# Tree Adjunction as Minimalist Lowering

**Thomas Graf**
Department of Linguistics
University of California, Los Angeles
3125 Campbell Hall
Los Angeles, CA 90095-1543, USA
tgraf@ucla.edu

## Abstract

Even though Minimalist grammars are more powerful than TAG on the string level, the classes of tree languages the two define are incomparable. I give a constructive proof that if the standard Move operation in Minimalist grammars is replaced by *Reset Lowering*, every TAG tree language can be generated. The opposite does not hold, so the strong generative capacity of Minimalist grammars with Reset Lowering exceeds that of TAG.

## 1 Introduction

The comparison of grammar formalisms with respect to their expressive power has been essential to furthering our understanding of their inner workings (Weir, 1988; Joshi et al., 1991; Vijay-Shanker and Weir, 1994). Considering a formalism in isolation only takes us so far, it is by connecting it to other proposals that we see which parts are indispensable, why this is the case, and how they grant the grammar its power.

In recent years, significant attention has been devoted to the comparison of TAG and Minimalist grammars (MGs; Stabler, 2011). These two formalisms are interrelated in peculiar ways. MGs are weakly equivalent to MCFGs (Michaelis, 1998; Michaelis, 2001; Harkema, 2001) and thus subsume TAG on the string level. But with respect to the tree languages they generate, the two are in fact incomparable (Fujiyoshi and Kasai, 2000; Mönnich, 1999; Mönnich, 2006; Mönnich, 2007; Kobele et al., 2007), due a profound difference in how trees are cut up and reassembled in the respective formalisms.

What gives TAGs an edge over MGs is the limited kind of context-free tree manipulation granted by tree adjunction. Tree adjunction slices a tree in half and "glues" it back together with new material inserted in the middle, similar to how context-free string languages insert new substrings inside an existing string. This allows TAGs to generate tree languages that satisfy certain context-free path conditions, e.g. that a branch must be of the form $a^n b^n$ when read from the root towards the leaf.

In this paper, I show that the chasm between MGs and TAGs can be overcome by replacing the standard Move operation with *Reset Lowering*. The idea builds on earlier work of mine in Graf (2012), where I present a general schema for enriching MGs with new variants of Move without increasing their weak generative capacity. Using Reset Lowering, it is straight-forward to translate (suitably preprocessed) TAG derivation trees into Minimalist derivation trees that encode the same derived trees (*modulo* empty nodes left behind by movement).

**Organization.** I first introduce MGs with Reset Lowering in Sec. 2. The presentation is slightly informal, but all technical details can be deduced from Graf (2012). The remainder of the paper is devoted to the translation from TAG to MGs (Sec. 3). After a few remarks on how a given TAG's elementary trees can be brought in line with the kind of X′-like phrase structure template used by MGs (Sec. 3.2), I describe the actual translation in Sec. 3.3. I also discuss how the output of the translation can be guaranteed to be a well-formed Minimalist derivation tree language (Sec. 3.4). Some familiarity with TAG and MGs is presupposed on the reader's part.



*Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, pages 19–27,
Paris, September 2012.

## 2 Minimalist Grammars with Reset Lowering

MGs are a highly lexicalized framework using two basic operations, *Merge* and *Move*. Merge combines two distinct trees and projects a label in an X′-style fashion, whereas Move applies to a single tree $t$, takes some subtree $s$ of $t$ and puts it into the specifier of the currently highest phrase of $t$. Every lexical item (LI) is associated with a sequence of Merge and Move features that need to be checked by these operations in the order that they occur in. Both types of features come in two polarities, positive and negative. By convention, every LI $l$ has exactly one negative Merge feature (its *category feature*), which all of $l$'s positive polarity features must precede and all other negative polarity features must follow. If two LIs have matching features of opposite polarities as their first unchecked feature, the corresponding operation is triggered. By the end of the derivation, all features must have been checked off except for the category feature of the LI that projects the root of the tree, which must be a specifically designated *final category*.

The MG apparatus can be viewed as a combination of well-formedness conditions on derivation trees combined with a mapping from derivation trees to derived trees. Graf (2012) uses this perspective to generalize both components along several dimensions while keeping MGs within the bounds of MCFLs. Three parameters are of interest here. First, every positive feature $f$ is also specified for directionality, indicating whether the subtree headed by the LI with the matching feature for $f$ is the left or the right daughter of the root of the newly constructed tree. Second, the size of the constituent carried along by an LI $l$ that undergoes movement is no longer fixed to the entire phrase headed by $l$ but can be specified explicitly for each feature. Third, the target site of Move is no longer restricted to a c-commanding position; any position that can be picked out by a formula of monadic second-order logic is sufficient.

The expanded feature system and the Minimalist lexicons one can build from them are defined as follows:

**Definition 1.** Let BASE be a non-empty finite set of *feature names*. Furthermore, OP $:= \{merge, move\}$, POL $:= \{+, -\}$, SIZE $\subset \mathbb{N}$ finite, and DIR $:= \{\lambda, \rho\}$ are the sets of *operations*, *polarities*, *sizes*, and *directionality parameters*, respectively. A *feature system* is a non-empty set $Feat \subseteq$ BASE $\times$ OP $\times$ POL $\times$ SIZE $\times$ DIR.

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

Note that the double colon serves only cosmetic purposes, and that features must still appear in the specific order described at the beginning of the section. Moreover, not all components are always meaningful: SIZE is irrelevant for all Merge features, DIR for all negative polarity features.

Derivation trees play a central role in this paper. Their leaves are labeled by LIs, while binary branching nodes are labeled Merge and unary branching ones Move. The crucial difference between derivation trees and derived trees is that movement is only indicated by the presence of a Move node marking its target site, while the subtree to be displaced remains *in situ* (skip ahead to Fig. 1 for an example).

The main duty of derivation trees is to encode the operations of the Minimalist feature calculus. Counting from an LI $l$ towards the root, the $i$-th node is *associated* to the $i$-th positive polarity feature of $l$ (if it exists). Every interior node mode must be associated to exactly one feature of exactly one LI — its *feature associate* — and every positive polarity feature of every LI must be a feature associate of exactly one node. The LI carrying an interior's node feature associate is also called its *lexical associate*. Furthermore, there must be a matching feature for every interior node's feature associate, where two features *match* iff they agree on their name, operation, and size, but have opposite polarities.

In general, there will be many matching features, but only one of them can be involved in checking off an interior node's feature associate. How this feature is determined varies between Merge and Move. For a Merge node $n$ with lexical associate $l$, it is the category feature of the unique LI $l' \neq l$ that is the lexical associate of a node immediately dominated by $n$ in the derivation tree. For $n$ a Move node, on the other hand, the feature is determined via *occurrences*, where the definition of occurrences depends on the type of movement to be modeled.

I only present the definition for the type of

movement used throughout this paper, *Reset Lowering*. First, the $k$-root of $l$ is the unique node $n$ such that the shortest descending path from $n$ to $l$ has length $k$. The 0-root of $l$ is $l$ itself.

**Definition 3.** A Move node $m$ associated to a feature $f$ of size $k$ is a *potential $i$-occurrence of $l$* iff the $i$-th negative Move feature of $l$ matches $f$ and the $k$-root of $l$ c-commands $m$ in the derivation tree. It is a potential occurrence of $l$ iff it is a potential $i$-occurrence of $l$ for some $i > 0$. It is an *$i$-occurrence of $l$* iff it is a potential $i$-occurrence of $l$ and there is no $l' \neq l$ such that the $k$-root of $l$ c-commands $l'$ and $n$ is a potential occurrence of $l$.

Intuitively, Move node $n$ is an $i$-occurrence of $l$ iff I) $l$ has the right kind of feature, and II) the root of the subtree to be displaced c-commands $n$ (i.e. the target site), and III) no other LI that is closer to $n$ satisfies requirements I) and II).

Now the distribution of Move nodes can be regulated by two conditions.

**Move.** For every LI $l$ with negative Move features $f_1, \ldots, f_n$, there exist distinct nodes $m_1, \ldots, m_n$ such that $m_i$ (and no other node) is the $i$-th occurrence of $l$, $1 \leq i \leq n$.

**SMC.** Every Move node is an occurrence of exactly one LI.

This ensures in a purely tree-geometric fashion that all movement features in the derivation get checked.[1] Given a grammar $G$ with lexicon $L$, its *Minimalist derivation tree language* is the set of all derivations that can be assembled from items in $L$ and satisfy the conditions above.

Once the occurrences of all LIs $l$ are known, the mapping from derivation trees to derived trees is straightforward. First, a branch is drawn from $o$ to the respective $k$-root of $l$, where $o$ is the $i$-occurrence of $l$ with the highest value for $i$. Afterwards, the branch from the $k$-root to its mother is removed, so that $o$ is the only mother of the $k$-root now. When this has been done for all LIs, any remaining unary branching nodes are given the empty string $\varepsilon$ as their second daughter. The proper linearization of the derivation tree is controlled by the directionality of the positive features. Finally, it only remains to relabel the interior nodes by the distinguished symbols $<$ and $>$, which point in the direction of the projecting head — an interior node is labeled $<$ iff its feature associate has directionality $\rho$.

Let us finish with an example grammar for the language $a^n b^n$, $n \geq 2$. An easy way of generating it is to create multiple instances of $ab$ and then lower each $\sigma \in \{a, b\}$ into the specifier of the next lowest $\sigma$. The grammar in Fig. 1, with $f$ as its only final category, does just that. For the sake of succinctness, Merge features are given in lower case, Move features in upper case, polarities as superscripts, directionality as subscripts, and size in square brackets.

It should be easy to see that this grammar can be extended to $a_1^n \ldots a_k^n$ for any choice of $k$, proving that MGs with Reset Lowering trump TAG in terms of weak generative capacity (possible restrictions will be hinted at in Sec. 3.3.6). The same holds for the weakly equivalent standard MGs, though, yet they cannot generate all TAG tree languages. The translation presented in the next section proves that MGs with Reset Lowering can.

## 3 The Translation Procedure

### 3.1 Overview

The mapping from TAGs to MGs proceeds in three stages. First, the TAG grammar must be preprocessed to accommodate MGs' restriction to binary branching, X′-like tree structures. Afterwards, the translation proper is defined over derivation trees in a piece-wise by case fashion. This is done via three three functions $\phi$, $\mu$ and $\tau$. The role $\phi$ is to determine which features are needed for which nodes, and $\mu$ uses this information to transfer elementary trees in fragments of Minimalist derivations, which are then pieced together by $\tau$. The specifics of $\mu$ vary depending on what kind of TAG operation needs to be emulated. While substitution is easily handled by standard Merge, reigning in adjunction via Reset Lowering depends on a trick: instead of adjoining tree $u$ at node $n$, one can Merge it as a sister of $n$ and subsequently lower the subtree rooted in $n$ to where the foot node would be in $u$. The procedure is sketched in Fig. 2. After the derivation tree transduction from TAGs to MGs is put in place, it only remains to ensure that the resulting MG does

[1] For *Reset Lowering*, it also implies that every LI has at most one negative movement feature of size $k$ (otherwise every Move node c-commanded by its $k$-root would count as an occurrence, in violation of **Move**).

a1) $a :: A^+_\rho[0]$ *a-lo*$^-$
a2) $a :: A^+_\rho[1]$ *a-lo*$^-$
a3) $a :: A^+_\rho[0]$ *b*-lo$^+_\lambda$ *a-mid*$^-$ $A^-[1]$
a4) $a :: A^+_\rho[1]$ *b*-lo$^+_\lambda$ *a-mid*$^-$ $A^-[1]$
a5) $a :: A^+_\rho[0]$ *b*-mid$^+_\lambda$ *a-mid*$^-$ $A^-[1]$
a6) $a :: A^+_\rho[1]$ *b*-mid$^+_\lambda$ *a-mid*$^-$ $A^-[1]$
a7) $a :: b\text{-}lo^+_\lambda$ *a-hi*$^-$ $A^-[0]$
a8) $a :: b\text{-}mid^+_\lambda$ *a-hi*$^-$ $A^-[0]$

b1) $b :: B^+_\rho[0]$ *a*-lo$^+_\lambda$ *b-lo*$^-$
b2) $b :: B^+_\rho[1]$ *a*-lo$^+_\lambda$ *b-lo*$^-$
b3) $b :: B^+_\rho[0]$ *a*-mid$^+_\lambda$ *b-mid*$^-$ $B^-[1]$
b4) $b :: B^+_\rho[1]$ *a*-mid$^+_\lambda$ *b-mid*$^-$ $B^-[1]$
b5) $b :: a\text{-}hi^+_\lambda$ *b-hi*$^-$ $B^-[0]$
e) $\varepsilon :: b\text{-}hi^+_\lambda$ $f^-$

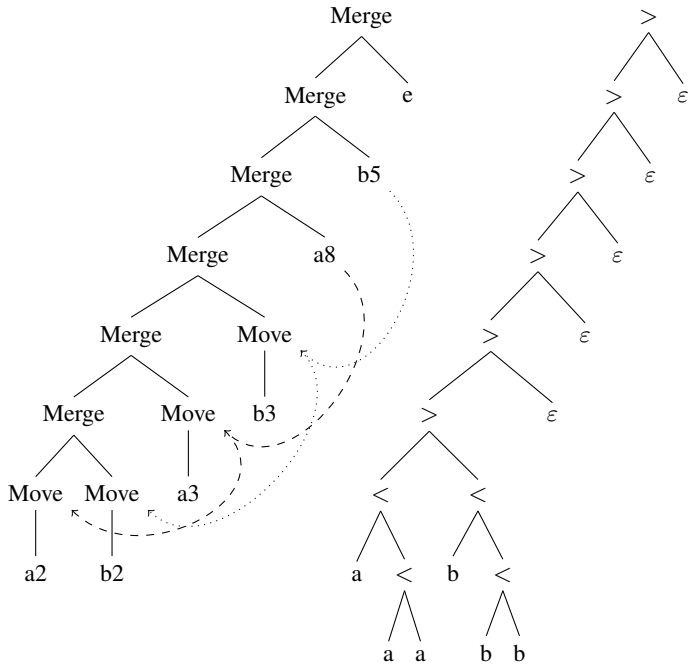


Figure 1: Top: Reset Lowering grammar for $a^n b^n$, $n \geq 2$; Bottom: derivation and derived tree for $a^3 b^3$

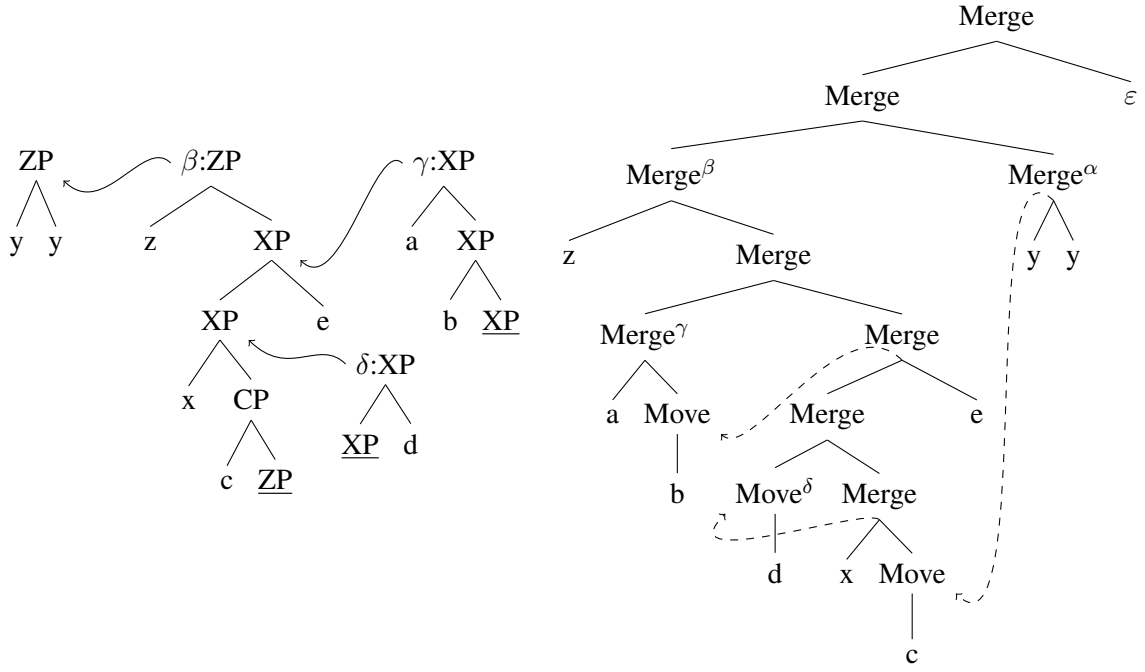


Figure 2: Tree Adjunction as Reset Lowering; solid arrows represent adjunction, dashed arrows movement, and the roots of the elementary trees are indicated by superscripts.

not overgenerate. This is a simple task, though, thanks to the attractive closure properties of MGs (Graf, 2011; Kobele, 2011).

### 3.2 Step 1: Preprocessing

We already saw in Sec. 2 that the tree languages derived by MGs are somewhat peculiar in that they are strictly binary branching and follow a projection scheme inspired by X′-theory. While the usual binarization strategies can easily be applied to TAG and need not be discussed here, imposing projection on an arbitrary TAG is slightly more tricky. Since not all elementary trees of a given TAG may necessarily contain any LIs, there might be no upper bound on the length of the shortest descending path in a derived tree from some interior node to some LI. This is impossible in MGs and thus needs to be prevented. A simple, albeit not particulary elegant solution is to insert empty LIs where necessary. How exactly one goes about this has no bearing on the translation procedure as long as every interior node with a non-terminal symbol is a projection of some LI.

Since every TAG is required to obey projection, it makes sense to introduce some extra notation to talk about trees more efficiently. The label of a node $n$ is given by $\ell(n)$. For every node $n$, its head $\hbar(n)$ is the leaf $l$ that $n$ is a projection of. If $t$ is a tree with root $r$, then $\hbar(t) := \hbar(r)$. In the other direction, $\pi(l) := p_1 \cdots p_n$ is the string of all projections of $l$ such that each $p_1$ is the parent of $l$ and each $p_i$ is immediately dominated by $p_{i+1}$, $1 \leq i < n$. A node $n$ is a *maximal projection* of $l$ iff $l = n$ or $n$ the last symbol of $\pi(l)$. Note that $l$ can be its own maximal projection even though it is never included in $\pi(l)$. Parts of trees will sometimes be specified via functional notation such that $f(a, b)$ is a tree in which $f$ immediately dominates $a$ and $b$.

The following terminology will be adopted to avoid confusion brought about the fact that TAG allows for terminal nodes to be decorated with non-terminal symbols: leafs labeled with terminal symbols will be referred to as LIs, while non-terminal is used exclusively for interior nodes. Keep in mind that thanks to the projection requirement the only nodes with non-terminal symbols that aren't part of some LI's projection are foot nodes and substitution nodes.

### 3.3 Step 2: Translating TAG Derivations

#### 3.3.1 Derivation Trees

The translation $\tau$ from TAGs to MGs operates at the level of derivation trees, which for TAG are defined as follows. Let $E$ be some finite set of elementary trees and $\nu$ some function that assigns each $e \in E$ a unique name. A *derivation tree over* $E$ is an ordered tree over the (unranked) alphabet $\Sigma := \{\nu(e) \mid e \in E\} \times (\{\varepsilon\} \cup A)$, where $A$ is the smallest set containing an address for every node in every $e \in E$ (as a notational shorthand, $e$ is often used instead of $\nu(e)$ where convenient). Since every $e \in E$ is finite in size and $E$ has finite cardinality, $A$ is finite, too, wherefore $\Sigma$ is indeed an alphabet. If node $n$ in derivation tree $t$ is immediately dominated by node $m$ such that $m$ and $n$ have labels $\langle u, a\rangle$ and $\langle v, b\rangle$, respectively, that is to be interpreted as tree $v$ adjoining in tree $u$ at the node $p$ with address $b$ (we require that this node exists in $u$). Often $p$ will simply be referred to as $b$. The second component of a label is $\varepsilon$ iff it is the label of the root of the derivation tree. Furthermore, if nodes $m$ and $n$ are siblings, their labels must differ in their second component. That is to say, no two trees ever adjoin to the same node, which guarantees that the branching factor of derivation trees is bounded and that there is a unique derived tree language. Note that elementary trees containing foot nodes and substitution nodes must have at least two nodes total. For every TAG $G$ with set $E$ of elementary trees, it holds that its derivation tree language is a subset of the set of all derivations over $E$.

#### 3.3.2 Single Elementary Tree

We start with the simplest case, a derivation tree $t$ that consists of only one node $u$. Note that $u$ cannot contain a foot node or substitution node, for then $t$ would not be well-formed. Hence every node in $u$ is either an LI or part of the projection of some LI. We convert $u$ into an MG derivation in two steps.

For every LI $n$ in $u$, $\phi(n) := \langle \ell(n), merge, -\rangle$. If $n$ is a non-terminal and its left daughter $d$ is a maximal projection, $\phi(n) := \langle \ell(\hbar(d)), merge, +, \lambda\rangle$. Substitute $\rho$ for $\lambda$ if $d$ is the right daughter of $n$. Since $u$ is binary branching and every non-terminal in $u$ must be a projection of some LI, $\phi$ is well-defined for all nodes in $u$.

Now for every LI $n$ with $\pi(n) := p_1 \cdots p_n$,

let $\mu(n) := \ell(n) :: \phi(p_1) \cdots \phi(p_n)\ \phi(n)$. For all non-terminals $n$, $\mu(n)$ is the second component of $\phi(n)$, which so far is restricted to Merge. As $\phi$ before, $\mu$ is well-defined for all of $u$. This carries over to the standard extension of $\mu$ from nodes to trees, which is denoted by $\hat{\mu}$. It is easy to see that $\hat{\mu}$ maps $u$ to its corresponding MG derivation. The required feature values are determined by $\phi$ — including values for the linear order of nodes — and then instantiated on the heads of the respective projections. Interior nodes are universally labeled Merge. Therefore the derived tree encoded by the Minimalist derivation $\hat{\mu}$ is identical to $u$ (under a deterministic relabeling of interior node labels); we may safely set $\tau(t) := \hat{\mu}(u)$.

### 3.3.3 Substitution

Now consider a derivation $t$ in which trees $v_1, \ldots, v_n$ substitute into tree $u$ at addresses $a_1, \ldots, a_n$ (and there is no $v_j \neq v_i$, $1 \leq i \leq n$ that is dominated by $u$ in $t$). Then $u$ contains $n$ distinct substitution nodes $s_1, \ldots, s_n$, where $n$ is bounded by the size of $u$. By assumption (cf. Sec. 3.2) there are at least two nodes in $u$. Because substitution nodes must be leaves, this entails that each $s_i$ has a parent $p_i$, which is the projection of some LI.

Let $\phi(p_i) := \langle \ell(\hbar(v_i)), merge, +, \lambda/\rho \rangle$ if $s_i$ is the left/right daughter of $p_i$. In addition, $\mu(s_i) := \square_i$. Then $\tau(\langle u, a \rangle (\langle v_1, a_1 \rangle, \ldots, \langle v_n, a_n \rangle))$ is the result of replacing each $\square_i$ in $\hat{\mu}(u)$ by $\tau(\langle v_i, a_i \rangle (t_1, \ldots, t_k))$, where $t_1, \ldots, t_k$ are all the subtrees immediately dominated by $v_i$ in $t$, $k \geq 0$. This yields once again a well-formed Minimalist derivation, as the feature instantiation of $p_i$ and $\hbar(v_i)$ via $\phi$ ensures for all $1 \leq i \leq n$ that $\hbar(v_i)$ is selected by $\hbar(p_i)$.

### 3.3.4 Adjunction

Finally, suppose that trees $v_1, \ldots, v_n$ adjoin into tree $u$ at $a_1, \ldots, a_n$. Each $v_i$ contains a (unique) foot node $f_i$, which has a mother $m_i$ that is the projection of some LI. This holds because $v_i$ consists of at least two nodes and we preprocessed all elementary trees in order to make them projective. Note that for every TAG, adjoining $v$ at the foot node of $u$ is equivalent to adjoining $u$ at the root of $v$, so we do not consider the case where $a_i$ is a foot node. Adjunction at substitution nodes is superfluous for the same reason and usually forbidden. The only remaining cases, then, are for $a_i$ to be an LI or some projection thereof.

The translation of $v_i$ is less involved than that of $u$, so it makes sense to discuss it first. For $\diamond$ some distinguished symbol, $\mu(f_i) := \diamond$. The $\diamond$ is used to mark the foot node $f_i$ for deletion by $\tau$. As for $m_i$, we set $\phi(m_i) := \langle \circ, move, +, connection(v_i) - 1 \rangle$, where $\circ$ is some arbitrary feature name. This turns $m_i$ into a Move node that will serve as the landing site for a subtree in $u$ as sketched in Fig. 2. The function $connection(v_i)$ determines the size of the movement feature and returns integer $n$ iff the Merge node immediately dominating $\square_i$ in $\hat{\mu}(u)$ is the $n$-th projection of some node. In order to understand why this is the desired value, we need to look at the translation of $u$ next.

For each $a_i$ in $u$, a Merge node must be added immediately above it that serves in selecting $v_i$. These Merge nodes require new features on $\hbar(a_i)$. Moreover, $\hbar(a_i)$ must also carry the requisite number of Move features to undergo lowering, and all of them must have the correct size value. This requires special definitions for $\mu(a_i)$, $\phi(a_i)$, and $\mu(\hbar(a_i))$, respectively. I only discuss the case where $a_i$ is an interior node. The procedure for $a_i$ an LI is almost the same (see Fig. 3).

The value of $\phi(a_i)$ is the feature string $[\phi]\, \langle \ell(\hbar(v_i)), merge, +, \lambda \rangle$, where $[\phi]$ is what $\phi$ would return at $a_i$ if it was a normal non-terminal (see Sec. 3.3.2). The corresponding part of the Minimalist derivation is $\mu(a_i) := \text{Merge}(\square_i, [\mu])$, where $[\mu]$ is what $\mu$ would return at $a_i$ if it was a normal non-terminal. These notational contortions emulate the effect of treating $a_i$ as usual except that a Merge node is added above it that $v_i$ can be attached to.

Due to the complexities of the movement mechanism in the specification of $\hbar(a_i)$, this part is best split out into a separate component. Let $\xi(x, n) := \langle \circ, move, -, \delta_n \rangle$ if some $a_i$ is the $n$-th projection of $x$, and $\varepsilon$ otherwise. The integer $\delta_n$ is given by $|\, \{1 \leq j < n \mid \xi(x, j) \neq \varepsilon\}\, |$. As a more compact notation, let $\xi_i^n(x) := \xi(x, n)\ \xi(x, n-1) \cdots \xi(x, i+1)\ \xi(x, i)$ be the string concatenation of the outputs of $\xi(x, i), \ldots, \xi(x, n)$ listed in reverse. Now for $\pi(\hbar(a_i)) := p_1 \cdots p_z$, we define $\mu(\hbar(a_i)) := \ell(\hbar(a_i)) :: \phi(p_1) \cdots \phi(p_z)\ \phi(\hbar(a_i))\ \xi_1^z(\hbar(a_i))$.

The peculiar formula for $\delta_n$ stems from a complication in how projections should be counted. Recall from the discussion in Sec. 2 that a fea-

ture's size plays an essential role in the mapping from derivation trees to derived trees by virtue of picking out the root of the moved subtree. For our purposes, a size value $j$ should refer to the $j$-th projection of $\hbar(a_i)$ in $u$. But the mapping to derived trees operates over Minimalist derivation trees, and since $\mu$ adds new projections for $\hbar(a_i)$, that node's $j$-th projection in $u$ might be its $k$-th projection in the corresponding Minimalist derivation tree fragment $\hat{\mu}(u)$, $k > j$ (see also Fig. 2). In order to correctly compute $j$, though, it suffices to know how many of the lower projections have been expanded into two Merge nodes rather one, which can be deduced from the number of values that aren't mapped to the empty string by $\xi$.

There is still one minor problem pertaining to adjunction at the root node of $u$ when $u$ is also the root of the derivation tree $t$. In this case, $\hbar(u)$ won't get its category feature $\langle l, merge, - \rangle$ checked, so it cannot undergo movement. This can be fixed by adding another LI on top of the derivation, as was done for the example grammar in Fig. 1: For $a_i$ the root of $u$, $u$ the root of $t$, and $f$ some feature name, $\mu(a_i) :=$ Merge(Merge, $\varepsilon :: \langle \ell(\hbar(a_i)), merge, +, \lambda \rangle \langle f, merge, - \rangle$).

With these pitfalls out of the way, it only remains for us to assemble the individual outputs of $\mu$ into a coherent derivation: $\tau(\langle u, a \rangle (\langle v_1, a_1 \rangle, \ldots, \langle v_n, a_n \rangle))$ is the result of deleting every node labeled $\diamond$ in $\mu(u)$ and replacing each $\square_i$ as before.

### 3.3.5 Putting it All Together

The full specification of the translation procedure is given in Fig. 3. For a little bit of extra rigor, $\phi$ has been split into two functions, while space restrictions forced the use of additional notational shorthands. The result of removing all instances of $\diamond$ from tree $u$ is denoted by $u \setminus \diamond$, and $u \leftarrow [t_1, \ldots, t_n]$ is the tree obtained from $u$ by replacing each $\square_i$ by $t_i$. Given a feature $f$, $\omega(f)$ is the second component of $f$, i.e. the type of operation $f$ regulates.

Several parameters must be kept track of in order for the functions to be well-defined. They are the current tree $u$, and its derivational daughters $v_1, \ldots, v_i$ with their respective addresses $a_1, \ldots, a_n$. In addition, computing $connection(v_i)$ requires access to the derivational mother $o$ of $u$, or alternatively, storage of the value during the computation of $o$. The (locally bounded) passing around of these parameters is not made explicit in the definitions in order to minimize notational clutter.

### 3.3.6 Correctness

The correctness of the translation for single node derivations as well as instances of substitution has already been established. It should also be clear that all cases of adjunction yield an output, so $\tau$ is at least well-defined. Moreover, these outputs are definitely Minimalist derivation trees. It still needs to be shown, though, that they are well-formed and that the intended derived tree can be obtained from them. The former depends on whether **Move** and **SMC** are satisfied. The latter follows rather straight-forwardly if this is the case.

The definition of Reset Lowering entails that every Move node $m$ is an occurrence for at most one LI. Otherwise, there would be two LIs $l \neq l'$ whose $k$-th projections both c-command $m$ without at least one c-commanding the other, which is impossible ($k$ is the size value of the feature associate of $m$). That there is at least one LI $l$ for every $m$ follows from the definition of $\mu$, which inserts the tree containing $m$ as the sister of the $k$-root of $l$. The corresponding negative movement is also instantiated on $l$ at the right position of the feature string. Thus **SMC** always holds.

**Move** is satisfied, too, because LI $l$ contains a negative movement feature iff it selects a tree containing the required Move node $m$. Said tree cannot contain another LI $l'$ that $m$ is a potential occurrence for, because then there would also be another Move node $m'$. This argument can be continued until eventually there must be a Move node that isn't an occurrence for any LI, or for more than one. Either case violates **SMC**, yielding a contradiction.

As a result of all this, the Move node in tree $\hat{\mu}(v_i)$ is an occurrence of $\hbar(a_i)$, and for every negative Move feature on an LI there is an occurrence $m$ such that lowering to $m$ yields the same derived tree as adjunction. That this ensures the generation of the correct derived tree (*modulo* empty nodes left behind by movement) only requires a short proof by induction.

As a quick side remark, the use of only one feature name for movement features in the translation might provoke the conjecture that Reset Lowering

$$\varphi(x) := \begin{cases} \varphi(x_1) \cdots \varphi(x_n) & \text{if } x \text{ is a string of nodes } x_1, \ldots, x_n, \\ \langle \circ, \mathit{move}, +, \mathit{connection}(v_i) - 1, \lambda/\rho \rangle & \text{if foot node } f_i \text{ is the left/right daughter of } x, \\ \langle \ell(\hbar(v_i)), \mathit{merge}, +, \lambda/\rho \rangle & \text{if } a_i \text{ is a substitution node and the left/right daughter of } x, \\ \langle \ell(\hbar(y)), \mathit{merge}, +, \lambda/\rho \rangle & \text{if the daughter } y \text{ of } x \text{ is not part of the same projection}, \\ \langle \ell(x), \mathit{merge}, - \rangle & \text{if } x \text{ is an LI.} \end{cases}$$

$$\phi(x) := \begin{cases} \varphi(x)\, \langle \hbar(v_i), \mathit{merge}, +, \lambda \rangle & \text{if } x = a_i \text{ and } x \text{ is an interior node}, \\ \langle \hbar(v_i), \mathit{merge}, +, \lambda \rangle\, \varphi(x) & \text{if } x = a_i \text{ and } x \text{ is an LI}, \\ \varphi(x) & \text{otherwise.} \end{cases}$$

$$\xi(x, n) := \begin{cases} \langle \circ, \mathit{move}, -, n + |\, \{1 \leq j < n \mid \xi(x, j) \neq \varepsilon\}\, | \rangle & \text{if some } a_i \text{ is the } n\text{-th projection of } x, \\ \varepsilon & \text{otherwise.} \end{cases}$$

$$\mu(x) := \begin{cases} \text{Merge}(\text{Merge}, \varepsilon :: \langle \ell(\hbar(x)), \mathit{merge}, +, \lambda \rangle\, \langle f, \mathit{merge}, - \rangle) & \text{if } x = a_i \text{ is the root of } u \text{ \& } u \text{ the root of } t, \\ \ell(x) :: \phi(\pi(x))\, \phi(x)\, \xi(x, n) \cdots \xi(x, 1) & \text{if some } a_i \text{ equals } x \text{ or is a projection of } x, \\ \text{Merge}(\square_i, \omega(\phi(x))) & \text{if } x = a_i \text{ and } x \text{ is an interior node}, \\ \text{Merge}(\square_i, \phi(x)) & \text{if } x = a_i \text{ and } x \text{ is an LI}, \\ \square_i & \text{if } x \text{ is a substitution node}, \\ \diamond & \text{if } x \text{ is a foot node}, \\ \omega(\phi(x)) & \text{if } x \text{ is an interior node}, \\ \ell(x) :: \phi(\pi(x))\, \phi(x) & \text{if } x \text{ is an LI.} \end{cases}$$

$$\hat{\mu}(x(x_1, \ldots, x_n)) := \begin{cases} \mu(x) & \text{if } n = 0, \\ \mu(x)(\mu(x_1), \ldots, \mu(x_n)) & \text{otherwise.} \end{cases}$$

$$\tau(\langle \nu(e), a \rangle\, (t_1, \ldots, t_n)) := \begin{cases} \hat{\mu}(e) \setminus \diamond & \text{if } n = 0, \\ (\hat{\mu}(e) \setminus \diamond) \leftarrow [\tau(t_1), \ldots, \tau(t_n)] & \text{otherwise.} \end{cases}$$

Figure 3: Definition of the translation from a TAG derivation $t$ to the corresponding Minimalist derivation; where multiple conditions are satisfied at once, only the highest one applies; see Sec. 3.3.5 for an explanation of notation.

MGs satisfying this condition are weakly equivalent to TAGs (as Graf, 2012 erroneously does). However, even with just one feature name Reset Lowering MGs can still generate $a_1^n, \ldots, a_k^n$ for arbitrary $k$. This is so because features with different size values are considered distinct by **SMC** despite their identical feature name. For example, the grammar in Fig. 1 can be made to use only one feature name by having $b$ merge with an empty head before selecting $a$ so that the size value of its movement feature can always be one bigger than that of $a$'s. If this loop-hole can be patched, though, a weak equivalence proof seems feasible.

### 3.4 Step 3: Intersection

The correctness of the translation only entails that the output of $\tau$ applied to a TAG derivation is a Minimalist derivation that can be mapped to the intended derived tree. Nothing so far guarantees that translating the TAG derivation tree language actually yields a Minimalist derivation tree language (MDTL), as not every set of well-formed Minimalist derivations is a well-formed MDTL. The closure of MDTLs under intersection with regular tree languages (Graf, 2011; Kobele, 2011), however, makes it straight-forward to construct an MDTL from the output of the translation.

It is a well-known fact that the image of a regular tree language under a linear transduction is also regular (Gécseg and Steinby, 1984), and $\tau$ is exactly such a (non-deterministic) transduction.[2] As TAG derivation tree languages are regular, so is the tree language produced by $\tau$ applied to a given TAG $G_T$. Now let $G_M$ be the MG whose lexicon contains every LI that occurs in some tree

[2] Given a look-ahead of 1, it is even deterministic.

in $\tau(G_T)$. The intersection of $G_M$'s derivation tree language with $\tau(G_T)$ yields the MDTL of some MG that generates the same derived trees as $G_T$.

## 4 Conclusion

I have given a productive proof via a translation procedure that MGs with Reset Lowering instead of standard Move can generate all TAG tree languages. I also showed that they are more powerful than TAGs on a string level, but a stronger version of **SMC** might actually be sufficient to make the two formalisms equivalent in this respect. For future work, it will be interesting to see if the translation can be extended to generalizations of TAG such as the one in Rogers (2003). If so, this would be an indication that the relation between adjunction and Reset Lowering isn't merely accidental but provides a fresh perspective on TAG.


## Acknowledgments

My sincerest thanks go to Ed Stabler and Michael Freedman for earlier discussions of the material, as well as the three anonymous reviewers for their helpful comments that lead to several improvements in the presentation of the material.


## References


A. Fujiyoshi and T. Kasai. 2000. Spinal-formed context-free tree grammars. *Theory of Computing Systems*, 33:59–83.

Thomas Graf. 2011. Closure properties of minimalist derivation tree languages. In Sylvain Pogodalla and Jean-Philippe Prost, editors, *LACL 2011*, volume 6736 of *Lecture Notes in Artificial Intelligence*, pages 96–111.

Thomas Graf. 2012. Movement-generalized minimalist grammars. In Denis Béchet and Alexander J. Dikovsky, editors, *LACL 2012*, volume 7351 of *Lecture Notes in Computer Science*, pages 58–73.

Ferenc Gécseg and Magnus Steinby. 1984. *Tree Automata*. Academei Kaido, Budapest.

Henk Harkema. 2001. A characterization of minimalist languages. In Philippe de Groote, Glyn Morrill, and Christian Retoré, editors, *Logical Aspects of Computational Linguistics (LACL'01)*, volume 2099 of *Lecture Notes in Artificial Intelligence*, pages 193–211. Springer, Berlin.

Aravind Joshi, K. Vijay-Shanker, and David Weir. 1991. The convergence of mildly context-sensitive grammar formalisms. In P. Sells, Shieber S. M., and T. Warsow, editors, *Foundational Issues in Natural Language Processing*, pages 31–81. MIT Press, Cambridge, MA, USA.

Gregory M. Kobele, Christian Retoré, and Sylvain Salvati. 2007. An automata-theoretic approach to minimalism. In James Rogers and Stephan Kepser, editors, *Model Theoretic Syntax at 10*, pages 71–80.

Gregory M. Kobele. 2011. Minimalist tree languages are closed under intersection with recognizable tree languages. In Sylvain Pogodalla and Jean-Philippe Prost, editors, *LACL 2011*, volume 6736 of *Lecture Notes in Artificial Intelligence*, pages 129–144.

Jens Michaelis. 1998. Derivational minimalism is mildly context-sensitive. *Lecture Notes in Artificial Intelligence*, 2014:179–198.

Jens Michaelis. 2001. Transforming linear context-free rewriting systems into minimalist grammars. *Lecture Notes in Artificial Intelligence*, 2099:228–244.

Uwe Mönnich. 1999. On cloning context-freeness. In Hans-Peter Kolb and Uwe Mönnich, editors, *Mathematics of Syntactic Structure*, pages 195–231. Walter de Gruyter, Berlin.

Uwe Mönnich. 2006. Grammar morphisms. Ms. University of Tübingen.

Uwe Mönnich. 2007. Minimalist syntax, multiple regular tree grammars and direction preserving tree transductions. In James Rogers and Stephan Kepser, editors, *Model Theoretic Syntax at 10*, pages 83–87.

James Rogers. 2003. Syntactic structures as multidimensional trees. *Research on Language and Computation*, 1(1):265–305.

Edward P. Stabler. 2011. Computational perspectives on minimalism. In Cedric Boeckx, editor, *Oxford Handbook of Linguistic Minimalism*, pages 617–643. Oxford University Press, Oxford.

K. Vijay-Shanker and David J. Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27:511–545.

David Weir. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania. Available as Technical Report MS-CIS-88-74 of the Department of Computer and Information Sciences, University of Pennsylvania.