

Derivational Minimalism

Edward Stabler

University of California, Los Angeles, CA 90024-1543, USA

Abstract. A basic idea of the transformational tradition is that constituents move. More recently, there has been a trend towards the view that all features are lexical features. And in recent “minimalist” grammars, structure building operations are assumed to be feature driven. A simple grammar formalism with these properties is presented here and briefly explored. Grammars in this formalism can define languages that are not in the class of languages definable by tree adjoining grammars.

1 Minimalist grammars

Adapting the general framework of [13], a grammar is regarded as a specification of a lexicon and generating functions for building complex expressions:

A grammar $G = (V, Cat, Lex, \mathcal{F})$, where
 V is a set, (non-syntactic features)
 Cat is a set, (syntactic features)
 Lex is a set of expressions built from V and Cat , (the lexicon)
 \mathcal{F} is a set of partial functions from (the generating functions)
tuples of expressions to expressions

In the minimalist grammars presented here, expressions will be a certain kind of finite, binary ordered trees with labels only at the leaves. The *language* defined by such a grammar is the closure of the lexicon under the structure building functions, $L(G) = CL(Lex, \mathcal{F})$.

1.1 Trees

A finite tree is given by a set of nodes with a dominance relation of the usual kind, $\tau = (N_\tau, \triangleleft_\tau^*)$. We leave off the subscripts when no confusion will result. We use the following notation:

$x \triangleleft y$	x is the parent of y
$x \triangleleft^+ y$	x properly dominates y
$x \triangleleft^* y$	x dominates y

The root of any tree τ is the minimal element of N , and the *leaves* are the maximal elements. That is, the set of leaves $L_\tau = \{x \mid \neg(\exists y) x \triangleleft y\}$.

Constituents are standardly picked out with the dominance relation. The constituent with root node x is the part of the structure that x dominates. The set of nodes dominated by x is $\uparrow x = \{y \in N \mid x \triangleleft^* y\}$. So then for any leaf x , $\uparrow x = \{x\}$. The subtree τ_x with root x is

$$\tau_x = (\uparrow x, (\triangleleft^* \upharpoonright \uparrow x)).$$

Our trees are linearly ordered by an additional *precedence* relation, $\tau = (N_\tau, \triangleleft_\tau^*, \prec_\tau^*)$. We use the following notation:

$$\begin{array}{ll} x \prec y & x \text{ immediately precedes } y \\ x \prec^+ y & x \text{ properly precedes } y \\ x \prec^* y & x \text{ precedes } y \end{array}$$

We will assume that for any two distinct nodes, either one dominates the other or one precedes the other (but never both), and we assume precedence is inherited through dominance in the usual way:

$$(\forall w, x, y, z) (x \prec^* y \wedge x \triangleleft^* w \wedge y \triangleleft^* z) \rightarrow (w \prec^* z)$$

One additional relation is added to tree structures in order to obtain appropriate objects for minimalist grammar. When two constituents combine, one of them always “projects over” the other.¹ To represent that a determiner d *projects over* a noun n to form a DP, we let $<$ represent this relation between the sisters in a tree, writing $d < n$, and we can adopt the usual notation for the transitive and reflexive, transitive closures of this relation, respectively:

$$\begin{array}{ll} x < y & x \text{ immediately projects over } y \\ x <^+ y & x \text{ properly projects over } y \\ x <^* y & x \text{ projects over } y \end{array}$$

Regarding the projection relation $<$ as reflexive, we can say that whenever a node has a child, there is a unique child that projects over every child of that parent:

$$(\forall x) ((\exists y)(x < y)) \rightarrow ((\exists y)(\forall z)(x < z \rightarrow y < z))$$

¹ This approach is inspired by Chomsky’s [7, p245] suggestion, “If constituents α, β of K have been formed in the course of computation, one of the two must project – say, α . At the LF interface, maximal K is interpreted as a phrase of the type α (e.g., as a nominal phrase if [the head of K] $H(K)$ is nominal); and it behaves in the same manner in the course of computation.” Note that this assumption is empirically loaded, and far from obviously correct. The relevant properties of a complex are not in general determined by just one of the immediate subconstituents. In the minimalist tradition, this fact is accommodated in part by allowing, in certain cases, the projecting head to incorporate features of its sister, and by allowing properties of subconstituents (namely, in our formalization, the $-x$ subconstituents) to influence the elaboration of structure.

Adding $<^*$ to our tree structures, they have the form $\tau = (N_\tau, \triangleleft_\tau^*, \prec_\tau^*, \triangleleft_\tau^*)$. These are the basic structures of our minimalist grammar.

In terms of such structures, we can define some structural notions that are relevant for the specification of the grammar. Given any tree $\tau = (N, \triangleleft^*, \prec^*, \triangleleft^*)$ and any $x, y \in N$, x is a *head of y* iff either

y is a leaf and $x = y$, or

$(\exists z) (y \triangleleft z \wedge (\forall w)(y \triangleleft w \rightarrow z \triangleleft^* w) \wedge x \text{ is a head of } z)$.²

The following basic results are easily obtained from these definitions.

Proposition 1. *Every head is a leaf, and every leaf is a head of itself.*

Proposition 2. *If x is a head of y then $y \triangleleft^* x$.*

Proposition 3. *Every node y has a unique head x .*

We define the *maximal projection of head x* to be the least node y with head x . That is, the maximal projection y of x is the node $y \in \{z \in N \mid \text{the head of } z \text{ is } x\}$ such that there is no w in this set with the property that $w \triangleleft^+ y$.

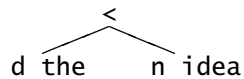
Finally, for convenience, it is useful to call a node x a *specifier of head y* iff

- i. x is a maximal projection,
- ii. the parent of x has head y , and
- iii. $x \triangleleft^+ y$.

And similarly we call a node x a *complement of head y* iff

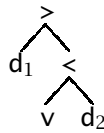
- i. x is a maximal projection,
- ii. the parent of x has head y , and
- iii. $y \triangleleft^+ x$.

In the linguists' standard depiction of a labeled, ordered tree, the minimal element is at the top, with immediate dominance shown by arcs and linear precedence shown by left-to-right order. Our trees have a third relation, so there is a question about how to depict them. Since we will only be considering binary branching trees, a simple idea presents itself. Since only the leaves are labeled, we can indicate at each internal node, which of the two daughters projects over the other.

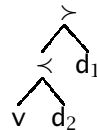


Notice that in this tree, $<$ and \triangleleft coincide. They do not coincide in a tree like the following (which is the way we will represent what linguists call “VP shells”):

² Again, this treatment of heads is inspired by one of Chomsky's [7, pp245-245] suggestions: “It is natural, then, to take the label of [the complex] K [formed from α and β] to be not α itself, but rather $H(K)$, a decision that leads to technical simplification.”



It is perhaps enlightening to observe that, rather than letting the left-right dimension of our graphical depiction represent \prec^* we could have let it represent \prec , marking \prec explicitly in the tree to get a different depiction of the same structure:



In this kind of presentation, the projecting category is always on the left, while precedence gets explicitly marked. However, we will use the former convention, explicitly indicating the “projects over” relation.

1.2 Features

Vocabulary items are broken into pronounced parts and interpreted parts, $V = (P \cup I)$. The pronounced, *phonetic features* are indicated by placing slashes around the standard orthographic representation of the word. So for example, we will consider the set,

$$P = \{/marie/, /quechua/, /speaks/, /believes/, \dots\}.$$

The interpreted, *semantic features* are indicated by placing parentheses around standard orthographic representations, as in

$$I = \{(marie), (quechua), (speaks), (believes), \dots\}.$$

As an abbreviation, we will sometimes use, for example, *marie* to represent the sequence $/marie/(marie)$, and similarly for other vocabulary when corresponding phonetic and interpreted features are simultaneously available.

The *syntactic features* are usefully partitioned into four kinds,

$$Cat = (base \cup select \cup licensors \cup licensees).$$

We introduce each kind of feature in turn.

The basic syntactic categories,

$$base = \{c, t, v, d, n, \dots\},$$

represent the familiar categories: complementizer, tense, verb, determiner, noun, and so on. For each type of category, there are three ways of “selecting” a constituent of that category. We will use the feature =n to indicate

the simple selection of a noun phrase; the feature =N indicates that a noun phrase is selected and furthermore the phonetic features of the head of the noun phrase are moved to be suffixed to the sequence of phonetic features of the selecting head; and finally the feature N= indicates that a noun phrase is selected and furthermore the phonetic features of the head of the noun phrase are moved to be prefixed to the sequence of phonetic features of the selecting head. So the set of possible selection features is:

$$select = \{=x, =X, X= \mid x \in base\}.$$

The categories and selection features interact in local, head-head configurations. As we will see, the structure building rule *merge* will cancel a requirement =x, =X, or X= by combining the constituent with this requirement with a constituent of category x.

The remaining kinds of features are those involved in phrasal movement, which always involves a head-specifier relationship. In particular, a head will assign a feature +f to a -f constituent that moves covertly to its specifier (movements are explained below), or else a feature +F is assigned to a -f constituent that moves overtly to its specifier. So for example, we can have features that indicate requirements like

$$licensees = \{-case, -wh, \dots\},$$

with corresponding features that indicate assignments, like

$$licensors = \{+case, +CASE, +wh, +WH, \dots\}.$$

These features +f and +F are the triggers for phrasal movement. The structure building rule *move* will cancel a requirement -f by moving the phrase with this feature to the specifier of a head that has either +f or +F.

No limit has been indicated for the numbers of elements in *base*, *licensees* and *licensors*, but linguists typically assume that these sets will be finite and small. And then, if there are *k* *licensees* then there will be *2k* *licensors*.

The features written with capital letters =X, X=, +X are sometimes called *strong features*. (Cf. [22], [7]). Strong features cause phonetic material to move along with the affected syntactic features. The strong/weak distinction provides some control over how much structure moves along with the features that are canceled by the structure building rule. This carrying along of more or less additional structure is sometimes called “pied piping.”

We are now in a position to define what counts as a linguistic expression in our minimalist grammars. An *expression* is a finite, binary labeled ordered tree $\tau = (N_\tau, \triangleleft_\tau^*, <_\tau^*, <_\tau^*, Label_\tau)$, where the domain of *Label* is just the leaves *L* of the tree and the range is a regular set,

$$Label : L \rightarrow select^*(licensors)select^*(base)licensees^*P^*I^*.$$

The rationale for this particular ordering of features in the label will become clear below: the syntactic features are canceled in order.

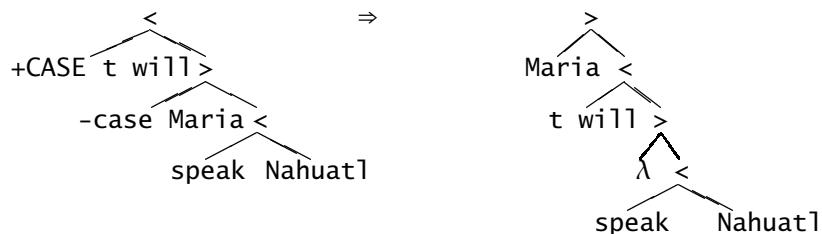
Notice that $Rng(Label) \subset (V \cup Cat)^*$. Furthermore, $\lambda \in Rng(Label)$; that is, “empty nodes” can occur in expressions.

i, below, may have a structure of the sort indicated in ii, where the universally quantified phrase has raised to take wide scope over some student:

- i. /some student likes every teacher/
- ii. (every teacher) some student likes /every teacher/

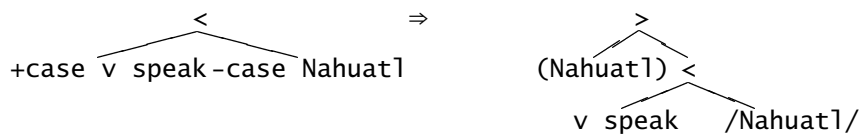
To build constructions like these, *move* is formalized as a unary operation on expressions.

Overt movement occurs when *move* applies to a tree whose head has an initial feature +X, and where the tree contains exactly one -x feature. This restriction to one -x feature is a kind of “economy” condition: it prevents two embedded -x constituents from competing for the first available specifier of a +x or +X head, so that a -x feature is always canceled by the “closest” possible +x or +X feature. The -x head may be arbitrarily deep in the tree. The movement raises the whole maximal projection of the -x head up to the specifier of the +X head.



Overt movement leaves behind just a node labeled by the empty sequence of features λ .

Covert movement is similar, except that it is triggered by an initial feature +x (rather than a “strong” +X), and it does not move the whole -x phrase, but only the interpreted and syntactic features. The phonetic features are left behind, as we see in the following “covert object shift:”



In this simple example, the -x constituent is a sister of the +x constituent, but as already mentioned, it can be arbitrarily deep in the tree.

2 Examples and basic properties

We explore the effect of varying the lexicon of the minimalist grammars, keeping $\mathcal{F} = \{merge, move\}$ fixed.

2.1 A simple SVO language

This first example is inspired by [7].⁵ It embodies particular grammatical assumptions (e.g. the assumption that a language like English may have a “covert object shift”), which other MG grammars will not share.

Let Lex_1 be the set containing the following 12 expressions:

d -case maria	d -case quechua	
=n d -case some	=n d -case every	
n student	n language	
=d +case =d v speaks	=c +case =d v believes	
=v +CASE t		
=t c	=t c -case	=t c -case /that/

The following 8 step derivation yields a structure that is pronounced: /some linguist speaks every language/. The derivation begins by merging the following two lexical elements:

from the lexicon:

=n d -case every n language

Applying *merge* to these two elements, in this order, yields a determiner phrase:

step 1 merge:

```

      <
     / \
    /   \
d -case every language
  
```

This is a determiner phrase that needs case. It can be merged with the lexical item that is pronounced /speaks/ to yield the following tree:

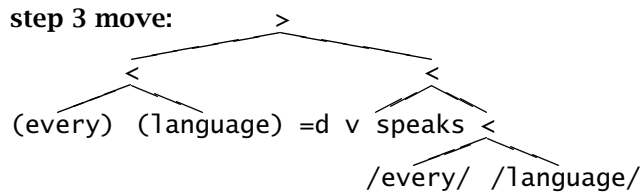
step 2 merge:

```

      <
     / \
+case =d v speaks <
                / \
              -case every language
  
```

⁵ In particular, we are roughly following the proposal [7, p358] made in the following passage: “...both Specs are within the minimal domain of v , so either is available for θ -marking the external argument of a transitive verb. Suppose that we allow this possibility, so that the outer Spec can host the external argument...Obj can only raise to the inner Spec ...to check the strength feature and undergo overt Case marking. If overt object raising takes place, then Subj will be merged in the outer Spec to receive the external θ -role provided by the configuration...On these assumptions, it follows that Subj always c-commands Obj within IP.” In fact, this kind of proposal has been familiar for some time. It was proposed, for example, in [17] that the position where the object receives case is below the subject, and nevertheless within VP.

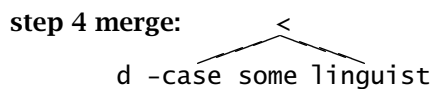
The head of the result of step 2 must assign case to its specifier, and there is exactly one *-case* element inside the tree, so *move* applies in step 3 to covertly move the object, leaving the pronounced material behind. (Languages with overt object shift are considered below.)



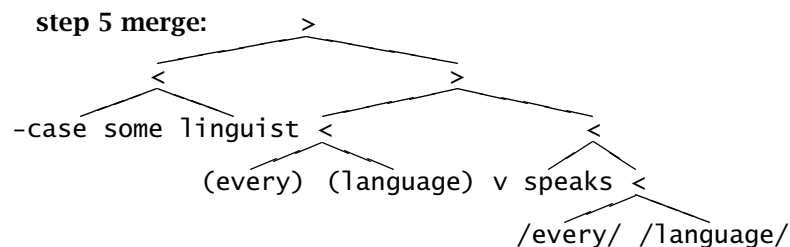
The first feature of the head of the verb phrase formed in step 2 is the selection feature for the subject, so we first build a subject with two new elements from the lexicon:

from the lexicon:

=n d -case some n linguist

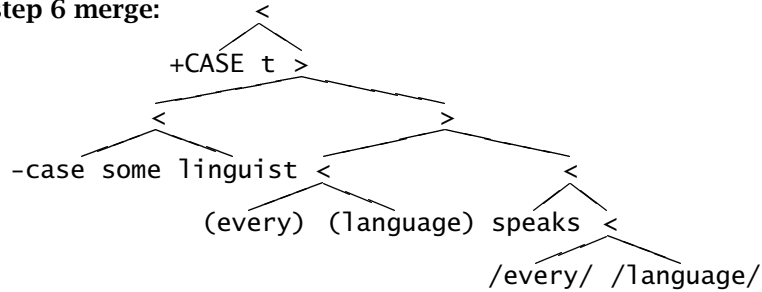


We can now apply merge to the result of step 3 and the result of step 4, in that order, to yield the following “VP shell:”



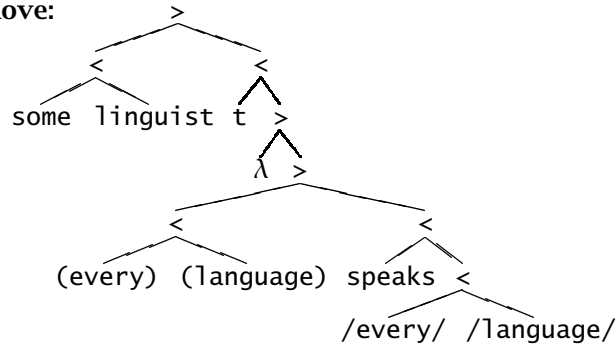
The result of step 5 is a verb phrase, so it is selected by the phonetically empty lexical item that has category *t*. Applying merge to this tense element and the result of step 5, in that order, we obtain:

step 6 merge:



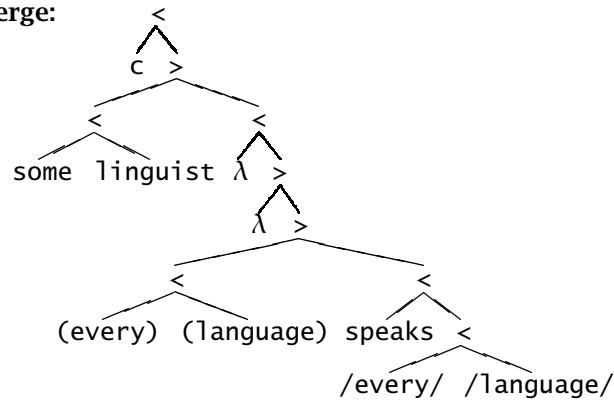
Since the result of step 6 is a tense phrase that must assign case to an overt specifier, move can apply to move the -case element to specifier position:

step 7 move:



The result of step 7 is a tense phrase, and since tense is selected by the lexical complementizer, *merge* can apply to yield the following tree:

step 8 merge:



This last tree is *well formed* in the sense that it has no outstanding syntactic features at all except the single category symbol *c* at its head. (We could imagine that the *c* is deleted when this structure is integrated into

the discourse.) In this case, we will say $MG_1 \Rightarrow^*$ some linguist speaks every language, and we will refer to the language of the grammar, the set of phonetic strings that have well formed syntactic structures as

$$strings(MG_1) = \{s \mid MG_1 \Rightarrow^* s\}.$$

Proposition 4. $MG_1 \Rightarrow^*$ some linguist speaks every language
(Furthermore, there is exactly one such derivation.)

Proposition 5.

$MG_1 \Rightarrow^*$ maria believes that some student speaks every language

Proposition 6. $MG_1 \not\Rightarrow^*$ speaks every

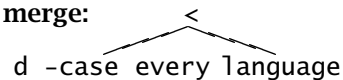
2.2 SOV languages

If we modify MG_1 so that the verbs are strong case assigners, we get phonetic forms with the SOV word order. Let Lex_2 be the set containing the following 11 expressions:

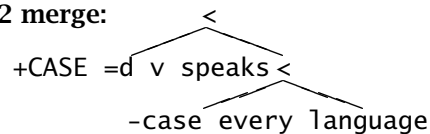
d -case maria	d -case quechua
=n d -case some	=n d -case every
n student	n language
=d +CASE =d v speaks	=c +CASE =d v believes
=v +CASE t	
=t c	=t c -case

We present the steps of one sample derivation:

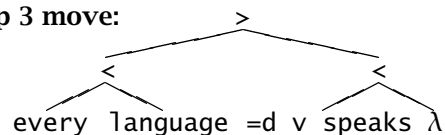
step 1 merge:



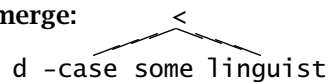
step 2 merge:

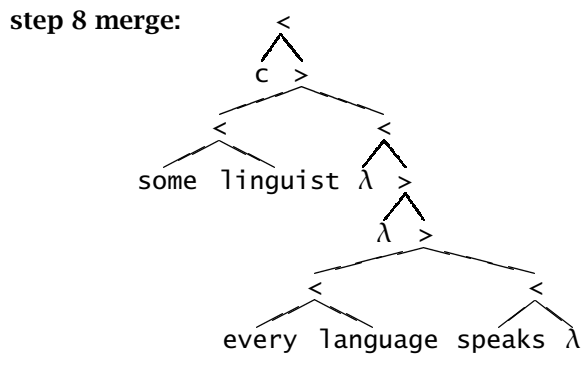
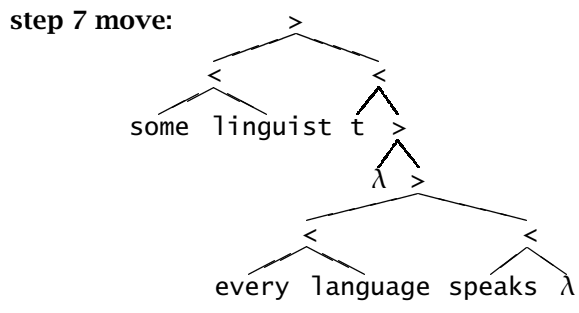
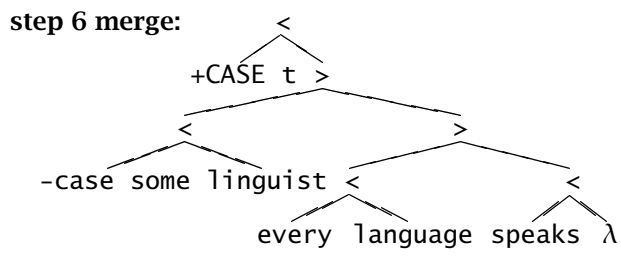
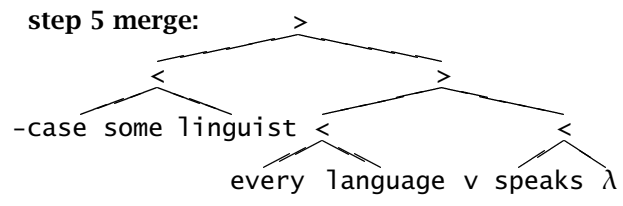


step 3 move:



step 4 merge:

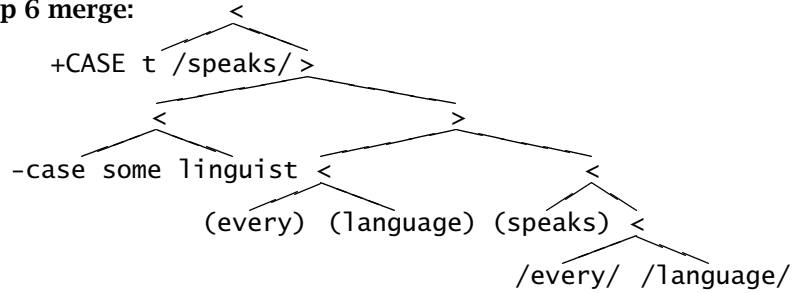




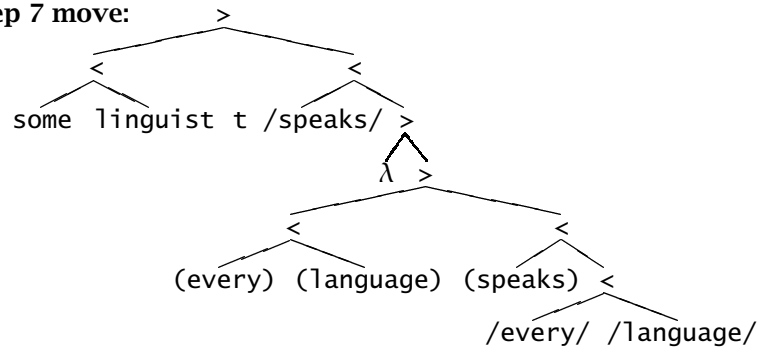
2.3 VSO languages

If we modify MG_1 so that c and t strongly select their complements, triggering head movement with right adjunction, then we get phonetic forms with VSO word order. Let Lex_3 be the set containing the following 11 expressions:

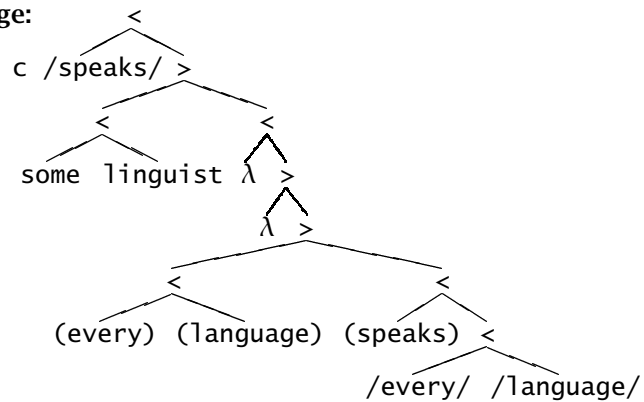
step 6 merge:



step 7 move:

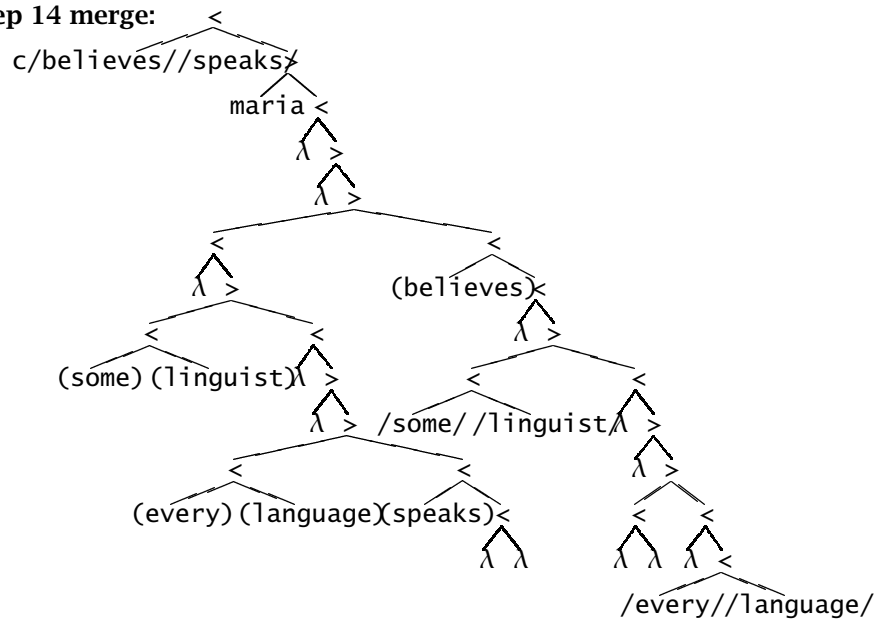


step 8 merge:



Notice that since *believes* strongly selects its clausal complement, we will get “verb clusters” in the highest c position with this grammar:

step 14 merge:



The pronounced positions of the verbs in these clusters exhibit “crossing” dependencies with respect to the pronounced positions of their respective subjects. Using subscripts to indicate which subjects go with which verbs, we derive the following orders of basic constituents in the phonetic form:

$$\begin{array}{c}
 V_1 S_1 O_1 \\
 V_1 V_2 S_1 S_2 O_2 \\
 V_1 V_2 V_3 S_1 S_2 S_3 O_3 \\
 \dots
 \end{array}$$

2.4 WW languages

Given the previous example, it is no surprise that the MG formalism allows the definition of simple “copying languages.” That is, a certain kind of “reduplication” can be defined even though there is no copying operation in the grammar. Consider for example the grammar defined by the following lexicon of 9 elements, Lex_4 :

=V ₁ c	=V ₂ c	c
=d ₁ v ₁ /1/	=d ₂ v ₂ /2/	
=C =d ₁ v ₁ /1/	=C =d ₂ v ₂ /2/	
d ₁ /1/	d ₂ /2/	

The exploration of derivations from this lexicon is left to the reader (particularly because the examples in the next two sections are of greater interest).

Proposition 7. $strings(MG_4) = \{ww \mid w \in \{1,2\}^*\}$. (cf. also [9])

The proof of this claim is based on the fact, easily seen from the lexicon, that for every v_1 introduced in a derivation there must be exactly one corresponding d_1 , a for every v_2 introduced in a derivation there must be exactly one corresponding d_2 . Furthermore, all the verbs will raise to right adjoin to the highest c, in order, leaving the corresponding d's in their original positions.

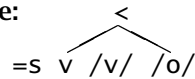
2.5 $c^n v^n s^n o^n$

Like other formalisms that can define ww languages, the MG formalism can also define four counting dependencies. Let Lex_5 be the set containing the following 7 expressions:

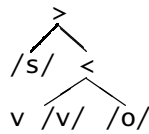
=V c -case /c/ =V c /c/ c
s /s/
o /o/
=o =s v /v/ C= =o +CASE =s v /v/

We present one example derivation:

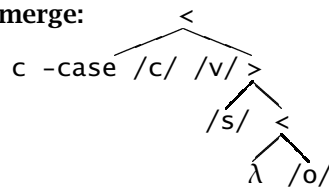
step 1 merge:



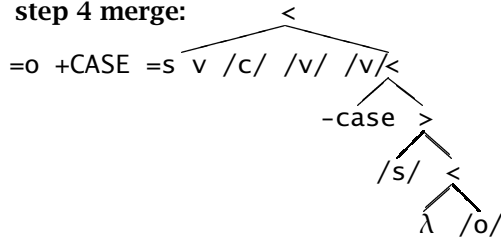
step 2 merge:



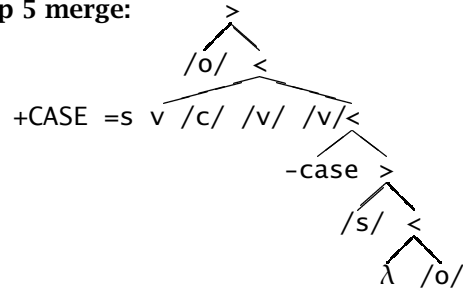
step 3 merge:



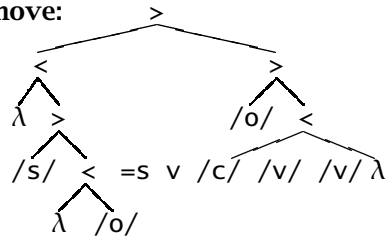
step 4 merge:



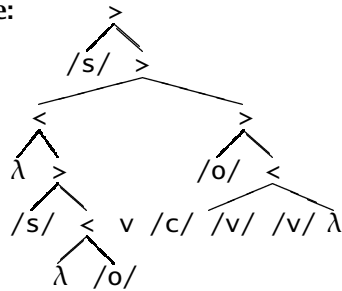
step 5 merge:



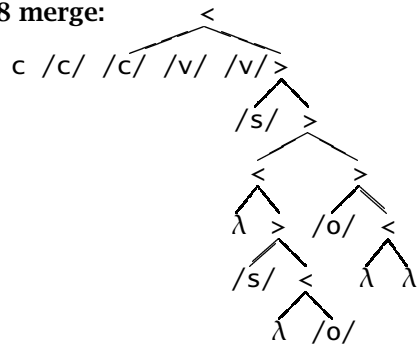
step 6 move:



step 7 merge:



step 8 merge:



Proposition 8. $strings(MG_5) = \{c^n v^n s^n o^n \mid n \geq 0\}$.

The proof of this proposition is based on the fact that any derivation of a well-formed structure must use one of the lexical complementizers: (i) $=V\ c\ -case\ /c/$ or (ii) $=V\ c\ /c/$ or (iii) c . The derivations in case (iii) are trivial. Any use of (i) in a derivation must have a corresponding use of $C= =o\ +CASE\ =s\ v\ /v/$ to allow for cancellation of the case features - this is the recursive case. And any use of (ii) must have a corresponding use of the non-recursive $=o\ =s\ v\ /v/$. An induction shows that either way, we will have equal numbers of c, v, s, o , in the appropriate order.

It is well known that neither $strings(MG_4)$ nor $strings(MG_5)$ are context free languages, but these languages are included in the “mildly context sensitive” class defined by [28], a class definable by tree adjoining grammars, by a certain kind of head wrapping grammars, and by a certain restricted form of combinatory categorial grammar. Let’s call these languages TAG languages. It has been argued that we need to be able to define copying (ww) constructions to get Dutch dependencies [4] and to get the Swiss-German string set [25].

Various linguists have argued that human languages cannot be properly handled by formalisms that yield only the TAG languages. [24] argues that we need more power to get the right dependencies in German scrambling. [31] and [23] show that we need more power to get the string sets of number names in English and Chinese. And [20] shows that we need more than TAG power to get the case system of Old Georgian, as that language is described in [2]. Given these proposals, the following, final example is of interest.

2.6 $c^n v^n x^n s^n o^n$

Languages with five counting dependencies are not included in the TAG languages. Notice that the previous example uses strong selection (in particular, head movement with right adjunction) to build up the $c^n v^n$, and it uses XP movement of (the “remnants” of) $-case\ CP$ to build up the $s^n o^n$, interleaving these two processes properly. While there can be at most one head extraction from any constituent, there can be as many XP movements as there are kinds of licensees. (That is, there can be anywhere from 0 to $|licensees|$ extractions from any constituent.) We use this idea to define a language with more than 4 counting dependencies. This example is based on one formulated by Anoop Sarkar (p.c.).

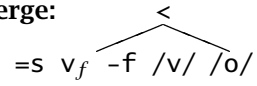
Let Lex_6 be the following set of 13 expressions:

$=T\ c\ /c/$	c	
$=V_f\ +D\ =x\ t\ -d$	$=V_f\ =x\ t\ -d$	$=V\ =x\ t\ =V\ +D\ =x\ t$
$=o\ =s\ v\ /v/$	$C= =o\ +F\ =s\ v\ /v/$	
$=o\ =s\ v_f\ -f\ /v/$	$C= =o\ +F\ =s\ v_f\ -f\ /v/$	
$s\ /s/$		
$o\ /o/$		
$x\ /x/$		

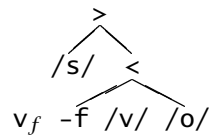
The following derivation of $/c\ c\ v\ v\ x\ x\ s\ s\ o\ o/$ in 13 steps illustrates the basic recursion in the grammar. The x ’s are collected in the highest spec-

ifiers of τ , the s's and o's are collected in the highest specifiers of v , and the c's and v's all raise to the highest c:

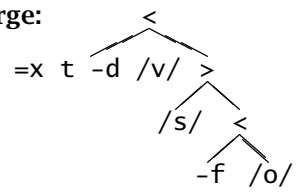
step 1 merge:



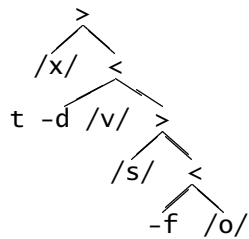
step 2 merge:



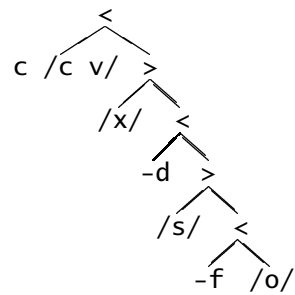
step 3 merge:



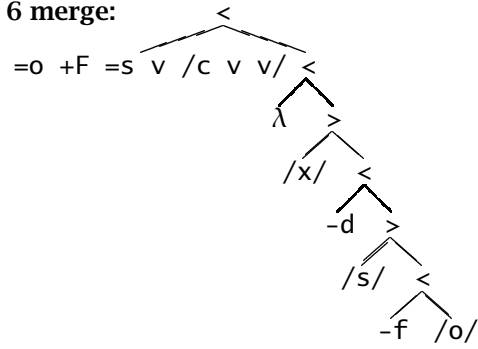
step 4 merge:



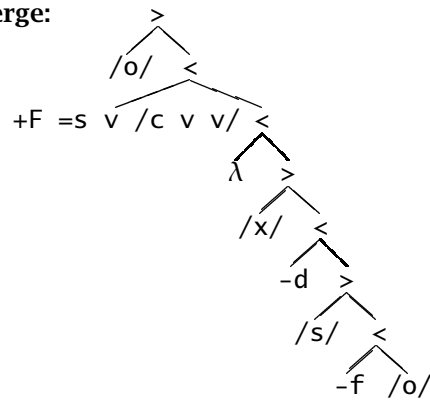
step 5 merge:



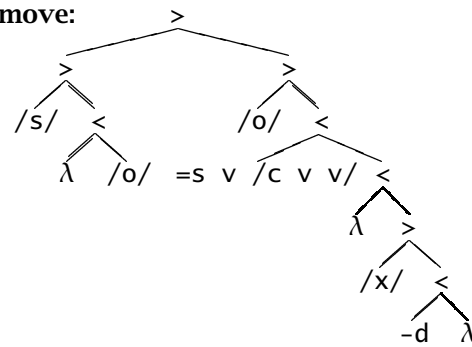
step 6 merge:



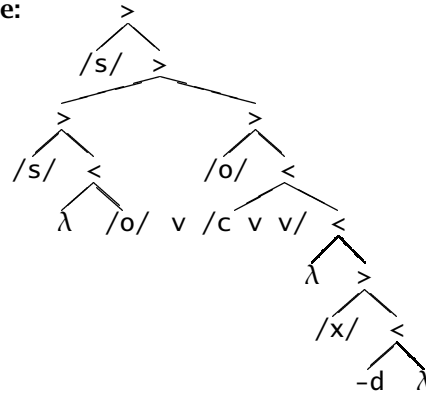
step 7 merge:



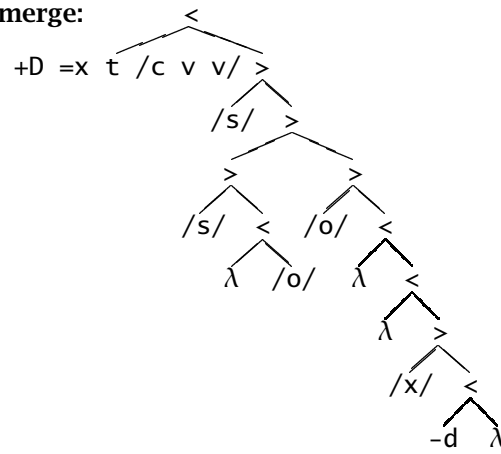
step 8 move:



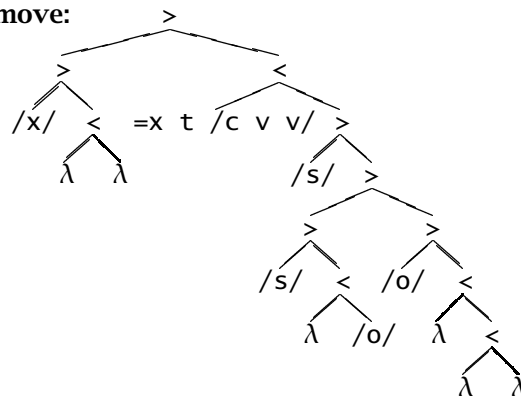
step 9 merge:



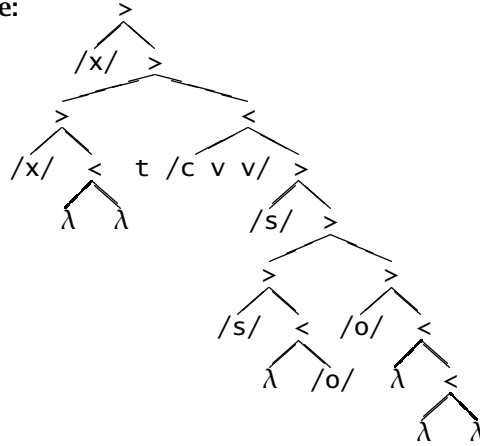
step 10 merge:



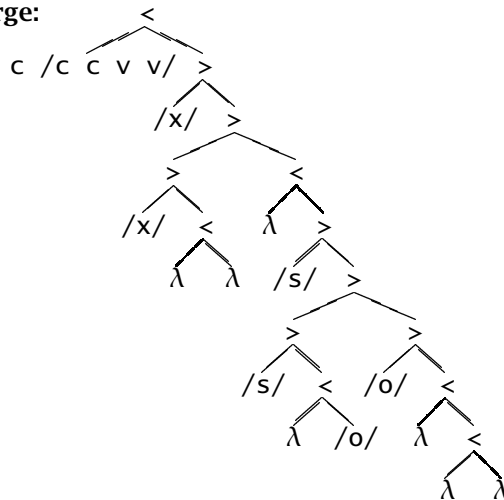
step 11 move:



step 12 merge:



step 13 merge:



Proposition 9. $strings(MG_6) = \{c^n v^n x^n s^n o^n \mid n \geq 0\}$.

Languages like this one, with five counting dependencies, are not TAG languages. It remains an open problem to specify how the MG-definable string sets compare to previously studied supersets of the TAG language class.

3 Beyond MGs

Following Chomsky and the earlier transformational tradition, the presentation of minimalist theory here is derivational. Brody [5] presents a representational minimalism that may appear quite different, but Cornell's [10] formalization of "representational minimalism" shows that the two are remarkably close. However, regardless of whether the presentation is derivational or representational, a simple formalism like the one explored in this paper requires some elaboration to handle human languages. For example, consider successive cyclic A-movement, as in

- i. John_i is t_i happy
- ii. John_i seems t_i to be t_i happy
- iii. John_i appears t_i to seem t_i to be t_i happy
- iv. ...

One way to allow this assumes that while triggering features like $=d$, $=n$, $+case$, $+CASE$ are deleted, the properties of the expressions d , n , $-case$ are optionally not deleted.

Chomsky explores some much more elaborate grammars in [7], considering especially the following ideas:

- Maybe the syntax of human languages defines unordered trees, with $<$ imposed on the leaves by non-syntactic conditions.
- Maybe projection need not be explicitly represented in all constructions, because it is fixed by language-universal principles.⁶
- There may be additional "economy principles," acting as a kind of filter on derivations, and potentially defining a different class of languages.
- Maybe the sharp distinction between head movement and phrasal movement can be eliminated in favor of a general account of what "pied pipes" in any movement, and some general minimality condition on movements.

Some of these ideas are explored in a formal setting in [27,30].

The MG grammar formalism is not symmetric with respect to linear order: all movements are leftward, the first selected constituent always attaches to the right of the selector, and all other selected constituents attach to the left. It is interesting to compare the asymmetries in this system with those considered by Kayne in [12]. Given any tree $\tau = (N, <^*, <^*, <^*)$, define the relations sister S , c-command CC , and asymmetric c-command ACC , all of them binary relations on N , as follows:⁷

⁶ [7, §4.4.2] suggests that while *merge* is just defined to project one rather than the other of its arguments, in the case of *move* operations, we can deduce from other general assumptions about "checking" and "economy" that in the result, the moved subtree is projected over by its sister.

⁷ The formulation of Kayne's view offered here is simplified in two significant respects. First, following [18] and [6], Kayne complicates these definitions for "adjunction structures." These have no exact analog in our system, but our specifiers

aSb iff $a \neq b$ and for some $c \in N$, $c \triangleleft a$ and $c \triangleleft b$.
 $aCCb$ iff for some c , aSc and $c \triangleleft^* b$.
 $aACCb$ iff $aCCb$ and $\neg bACCa$.

Now define a binary relation R on the leaves of the tree $L = \{x \mid \neg(\exists y) x \triangleleft y\}$:

aRb iff for some x, y , $x \triangleleft^* a$, $y \triangleleft^* b$, and $xACCy$.

Kayne's hypothesis (the "linear correspondence axiom") is this:

(LCA) In the structures of natural languages, R is a strict linear order on L .

That is, $\forall x, y, z \in L$,

- (i) if xRy then $x \neq y$, (irreflexivity)
- (ii) if xRy and yRz then xRz , (transitivity)
- (iii) if xRy then $\neg yRx$, and (asymmetry)
- (iv) if $x \neq y$ then either xRy or yRx . (linearity)

Note that axiom (iii) is redundant, since it is entailed by (i) and (ii), and that it is different from what is usually called antisymmetry ($(xRy \wedge yRx) \rightarrow x = y$). In terms of what languages can be defined, the LCA clearly imposes no restriction at all until it is combined with other constraints, but we can immediately consider whether the LCA is respected in the structures defined by MG grammars.

Clearly, the LCA is not respected in the trees defined by our grammars. For example, in the structure derived in §2.1 (as the result of step 8) the deepest 2 leaves are labeled /every/ and /language/, and these leaves are not linearly ordered by R . There are no nodes x, y such that $xACCy$, x dominates /every/, and y dominates /language/. Discussing this kind of case, Kayne says [12, pp9-10] that these two nodes "are in too symmetric a relation with one another...replacing one of the two symmetric nodes by a more complex substructure breaks the symmetry and renders the phrase marker admissible." But notice that in our system this structure is not "symmetric" in any interesting sense: these two nodes enter the derivation as the selecting node and the selected node respectively; one is a complement and one is not; and so on. Given these features, the MG formalism (and any near variant) is going to require syntactic analyses similar to some of Kayne's, even though it does not use the extra X-bar-like structure required by the LCA to mark the asymmetries. Considering the goal of properly capturing the asymmetries of human linguistic structures, it is not clear that the LCA

might be regarded as left-adjoined. Making this adjustment in the claim would not affect the very general remarks made here. Second, the LCA below could be taken as applying only to phonetically overt leaves of a tree, rather than to all the leaves, as suggested in [7, p337], for example. Again, for present purposes this adjustment is not relevant.

has any advantage over this or other similar mechanisms that yield an appropriate structural asymmetry in all definable languages.⁸

One other prominent idea in recent transformational grammar is that movement is actually copying and deletion, where deletion is not just a reflex of movement, but a process that is potentially sensitive to other aspects of structure. One empirical motivation for this idea comes from the existence of apparent “duplication” of phonetically overt substructures, where the duplicated elements are syntactically related but not part of the same word. This happens in a wide range of diverse languages and constructions: for example, in “predicate clefts” [14, §6], in “partial wh-movement” [19], and in “resumptive pronouns” [26]. The attempt to account for these and other phenomena has led to the consideration of grammatical constraints that refer specifically to the presence of phonetic material [26,21,16]. This contrasts with a relatively long-standing tradition according to which phonetic material is not relevant to the principles of grammar. For example, Bouchard [3] argues that the stipulation of properties depending on whether a constituent is empty or not can undermine the empirical force of the grammar. The simple MG formalism presented here follows this older tradition, making phonetic and interpretable material completely irrelevant to the syntactic derivation. The question of how it should be extended to handle duplications of phonetic material is considered in [27].

Appendix: Definition of the structure building operations

An *expression* is a tree τ , as defined in §1.2. An expression is *complex* (or a *non-head*) iff it has more than one node; otherwise, it is *simple* (or a *head*).

In the following definition it will be convenient to say that a tree τ *has feature* (or *category*) f iff the first element of the sequence that labels the head of τ is f . And we will say that a tree is *maximal* iff its root is the maximal projection of some head.

We use the standard labelled bracket notation $[_< \tau_0, \tau_1]$ to indicate an expression with immediate subtrees τ_0 and τ_1 , in that linear order, where τ_0 projects over τ_1 .

$$Dom(merge) = \{[\tau_0, \tau_1] \mid \begin{array}{l} \text{either } \tau_0 \text{ is a head that has feature } =x, =X, \text{ or } X=, \\ \text{and } \tau_1 \text{ has category } x, \\ \text{or } \tau_0 \text{ is a complex expression that has feature } =x, \\ \text{and } \tau_1 \text{ has category } x \end{array} \}$$

⁸ See [7, §4.8] for further discussion of this point.

$$\text{merge}(\tau_0, \tau_1) = \left\{ \begin{array}{l}
[<\tau'_0, \tau'_1] \text{ if } \tau_0 \text{ is a head that has feature } =x, \\
\quad \tau'_0 \text{ is like } \tau_0 \text{ except that } =x \text{ is deleted, and} \\
\quad \tau'_1 \text{ is like } \tau_1 \text{ except that } x \text{ is deleted} \\
[<\tau'_0, \tau'_1] \text{ if } \tau_0 \text{ is a head that has feature } =X, \\
\quad \tau'_0 \text{ is like } \tau_0 \text{ except that } =x \text{ is deleted} \\
\quad \text{and its phonetic features are the result of} \\
\quad \text{concatenating those of } \tau_0 \text{ and } \tau_1 \text{ in that order,} \\
\quad \tau'_1 \text{ is like } \tau_1 \text{ except } x \text{ and all phonetic features are deleted} \\
[<\tau'_0, \tau'_1] \text{ if } \tau_0 \text{ is a head that has feature } X=, \\
\quad \tau'_0 \text{ is like } \tau_0 \text{ except that } =x \text{ is deleted} \\
\quad \text{and its phonetic features are the result of} \\
\quad \text{concatenating those of } \tau_1 \text{ and } \tau_0 \text{ in that order,} \\
\quad \tau'_1 \text{ is like } \tau_1 \text{ except } x \text{ and all phonetic features are deleted} \\
[>\tau'_1, \tau'_0] \text{ if } \tau_0 \text{ is a complex that has feature } =x \\
\quad \tau'_0 \text{ is like } \tau_0 \text{ except that } =x \text{ is deleted, and} \\
\quad \tau'_1 \text{ is like } \tau_1 \text{ except that } x \text{ is deleted}
\end{array} \right.$$

$$\text{Dom}(\text{move}) = \{ \tau \mid \begin{array}{l} \tau \text{ has a feature } +x \text{ or } +X, \text{ and} \\ \tau \text{ has exactly one proper subtree with the feature } -x \end{array} \}$$

$$\text{move}(\tau) = [>\tau'_0, \tau'], \text{ where}$$

either τ has feature $+X$,
 τ_0 is a proper subtree of τ with feature $-x$,
 τ_0 is maximal,
 τ'_0 is like τ_0 except that $-x$ is deleted, and
 τ' is like τ except that $+X$ is deleted and
subtree τ_0 is replaced by a leaf with no features
or τ has feature $+x$,
 τ_0 is a proper subtree of τ with feature $-x$,
 τ_0 is maximal,
 τ'_0 is like τ_0 except that $-x$ and all phonetic features are deleted,
and τ' is like τ except that $+x$ is deleted and
all non-phonetic features in τ_0 are deleted

Acknowledgments: Thanks to Edward Keenan, Thomas Cornell, Sean Fulop, Francois Lamarche, Jeff MacSwan and Anoop Sarkar for stimulating discussions and helpful suggestions on previous versions.

References

1. Mark Baker. *The Polysynthesis Parameter*. Oxford University Press, New York, 1996.
2. Winfried Boeder. Suffixaufnahme in Kartvelian. In Frans Plank, editor, *Double Case: Agreement by Suffixaufnahme*. Oxford University Press, New York, 1995.

3. Denis Bouchard. *On the Content of Empty Categories*. Foris, Dordrecht, 1984.
4. Joan Bresnan, Ronald M. Kaplan, Stanley Peters, and Annie Zaenen. Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13(4):613–635, 1982.
5. Michael Brody. *Lexico-Logical Form: A Radically Minimalist Theory*. MIT Press, Cambridge, Massachusetts, 1995.
6. Noam Chomsky. *Barriers*. MIT Press, Cambridge, Massachusetts, 1986.
7. Noam Chomsky. *The Minimalist Program*. MIT Press, Cambridge, Massachusetts, 1995.
8. Sandra Chung and James McCloskey. Government, barriers and small clauses in modern Irish. *Linguistic Inquiry*, 18:173–238, 1987.
9. Thomas L. Cornell. Deriving the *ww* language with μ P style. University of Tübingen, 1996.
10. Thomas L. Cornell. Representational minimalism. University of Tübingen, 1996.
11. Jane H. Hill and Kenneth C. Hill. *Speaking Mexicano*. The University of Arizona Press, Tucson, Arizona, 1986.
12. Richard Kayne. *The Antisymmetry of Syntax*. MIT Press, Cambridge, Massachusetts, 1994.
13. Edward L. Keenan and Edward P. Stabler. Abstract syntax. In Anna-Maria Di Sciullo, editor, *Configurations: Essays on Structure and Interpretation*, pages 329–344, Somerville, Massachusetts, 1996. Cascadilla Press. Conference version available at <http://128.97.8.34/>.
14. Hilda Koopman. *The Syntax of Verbs: From Verb Movement Rules in the Kru Languages to Universal Grammar*. Foris, Dordrecht, 1983.
15. Hilda Koopman. On verbs that fail to undergo V-second. *Linguistic Inquiry*, 26:137–163, 1995.
16. Hilda Koopman. The spec-head configuration. *Syntax at Sunset: UCLA Working Papers in Syntax and Semantics*, edited by Edward Garrett and Felicia Lee, 1996.
17. Hilda Koopman and Dominique Sportiche. The position of subjects. *Lingua*, 85:211–258, 1991.
18. Robert May. *Logical Form: Its Structure and Derivation*. MIT Press, Cambridge, Massachusetts, 1985.
19. Dana McDaniel. Partial and multiple wh-movement. *Natural Language and Linguistic Theory*, 7:565–604, 1989.
20. Jens Michaelis and Marcus Kracht. Semilinearity as a syntactic invariant. In *Logical Aspects of Computational Linguistics (LACL)*, pages 37–40, 1996.
21. David Pesetsky. Principles of sentence pronunciation. MIT Lecture notes. Available at <http://rucss.rutgers.edu/roa.html>, 1994.
22. Jean-Yves Pollock. Verb movement, universal grammar, and the structure of IP. *Linguistic Inquiry*, 20:365–424, 1989.
23. Daniel Radzinski. Chinese number names, tree adjoining languages and mild context sensitivity. *Computational Linguistics*, 17:277–300, 1991.
24. Owen Rambow, K. Vijay-Shanker, and David Weir. D-tree grammars. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, 1995. Available at <http://xxx.lanl.gov/cmp-lg/ACL-95-proceedings.html>.
25. Stuart M. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–344, 1985.
26. Ur Shlonsky. Resumptive pronouns as a last resort. *Linguistic Inquiry*, 23:443–468, 1992.
27. Edward P. Stabler. *Acquiring and parsing languages with movement*. Basil Blackwell, Oxford, 1996. Forthcoming. Draft available at <http://128.97.8.34/>.

28. K. Vijay-Shanker and David Weir. The equivalence of four extensions of context free grammar formalisms. *Mathematical Systems Theory*, 27:511-545, 1994.
29. Sten Vikner. *Verb Movement and Expletive Subjects in the Germanic Languages*. Oxford University Press, New York, 1995.
30. Charles Yang. Minimal computation. Massachusetts Institute of Technology, 1996.
31. Arnold Zwicky. Some languages that are not context free. Technical Report Quarterly Progress Report 70, MIT Research Laboratory of Electronics, Cambridge, Massachusetts, 1963.