using the multi bottom-up transducer (mbutt) described in [12]. As is well-known, the string yield of an MG's derived tree language is an MCFL; in fact, MGs and MCFG are weakly equivalent [6, 13].

Definition 4. *A* Minimalist Grammar *is a 5-tuple* $G := \langle \Sigma, Feat, Lex, F, \mathcal{R} \rangle$ *such that*

- Lex *is a* $(\Sigma, Feat)$*-lexicon, and*
- $F \subseteq$ Base *is the set of final features, and*
- $\mathcal{R}$ *is a finite set of regular tree languages containing* ***F-Order***, ***Final***, ***Merge***, ***Move***, ***SMC***, *and nothing else.*

The MDTL of G *is* $\text{FSL}(\text{slice}(Lex)) \cap \bigcap_{R \in \mathcal{R}} R$*. The tree language* $L(G)$ *generated by* G *is the image of its MDTL under the mbutt of [12]. Its string language is the string yield of* $L(G)$*.*

3 New Movement Types

3.1 General Strategy

By regulating Move purely via structural conditions on the distribution of occurrences, we have successfully decoupled the SMC and the resource-sensitivity of MGs from the specifics of Move. Crucially, both **Move** and **SMC** hold independently of how occurrences are defined. In the previous section, proper dominance was used to capture *raising*, i.e. phrasal movement to a c-commanding position (not to be confused with raising constructions in the syntactic literature). However, if proper dominance was replaced by its inverse, the result would be *lowering* instead, which moves a phrase to a position it c-commands. As **Move** and **SMC** are unaffected by this change, the MDTL of an MG with lowering rather than raising would still be regular. In fact, regularity is preserved as long as the relation replacing proper dominance in the definition of occurrences is rational.

The mapping from derivation trees to derived trees, however, increases in complexity as the number of Move operations proliferates. Even if the mbutt of [12] were powerful enough to compute the mapping for any given rational relation, it would quickly become prohibitively complex. A better transduction model is provided by MSO-transductions. First, limiting ourselves to MSO-transductions guarantees that the string yield of the derived tree language is an MCFL (see [14] and references therein). Second, the MSO-transduction makes it very easy to capture other parameters of Move such as the size of the moved subtree. With respect to both expressivity and elegance, then, MSO-transductions are the ideal choice.

In order to avoid the complexities brought about by finite-copying MSO-transductions (which are necessary for the insertion of traces), I opt to decompose the transduction into two simpler steps. The derivation tree is first mapped to a term graph, also known as a multi-dominance tree in the syntactic literature. This first step handles differences in linearization and the size of the moved constituent. The term graph is then unfolded into a tree. Depending on how this unfolding is specified, one can allow for copying and covert movement.

Working with MSO-transductions obviously requires the introduction of some logical machinery. Due to space restrictions — and because it has already been accomplished in [4] — I refrain from giving a full model-theoretic formalization of MDTLs. I am also confident that the reader is sufficiently familiar with MSO to see that all constraints and definitions presented in the following sections are MSO-definable (keep in mind that a Minimalist lexicon is finite, so disjunctions, conjunctions and the recursive definitions given here are always finitely bounded). Where MSO-formulas are used, I follow the syntax of $\mathcal{L}^2_{K,P}$ [16], writing $\triangleleft$ for immediate dominance.

3.2 Step 1: Derivations and Occurrences

Generalizing the MG formalism at the level of derivation trees is quite simple. First the feature system is extended by another set M-Type of *movement types*,

which represent various kinds of movement, for instance raising, lowering and sidewards movement; the movement type of a feature will usually be indicated as a superscript to avoid confusion. Each $\triangle \in$ M-Type in turn is associated with a pair $\langle R_0^{\triangle}, R^{\triangle} \rangle$ of rational relations. The first one determines the zero occurrence of an LI (e.g. the mother of its slice root for raising), while the second one assumes the role previously played by proper dominance in determining the LI's positive occurrences. These pairs are called *movement specifications*, and I will often refer to them by the movement type they specify.

New movement types do not always behave as expected, though. With raising movement, competition between potential occurrences is eliminated by stipulating that only the closest (i.e lowest dominating) movement node counts as an occurrence. No such safeguard exists for lowering, though, and as a result many configurations are ambiguous and thus blocked by **Move** and **SMC**, which jointly ensure that Move is deterministic.

Consider the configuration in Fig. 2. From a linguistic perspective, the in-

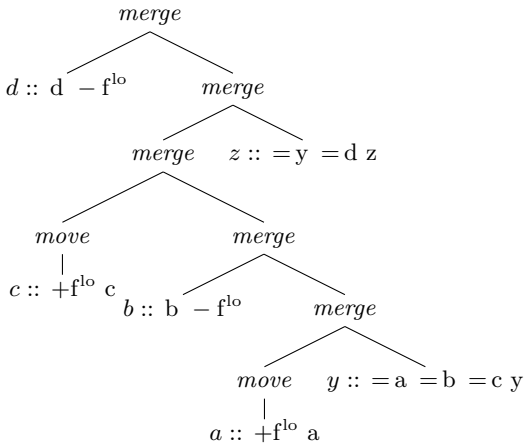

Fig. 2. The move node of LI b is a potential occurrence for both c and e, and e has two occurrences, the move nodes of b and d

tended result is clear: b and d should lower into the specifiers of a and c, respectively. Given the current conception of occurrences, however, this configuration is in fact illicit. The LI d has not one but two occurrences, namely the Move nodes of c and a. Neither properly dominates the other, so neither blocks the other, either. Furthermore, the Move node of a is an occurrence for both b and d. So both **Move** and **SMC** are violated. Note, however, that the zero occurrence of b is the Merge node immediately dominating it and that this Merge node intervenes between the zero occurrence of d and the Move node of a. It seems, then, that the derivation can be salvaged by stipulating that LIs and their occurrences may act as interveners, too. To this end, movement specifications are

modified to comprise another rational relation $P^{\triangle}$, and occurrences are defined in two steps.

- For all $i \geq 1$ and $\triangle \in$ M-Type, node m is a potential i-occurrence $pocc_i(l)$ of l iff
  - m matches $-f_i^{\triangle}$, and
  - $\langle m, pocc_{i-1}(l)\rangle \in R^{\triangle}$, and
  - there is no node z distinct from m such that
    - z matches $-f_i^{\triangle}$, and
    - $\langle z, pocc_{i-1}(l)\rangle \in R^{\triangle}$, and
    - $\langle m, z\rangle \in R^{\triangle}$.
- For all $i \geq 1$ and $\triangle \in$ M-Type, node m is an i-occurrence $occ_i(l)$ of l iff
  - m is a potential i-occurrence of l, and
  - there is no node l' distinct from l such that m is a potential j-occurrence of l', $j \geq 1$, and $\langle occ_{j-1}(l'), occ_{i-1}(l)\rangle \in P^{\triangle}$.
- Both $occ_0(l)$ and $pocc_0(l)$ hold of node m iff $\langle m, l\rangle \in R_0^{\triangle}$.

As multiple movement types may be realized in the same grammar, I will sometimes say that m is a $\triangle$-*occurrence of* l to express that m is a positive occurrence of l and matches some $-f_i^{\triangle}$.

The reader is encouraged to verify for himself that this new definition yields the same result as the previous one on page 5 if $R^{\triangle} = P^{\triangle}$ is taken to be proper dominance (as far as I can tell, the equivalence of $R^{\triangle}$ and $P^{\triangle}$ holds for all movement operations in the syntactic literature). Restricting proper dominance to paths containing at most $k+1$ left branches, on the other hand, gives rise to k-local movement in the sense of [4]. And as expected, lowering can be captured by using inverse proper dominance.

Sidewards movement [15, 20] is another prominent kind of movement. It does away with the c-command requirement on raising so that, for instance, complements and specifiers of specifiers are viable landing sites. For this reason, sidewards movement is heavily relied on in movement-based analyses of control and various extraction phenomena [cf. 7]. One conceivable formalization of sidewards movement is (inverse) slice containment: x slice contains y iff the slice containing a node immediately dominating x contains a node properly dominating y. If the zero occurrence of LI l is fixed to be the mother of the slice root of l, slice containment allows for locally restricted sidewards movement. The choice of slice might be freely altered to increase the locality domain, e.g. to the lowest slice of an LI of category c that properly dominates x. Minor modifications of this kind allow for sidewards movement to subsume previously implemented variants of Move such as Across-the-Board extraction [10] and clustering [2].

One peculiarity brought about by non-standard movement types is that the linguistic conception of derivation trees as a temporally ordered record of how derived trees are assembled in a step-wise fashion loses most of its intuitiveness. Going back to Fig. 2, for example, we see that b enters the derivation later than its last occurrence, the move node licensed by a (and similarly for d). Under the standard construal of derivation trees, this would entail that before b was

inserted into the derived tree it had already lowered into the specifier of a. This apparent contradiction can be resolved by viewing derivation trees as a relative of proof nets (see [17, 19] and references therein): they are merely a graph-theoretic representation of the Minimalist feature calculus and its checking requirement, with movement types corresponding to specific rules of inference in this calculus.

3.3 Step 2: Mapping to Term Graphs

The next step is the mapping from derivation trees to the derived structures posited by linguists. Recent Minimalist reasoning maintains that derived trees are actually multi-dominance trees — tree-like structures in which a node may have multiple mothers. Movement of a phrase XP to YP no longer involves displacement of XP into the (newly created) specifier of YP, with a trace or copy of XP being left behind in the original position. Instead, only a new dominance branch is added between YP and XP, making the latter a specifier of the former. The derived tree in Fig. 1, for instance, may be converted into this new format by interpreting the movement arrows as dominance branches. More importantly, the multi-dominance tree can also be obtained from the derivation tree by adding branches from the slice root of an LI to all its positive occurrences (ignoring linearity and labels, for now).

The mathematical analog of converting derivation trees into multi-dominance trees is transducing trees into term graphs. For the simple mapping required for MGs, MSO term graph transductions [1] are more than sufficient. Recall that such a transduction is specified by a pair $\langle \Psi, \Theta \rangle$. The first component, Ψ, is a set of formulas determining which nodes of the input tree are present in the term graph and what their label is, while Θ consists of formulas defining the relations that hold between the nodes of the term graph. Crucially, those formulas may only use predicates from the MSO-language used to define the input tree.

In our case, all nodes of the input tree are present in the term graph, so we only need to worry about changing the label and defining precedence $\prec$ and immediate dominance $\blacktriangleleft$. The latter is readily stated in terms of occurrences and immediate dominance in the derivation tree ($occ_i(x, l)$ holds iff x is the i^{th} occurrence of l, and $|\delta|$ is the maximum of licensee features occurring on a single LI in the grammar's lexicon).

$$x \blacktriangleleft y \leftrightarrow x \triangleleft y \vee \exists l \Big[\bigvee_{1 \leq i \leq |\delta|} occ_i(x, l) \wedge \mathit{sliceroot}(y, l) \Big]$$

The predicate *sliceroot* ensures that the dominance branch is added between the occurrence and the root of the slice whose LI is undergoing movement. In other words, it enforces phrasal movement. But we can of course replace *sliceroot* by a different predicate to yield other kinds of movement. For instance, if y and l are identical, the result is head movement. For pied-piping, y is the slice root of some slice containing the slice of l. Wee see, then, that *sliceroot* can be freely exchanged for other MSO-definable predicates to alter the size of the moved subtree.

More precisely, movement specifications are extended to $\langle R_0^{\triangle}, R^{\triangle}, P^{\triangle}, \mathit{root}^{\triangle} \rangle$, where $\mathit{root}^{\triangle}$ is an MSO formula with two free variables that picks out a unique node to serve as the root of the subtree carried along by $\triangle$-movement. Then $\blacktriangleleft$ is defined as follows ($\mathit{occ}_i^{\triangle}(x,l)$ holds iff both $\mathit{occ}_i(x,l)$ and x is a $\triangle$-occurrence):

$$x \blacktriangleleft y \leftrightarrow x \triangleleft y \vee \exists l \Big[\bigvee_{\substack{1 \leq i \leq |\delta| \\ \triangle \in \text{M-Type}}} \big(\mathit{occ}_i^{\triangle}(x,l) \wedge \mathit{root}^{\triangle}(y,l) \big) \Big]$$

Certain restrictions must be put in place, though, to ensure that the output of the transduction is indeed a term graph and in line with certain Minimalist intuitions. For all $t \in \text{FSL}(\text{slice}(\mathit{Lex}))$, nodes x, y, l of t, $\triangle, \circ \in$ M-Type, and string δ of licensee features:

Containment If $-f^{\triangle}$ precedes $-f^{\circ}$ in δ and both $\mathit{root}^{\triangle}(x,l)$ and $\mathit{root}^{\circ}(y,l)$ hold, then x reflexively dominates y.
Dominance If $\mathit{root}^{\triangle}(x,l)$ holds, then x reflexively dominates l.
Exocentricity If m is a positive occurrence of l, m is not associated to a feature of l.
No Cycle If $x \blacktriangleleft^{+} y$ holds, then $y \blacktriangleleft^{+} x$ does not.

The first three conditions are linguistically motivated. They prevent LIs from triggering displacement of unrelated subtrees, stop moved subtrees from seemingly recombining with material that had previously been left behind, and prohibit LIs from licensing their own movement. The last one ensures that no cycles are present in the output graph. Keep in mind that the transitive closure of $\blacktriangleleft$ is MSO-definable, so **No Cycle** is indeed an MSO formula.

Besides dominance, one must also take care of precedence and the output labels. This offers another opportunity to reign in a variant of Move, namely rightward movement (also known as extraposition). The feature system is enriched by yet another component, Headedness $:= \{\mathit{left}, \mathit{right}\}$. Headedness information simplifies the task of defining predicates for left daughter $\blacktriangleleft_1$, right daughter $\blacktriangleleft_2$, and precedence $\prec$ (which isn't necessarily a strict order in term graphs). Only the headedness of positive polarity features is taken into account. This is formally expressed by restricting the predicates $\mathit{left}(x)$ and $\mathit{right}(x)$ to interior nodes associated to features of the respective headedness. Furthermore $x \sim y$ iff x and y are nodes of the same slice.

$$x \blacktriangleleft_1 y \leftrightarrow x \blacktriangleleft y \wedge \big((x \sim y \wedge \mathit{left}(x)) \vee (x \not\sim y \wedge \mathit{right}(x)) \big)$$

$$x \blacktriangleleft_2 y \leftrightarrow x \blacktriangleleft y \wedge \big((x \sim y \wedge \mathit{right}(x)) \vee (x \not\sim y \wedge \mathit{left}(x)) \big)$$

$$x \prec y \leftrightarrow \exists x' \exists y' \exists z \big[(x' \approx x \vee x' \blacktriangleleft^{+} x) \wedge (y' \approx y \vee y' \blacktriangleleft^{+} y) \wedge z \blacktriangleleft_1 x' \wedge z \blacktriangleleft_2 y' \big]$$

Relabeling the interior nodes based on headedness is just as simple.

$$< (x) \leftrightarrow (\mathit{merge}(x) \vee \mathit{move}(x)) \wedge \mathit{left}(x)$$

$$> (x) \leftrightarrow (\mathit{merge}(x) \vee \mathit{move}(x)) \wedge \mathit{right}(x)$$

Finally, LIs must lose all their features but keep their string exponents. In principle one would have to ensure that the LI of the highest slice keeps its category feature, but this requirement needlessly complicates the transduction and is itself merely an artefact of the original MG formalism.

$$\bigwedge_{\sigma \in \Sigma} \Big(\sigma(x) \leftrightarrow \bigvee_{l::=\sigma :: f_1 \cdots f_n \in Lex} l(x) \Big)$$

3.4 Step 3: Unfolding into Derived Trees

The usual way to unfold a term graph into a tree requires unbounded copying: given a subtree t whose root has n mothers $m_1, \ldots, m_n$, create n copies t_i of t such that m_i is the mother of t_i. While this is a feasible strategy to accommodate MGs with copying [9], it increases weak generative capacity. I therefore restrict my attention to unfoldings without unlimited copying.

Let us first consider the case of standard MGs. Suppose LI l has n occurrences, so that the nodes $m_1 := occ_1(l), \ldots, m_n = occ_n(l), m_{n+1}$ all dominate the slice root of l; $m_{n=1}$ is the unique Merge node that introduced l into the derivation. Then the unfolding just has to create $n-1$ traces and replace the dominance branches between the slice root of l and each m_i, $i < n$, by a dominance branch between a trace and m_i. As a result, only the last occurrence of l immediately dominates its slice root, which is tantamount to saying that the constituent headed by l moved into the specifier immediately dominated by m_n.

In the syntactic literature, a distinction is made between overt and covert movement, however, and only the former is visible. For the purposes of unfolding this means that the branch to l's slice root should not be preserved for the last occurrence of l, but the occurrence with the highest index that licensed overt movement. To this end, the feature system is once again modified so as to indicate overtness via the diacritics o and c. The matching relation also needs to be extended accordingly to ensure that licensor and licensee features agree on (c)overtness.

This system is still unsatisfactory, though, as MGMGs allow for the size of the moved constituent to vary with feature type. This entails that more than just one occurrence of an LI l may dominate parts of the material that was displaced by moving l. The challenge is to find the last occurrence for each of these parts. Given LI l, derivation tree t and $\triangle, \circ \in$ M-Type, $\triangle \cong \circ$ iff t contains a node x such that $root^{\triangle}(x, l) = root^{\circ}(x, l) = 1$. Now for every LI l, $\triangle, \circ \in$ M-Type, and $j > i \geq 1$, $occ_i^{\triangle}(l)$ is a *landing site* iff $occ_i^{\triangle}$ is associated to an overt feature and there is no $occ_j^{\circ}(l)$ such that $\triangle \cong \circ$. The unfolding then turns the term graph into a tree such that if $occ_i^{\triangle}(l)$ is a landing site, it immediately dominates the $\triangle$-root of l. All branches originating from a Merge node immediately dominating the root of a displaced subtree or from an occurrence of l that is not a landing site are replaced by branches immediately dominating a trace.

Clearly the number of traces per term graph cannot exceed the total number of nodes in the graph, wherefore the unfolding is of linear size increase. It follows immediately that the composition of our MSO term transduction and unfolding

is an MSO-definable tree transduction (with finite copying). Let τ be this tree transduction. As MDTLs are still regular, it must be the case for every single one of them that the string yield of its image under τ is an MCFL.

3.5 Defining Movement-Generalized Minimalist Grammars

Now we are finally in a position to define MGMGs.

Definition 5. *Let* Base *and* M-Type *be disjoint, non-empty, finite sets of* feature names *and* movement types*, respectively. Furthermore,* Op $:= \{merge, move\}$, Polarity $:= \{+, -\}$, Headedness $:= \{left, right\}$, *and* Overt $:= \{o, c\}$, *are the sets of* operations, polarities, headedness parameters, *and* overtness markers*, respectively. A* feature system *is a non-empty set Feat* $\subseteq$ Base $\times$ Op $\times$ Polarity $\times$ M-Type $\times$ Headedness $\times$ Overt. *Two features* match *iff they agree on their name, operation, movement type, and overtness but have opposite polarities.*

Definition 6. *Given* $\triangle \in$ M-Type*, the* movement specification of $\triangle$ *is given by a 4-tuple* $\left\langle R_0^{\triangle}, R^{\triangle}, P^{\triangle}, \mathrm{root}^{\triangle} \right\rangle$ *of binary rational relations.*

Definition 7. *A* Movement-Generalized Minimalist grammar G *over alphabet* Σ *and feature system Feat is a 6-tuple* $G := \langle \Sigma, Feat, Lex, F, \mathcal{R}, \mathcal{M} \rangle$*, where*

- *Lex is a* $(\Sigma, Feat)$*-lexicon, and*
- $F \subseteq$ Base *is a set of final categories, and*
- $\mathcal{R}$ *is a finite set of regular tree languages containing at least* ***Containment***, ***Dominance***, ***Exocentricity***, ***F-Order***, ***Final***, ***Merge***, ***Move***, ***No Cycle***, ***SMC****, and*
- $\mathcal{M}$ *is an* M-Type*-indexed family of movement types.*

Its MDTL is $\mathrm{FSL}(\mathrm{slice}(Lex)) \cap \bigcap_{R \in \mathcal{R}} R$*. The tree language* $L(G)$ *generated by* G *is the image of its MDTL under the MSO transduction* τ*, and its string language is the string yield of* $L(G)$*.*

Theorem 1. *MGs and MGMGs have the same weak generative capacity.*

4 Tree Adjunction $\equiv$ Reset Lowering

Even though MGs properly subsume TAGs with respect to weak generative capacity [13, 18], the two formalisms are incomparable at the level of tree languages [12, 14]. This result does not hold for MGMGs. In fact, TAGs with strictly binary branching and X′-like projection are equivalent to MGs with a limited kind of lowering, as I will briefly sketch now (for a full proof see [5]).

Consider the following scenario. The tree α consists of subtrees t(op) and b(ottom), with b rooted in the node V′ of t, which is a projection of LI l_α in b. The auxiliary tree β, whose foot node is $\underline{\mathrm{V}'}$ and whose root is a projection of LI l_β, adjoins into α at V′, yielding γ. It should be easy to see that γ can

be approximated via lowering. First, β is selected by l_α immediately after the subtree immediately dominated by V′ in b. After that, the foot node of β is replaced by an empty category with licensor feature $+f^\triangle$, and $-f^\triangle$ is inserted after the category feature of l_α. The $\triangle$-root of l_α is the sister of the root of β. The derived tree corresponding to this lowering step only differs from γ in the presence of two superfluous interior nodes immediately above the $\triangle$-root of b and the root of β, respectively; both can easily be detected and removed.

The procedure carries over to the general case without major problems, but it hinges on a particular definition of lowering. For example, if another auxiliary tree β' was to adjoin immediately above V′ in t, the algorithm would add the requisite nodes and features as intended (now using a new movement type $\circ$ to pick out the correct root). But since the Move nodes in β and β' are not related by inverse proper dominance, defining lowering in these terms is insufficient — only the first occurrence could ever be reached. Note, though, that each $u \in \{\beta, \beta'\}$ contains exactly one $\triangle$-occurrence of l_α, where the $\triangle$-root of l_α is the sister of the root of u. In a sense then, we do not want occurrences to be computed in sequence, but rather independently of each other using inverse proper dominance and picking the $\triangle$-root of l as the zero-occurrence for computing $\triangle$-occurrences. Emulating this behavior in the MGMG system is slightly cumbersome: for all $\triangle \neq \circ \in$ M-Type, $\langle x, y\rangle \in R^\triangle$ iff either y is a $\triangle$-root c-commanding x or there is a $\triangle$-root x' and a $\circ$-root y' such that y' properly dominates x, no $\circ$-root properly dominated by y' properly dominates x', y' c-commands y, and x' c-commands x. In conjunction with **Containment**, this always yields a well-defined relation that exhibits the desired behavior. I call this relation *reset lowering*.

Although many technical details are missing, it should nonetheless be clear that the translation described above can be carried out by a linear tree transducer. Since TAG derivation tree languages are regular, the output language L of the transducer is too. In order to convert L into an MDTL, one only needs to employ the label refinement algorithm given in [3]. The end result is an MG with reset lowering that generates the same tree language as the original TAG (under a simple homomorphism that removes the redundant interior nodes).

As for the translation in the other direction, I presuppose that all licensee features are built from the same feature name f, that is to say, there are no two distinct features $-f^\triangle$, $-g^\triangle$ for any $\triangle \in$ M-Type. The reader may verify for himself that the MGs created by the algorithm above satisfy this condition. From **Containment** and the definition of reset lowering it further follows that no LI has more than one feature of a specific movement type. Now we only have to apply the spirit of the previous translation in reverse. Suppose we are given subtrees t, b, β and a node u that is immediately dominated by a leaf v of t and immediately dominates the roots of b and β. Let α be the composition of t and b such that v immediately dominates the root of b. Then lowering of b into β corresponds to adjunction of β into α at the root of b.

Hopefully the reader can appreciate now why L is a tree adjoining language iff it is the derived tree language of some MGMG with reset lowering and only one feature name per movement type. For MGMGs with normal lowering, the

translation must fail because for every $i > 1$ there is such an MGMG G with $\textsc{Base} = \{f\}$ that generates the language $a_1^n \cdots a_i^n$. The lexicon of G contains

- $a_j :: \mathrm{a_j}$ for all $j < i$,
- $a_i :: {=}\mathrm{a_1} \cdots {=}\mathrm{a_j}\ \mathrm{a_i}\ (-\mathrm{f}^{\mathrm{a_1}} \cdots - \mathrm{f}^{\mathrm{a_i}})$,
- $a_i :: {=}\mathrm{a_i}\ + \mathrm{f}^{\mathrm{a_1}} {=}\mathrm{a_1} \cdots + \mathrm{f}^{\mathrm{a_i}}\ \mathrm{a_i}\ (-\mathrm{f}^{\mathrm{a_1}} \cdots - \mathrm{f}^{\mathrm{a_i}})$,

where the $\mathrm{a_j}$-root of an LI is either the Merge node immediately dominating a_j or the Move node immediately above that (if it exists).

Conclusion

MGs can easily be generalized to MGMGs once we view them in terms of their derivation trees. The notion of occurrence, which is used to regulate the distribution of Move nodes, can be redefined to allow for variants of Move with different directionality (lowering, sidewards movement etc.). The mapping from derivation trees to derived trees, on the other hand, furnishes parameters to determine linearization, the size of the moved subtree, and the overt/covert distinction. As all these modifications are required to be MSO-definable, MGMGs have the same weak generative capacity as MGs despite their greatly increased strong generative capacity.

Acknowledgments I would like to thank the LACL reviewers, Ed Stabler, and all the members of the UCLA MathLing Circle; without their questions and suggestions, this paper would have been even less approachable. Furthermore, several discussions with Michael Freedman during ESSLLI 2011 on an earlier version of the TAG-to-MG translation improved my understanding of the TAG formalism in various ways.

Bibliography

[1] Bloem, R., Engelfriet, J.: A comparison of tree transductions defined by monadic second-order logic and attribute grammars. Journal of Computational System Science 61, 1–50 (2000)
[2] Gärtner, H.M., Michaelis, J.: On the treatment of multiple-wh-interrogatives in minimalist grammars. In: Hanneforth, T., Fanselow, G. (eds.) Language and Logos, pp. 339–366. Akademie Verlag, Berlin (2010)
[3] Graf, T.: Closure properties of minimalist derivation tree languages. In: Pogodalla, S., Prost, J.P. (eds.) LACL 2011. Lecture Notes in Artificial Intelligence, vol. 6736, pp. 96–111 (2011)
[4] Graf, T.: Locality and the complexity of minimalist derivation tree languages. In: Proceedings of the 16$^{\text{th}}$ Conference on Formal Grammar (2011), to appear
[5] Graf, T.: Tree adjunction as lowering in minimalist grammars (2012), ms., UCLA

[6] Harkema, H.: A characterization of minimalist languages. In: de Groote, P., Morrill, G., Retoré, C. (eds.) Logical Aspects of Computational Linguistics (LACL'01), Lecture Notes in Artificial Intelligence, vol. 2099, pp. 193–211. Springer, Berlin (2001)
[7] Hornstein, N.: Movement and control. Linguistic Inquiry 30, 69–96 (1999)
[8] 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)
[9] Kobele, G.M.: Generating Copies: An Investigation into Structural Identity in Language and Grammar. Ph.D. thesis, UCLA (2006)
[10] Kobele, G.M.: Across-the-board extraction and minimalist grammars. In: Proceedings of the Ninth International Workshop on Tree Adjoining Grammars and Related Frameworks (2008)
[11] Kobele, G.M.: Minimalist tree languages are closed under intersection with recognizable tree languages. In: Pogodalla, S., Prost, J.P. (eds.) LACL 2011. Lecture Notes in Artificial Intelligence, vol. 6736, pp. 129–144 (2011)
[12] 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)
[13] Michaelis, J.: Transforming linear context-free rewriting systems into minimalist grammars. Lecture Notes in Artificial Intelligence 2099, 228–244 (2001)
[14] Mönnich, U.: Grammar morphisms (2006), ms. University of Tübingen
[15] Nunes, J.: The Copy Theory of Movement and Linearization of Chains in the Minimalist Program. Ph.D. thesis, University of Maryland, College Park (1995)
[16] Rogers, J.: A Descriptive Approach to Language-Theoretic Complexity. CSLI, Stanford (1998)
[17] Salvati, S.: Minimalist grammars in the light of logic. In: Pogodalla, S., Quatrini, M., Retoré, C. (eds.) Logic and Grammar — Essays Dedicated to Alain Lecomte on the Occasion of His 60th Birthday, pp. 81–117. No. 6700 in Lecture Notes in Computer Science, Springer, Berlin (2011)
[18] Seki, H., Matsumura, T., Fujii, M., Kasami, T.: On multiple context-free grammars. Theoretical Computer Science 88, 191–229 (1991)
[19] Stabler, E.P.: Remnant movement and complexity. In: Bouma, G., Kruijff, G.J.M., Hinrichs, E., Oehrle, R.T. (eds.) Constraints and Resources in Natural Language Syntax and Semantics, pp. 299–326. CSLI Publications, Stanford, CA (1999)
[20] Stabler, E.P.: Sidewards without copying. In: Penn, G., Satta, G., Wintner, S. (eds.) Formal Grammar '06, Proceedings of the Conference. pp. 133–146. CSLI Publications, Stanford (2006)
[21] Stabler, E.P.: Computational perspectives on minimalism. In: Boeckx, C. (ed.) Oxford Handbook of Linguistic Minimalism, pp. 617–643. Oxford University Press, Oxford (2011)
[22] Stabler, E.P., Keenan, E.: Structural similarity. Theoretical Computer Science 293, 345–363 (2003)