Bericht Nr. 79, Oktober 1996

Arbeitspapiere des
Sonderforschungsbereichs 340

ISSN 0947-6954/96

# Sprachtheoretische Grundlagen für die Computerlinguistik

**Tom Cornell:**

## A Minimalist Grammar for Deriving the Copy Language

Universität Stuttgart Universität Tübingen
IBM Deutschland Informationssysteme GmbH

**Arbeitspapiere des SFB 340**
Bericht Nr. 79, Oktober 1996

# A MINIMALIST GRAMMAR FOR DERIVING THE COPY LANGUAGE

TOM CORNELL

In these notes we present a minimalist derivational system modeled largely on Stabler's formalization (Stabler 1996a,b) of (ideas from) Chomsky's (1995) *Minimalist Program*. Our intention is to show that an extremely tightly constrained version of this system can nonetheless include grammars which generate the language

$$\mathcal{L}_{ww} = \{ww \mid w \in \{0, 1\}^*\}.$$

This language is known to be context-sensitive, and is the basis of the proofs in Huybregts (1984) and Shieber (1985) for the non-context-freeness of natural language. We focus here on an artificial language because it is possible to present a complete grammar which provably generates it. While the formalism is provably powerful enough to support grammars for (at least some) mildly context sensitive languages, it remains computationally attractive. In particular, its essentially linear use of lexical resources means that it should be decidable. This in turn may help to suggest ways to extend decidable grammar logics such as those presented in Rogers (1994) and Kracht (1995) beyond their current strictly context-free power without losing decidability.

Our concern here is mainly to isolate what we hope is the minimal apparatus necessary to derive strings in $\mathcal{L}_{ww}$ in a Chomsky-style Merge & Move calculus. It turns out we will in fact need very little. As might be expected from the natural languages which exemplify $\mathcal{L}_{ww}$-like behavior, we will only need head-movement. So, *a fortiori*, we will not need successive cyclic movement. This means we can stay within the "linear" fragment: every movement operation consumes some resources, and no operation increases resources. We will also need only one type of feature, namely categorial features (surely the minimal minimalist feature!)

ISSN 0947-6954/96
©1996 Der Autor

and their checking counterpart, selectional features. This means we will essentially be working in a categorial grammar with movement.

## 1. The Formalism

The formalism developed for Chomsky's Minimalist Program in Stabler (1996a) is fundamentally *derivational*: the well-formed structures are the derivable structures. Following Chomsky (1995) we take derivations to start with a *numeration*, essentially a multiset of lexical items which constitutes a (minimally) structured database of resources from which we are to derive appropriate phrase markers. (Compare this approach to the logical approach to categorial grammars as in Moortgat (1996) and Kurtonina and Moortgat (1996), and refs.)

Derivations represent the application of two possible operations, Merge and Move, to the numeration. Move is constrained by the *Last Resort* principle, so that if it need not take place, it cannot take place. All movements are ultimately driven by the need to satisfy *feature checking* requirements. That is, features come in *clashing pairs*, and the object of the derivation is to place all features into a restricted set of configurations with clashing counterparts. In our fragment, we will be concerned to match category features with selection features, but we might equally well consider case-marking and case-assigning features, or nominal vs. predicative agreement features. In the *linear fragment* of this "derivational minimalism" we insist that clashing pairs of features leave us with a one-to-one and (almost) onto function from the categorial features to the selection features that cancel them. I say "almost" because one category (the category of the matrix clause) is not selected by anything else. So our derivations will be of structures containing only a single uncancelled category feature, which will be taken to be the category which the grammar assigns to the (well-formed) sound-meaning pair the derivation generates.

So, we think of a given feature abstractly as appearing either "positively charged" or "negatively charged". In general, we distinguish *attractor* features from their clashing counterparts: movement will always be movement of an attracted feature (the *attractand*) into the *checking domain* of its attractor counterpart. For $\mathcal{L}_{ww}$, selectional features will be the only attractor features we need.

Attractor features in general are distinguished in Chomsky's work as being either *weak* or *strong*. Strength of a feature plays a role in the theory of movement. In Chomsky's theory of movement, a derivation is divided into two stages—the overt componenet and the covert component—which

$$\text{Category features } \mathcal{C}at = \{\mathsf{s}, \mathsf{f}, \mathsf{a}, \mathsf{b}\}$$
$$\text{Selection features } \mathcal{S}el = \{\overline{\phi} \mid \phi \in \mathcal{C}at\}$$
$$\text{Strong: } \overline{\mathsf{F}}, \text{ Weak: } \overline{\mathsf{f}}$$
$$\text{Alphabet } \Sigma = \{\mathsf{0}, \mathsf{1}\}$$

TABLE 1. Primitive symbolic vocabulary of the formalism.

differ essentially in that movement operations in the overt component are constrained to "pied pipe" phonological (here orthographical) features. In the covert component, only the syntactic features themselves move. Strong features trigger movement in the overt component, weak features trigger movement in the covert component.

Stabler adopts a variant of this theory which dispenses with different components, distinguishing instead different kinds of movements. For our purposes we can think of this as just distinguishing what we might as well call "strong movement" from "weak movement", where strong movement is movement which brings phonological features along with it while weak movement leaves them behind.[1] We adopt here Stabler's notational convention, presenting strong selection features in upper-case and the weak variant in lower-case. Note however that in $\mathcal{L}_{ww}$ all movements will be strong so the distinction is not used much here.

We will use the basic vocabulary of symbols and notation illustrated in Table 1. Given a category feature $\mathsf{x}$, the corresponding selectional feature can be written either $\overline{\mathsf{x}}$ or $\overline{\mathsf{X}}$. (We will use $\overline{\chi}$ to range over strong or weak variants of a feature $\overline{\mathsf{x}}$.) Intuitively, the category $\mathsf{s}$ is the category of complete expressions, i.e., strings of the language being defined; $\mathsf{f}$ is a recursive functor category, roughly analogous to the V+Infl complex of natural languages, and $\mathsf{a}$ and $\mathsf{b}$ are "constant" categories, which themselves take no arguments, and which serve as the primary (nonrecursive) arguments of the $\mathsf{f}$ categories.

A lexicon is a set of trees; in $\mathcal{L}_{ww}$ all these trees will be simple atoms. These are illustrated in Table 2. Lexical items are written in the following notation. The formal features of a lexical item are written above the line, its associated interpretation-features (here just its orthographical features) are written below. We take the formal features to be ordered;

[1] Stabler's system is in fact more complex than this, since there we must keep track of semantic features as well as phonological features. It appears that there are cases in which some, all or none of these interpretation-features are pied piped.

| | |
|---|---|
| $\dfrac{\mathsf{s},\overline{\mathsf{F}}}{\epsilon}$ ($\mathsf{s}$) | $\dfrac{\mathsf{f},\overline{\mathsf{F}},\overline{\mathsf{a}}}{\mathsf{0}}$ ($\mathsf{f_a}$) |
| $\dfrac{\mathsf{a}}{\mathsf{0}}$ ($\mathsf{a}$) | $\dfrac{\mathsf{f},\overline{\mathsf{F}},\overline{\mathsf{b}}}{\mathsf{1}}$ ($\mathsf{f_b}$) |
| $\dfrac{\mathsf{b}}{\mathsf{1}}$ ($\mathsf{b}$) | $\dfrac{\mathsf{f}}{\epsilon}$ ($\mathsf{f_0}$) |

TABLE 2. The lexicon for $\mathcal{L}_{ww}$.

this does not matter for any but the selectional features, and there we are just taking over the old assumption that we can distinguish internal and external arguments. However, this may be an innovation of Stabler's, not present in Chomsky (1995). The symbols appearing in parentheses will be used as convenient names for the lexical items. (We cannot just use the orthographic forms, as the string "0", for example, has two very distinct categories assigned to it.)

Note that for Chomsky all language variation is implemented in the lexicon. The definitions of Merge and Move are meant to be universal, so what trees we can derive depends only on what lexical material we have to start with. Therefore the lexicon in Table 2 is essentially what there is to our grammmar of $\mathcal{L}_{ww}$. The main points to observe are that

- nothing selects $\mathsf{s}$;
- $\mathsf{f}$ is always a complement, and
- $\mathsf{f}$ is only ever strongly selected—so $\mathsf{f}$'s will always have to move, bringing their orthography with them;
- $\mathsf{a}$ and $\mathsf{b}$ are only ever weakly selected;
- the recursive $\mathsf{f}$ items always select for an $\mathsf{a}$ or $\mathsf{b}$ with the same orthographical features—this will implement a simple (and admittedly artificial) sort of "spec-head agreement".

Strings of $\mathcal{L}_{ww}$ will be just those orthographic structures assigned the category $\mathsf{s}$ by the derivation system. The lexical items $\mathsf{a}$ and $\mathsf{b}$ are "constants", that is, pure argument categories with no selectional features of their own. The lexical items $\mathsf{f_a}$ and $\mathsf{f_b}$ are the main functor/predicate elements. They are so named because $\mathsf{f_a}$ requires an $\mathsf{a}$ specifier and $\mathsf{f_b}$ requires a $\mathsf{b}$ specifier. Note also that items in the $\mathsf{f}$ category come in both recursive and nonrecursive varieties. The nonrecursive constant $\mathsf{f_0}$

provides the base of the recursion.[2] So the well-formed syntactic structures, it will be shown, will all be of one general type: they will have an $\mathsf{s}$-projection ("$\mathsf{s}$P") at the root, some number (maybe zero) of $\mathsf{f_a}$P and $\mathsf{f_b}$P in a strictly right-branching structure (with their associated $\mathsf{a}$P and $\mathsf{b}$P specifiers), with a single $\mathsf{f_0}$P as the bottom-most phrase. Overt head movement will extract the $\mathsf{f_a}$ and $\mathsf{f_b}$ heads, moving them up under the root $\mathsf{s}$ head.

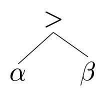
1.1. **Trees.** Syntactic phrase markers are just whatever structures our operations Merge and Move output, when we apply them (repeatedly) to the lexicon. However, despite the fact that there is no definition of what a phrase marker ought to look like outside of what is implicit in the definitions of Merge and Move, it is appropriate to consider explicitly what sort of structures we may look forward to.

Phrase markers are just trees in this formalism. Trees come in essentially two varieties: $X^0$ and XP. Note however that these categories overlap somewhat. Every tree is *headed* by a unique $X^0$ tree (the tree rooted in the node/segment called by Chomsky $X^{0\max}$). The features of the head project up the tree according to an extremely simple rule of feature percolation:

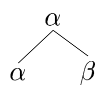
*all* (and only) features of the head project.

(As far as I know this does not exclude orthographic features, though what use might be made of them outside $X^0$ I cannot tell.)

Here we follow Stabler in using a notational convention which will serve to make these trees more readable. Interior nodes may bear labels from $\{<, >\}$. These are meant (following Stabler's practice) to indicate down which branch the head of the tree is to be found. So, for example, the tree

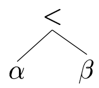


is headed by $\alpha$, that is, it stands for the tree

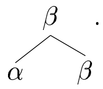


[2]There is no straightforward natural language analogue for $\mathsf{f_0}$. We could have made the grammar of $\mathcal{L}_{ww}$ more natural, but at the cost of making proofs more complicated.

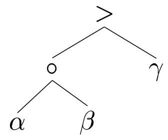
The tree



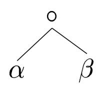
on the other hand is headed by $\beta$ and stands for the tree



Since we will be making extensive use of the head movement operation, we will be making extensive use of the adjunction structures which that operation creates. Normally, one handles adjunction by distinguishing between *segments* of a category (which correspond to nodes in the trees as they are customarily drawn) and the category itself, which is an unbroken chain of segments. The head of a tree is then that category which does not properly include any other category. However, in a system in which adjunction structures are created only by head movement and only serve to build up $X^0$'s, it is no longer clear whether the distinction we want is between categories and segments or between trees which are or are not $X^0$(word-internal/-external domains). That is, the key piece of information we have to recover from a tree is, for whatever reason, the identity of the node ($X^{0\max}$) where the $X^0$ proper leaves off and the rest of the structure begins. Whether the nodes which appear "south of the border" are categories or segments of categories is immaterial for our purposes. We will therefore adopt the following convention: interior nodes which belong to $X^0$ will be labeled with '$\circ$' rather than '$>$' or '$<$'. Since head adjunction structures are uniformly left-headed here, no information is lost thereby. (We can think of '$\circ$' as a segment-level or head-internal analog of '$>$'.) So the structure



is meant to represent a tree whose head is the complex $X^0$



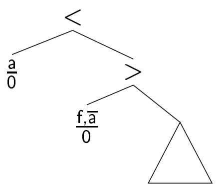
to which $\gamma$ has been attached as a complement. Note that the issue of $X^0$-internal headedness is far from clear-cut. Since heads adjoined into another head have their features visible to categories outside the complex $X^0$ so formed, it does not seem that in this case the features of the head are accurately described as "projecting over" the features of its adjuncts. However, we will assume that head adjunction has fixed consequences for the order in which the combined orthographical strings appear. For our own purposes we will assume that adjuncts all appear to the right of the material to which they are adjoined. This is necessary to derive the copy language with head movement, and reflects the apparent facts of Dutch and Swiss German, for example, though not of all head-adjunction phenomena. Note however that we could reverse the order; in that case, we would derive not the copy language but the palindrome language. While the latter language is known to be context free, that is, it is derivable from a recognizable set of trees, in our case we would still be assigning to its strings trees from a nonrecognizable set. The power of the system is not thereby compromised, but it is harder to see.

## 2. Feature Checking

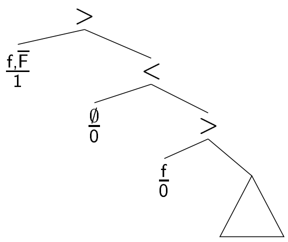
The essential intuition developed in Chomsky (1995) is that all movement is driven by the need to satisfy the demands of feature checking. In our particular examples, we will consider that heads marked with selection features will need to find, in their *checking domain*, a head with a corresponding categorial feature. So for example if $\alpha$ is a head with selection feature $\overline{\mathsf{b}}$, then that feature must be checked against some head of category $\mathsf{b}$ in $\alpha$'s domain. The acceptable structures will then be those in which all features needing to be checked (the selection features) have been cancelled against appropriately situated opposite-polarity versions of the same feature. Again, we will restrict ourselves here to the linear fragment, in which in addition all but one category feature must cancel a unique selection feature, leaving one remaining uncancelled feature, the category of the matrix.

The domain of feature checking is essentially two-fold in our system: specifiers of a phrase and adjuncts of a head are in the checking domain of the local head. For Chomsky, the checking domain of an unmoved head is essentially the set of categories attached to its projection, excluding the complement. Note that head movement extends the domain of the moved head. For our purposes this means that any unchecked features of a moved head can be checked in the domain of its landing site. Put another way, a given $X^0$ can be considered to have as features the union of the (unchecked) features of its constituents. (Though in fact none of our examples will afford an instance of this feature pied-piping.)

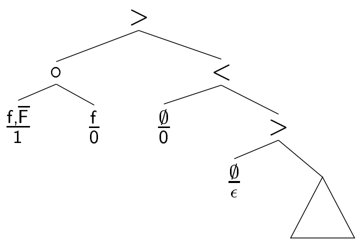
For example, in the tree




the pair of clashing features $\langle \mathsf{a}, \overline{\mathsf{a}} \rangle$ stand in a checking configuration (in particular $\frac{\mathsf{a}}{0}$ is attached to a projection of $\frac{\mathsf{f},\overline{\mathsf{a}}}{0}$ and is not its complement). So they can be cancelled against each other. On the other hand, in the tree




the pair $\langle \mathsf{f}, \overline{\mathsf{F}} \rangle$, where the first member is the category feature of the lower head, do not stand in a checking relation. In such a configuration we could raise the head of the complement to adjoin to the head of the overall tree, yielding




In this tree, the downstairs head $\frac{\mathsf{f}}{0}$ is now in the checking domain of the upstairs head $\frac{\mathsf{f},\overline{\mathsf{F}}}{1}$ so we can cancel them.

## 3. Operations

There are two basic operations in Chomsky's calculus: Merge and Move. Here we will distinguish two sorts of Move: strong Move and weak Move. Note that, in standard linguistic terms, all the instances of

Move we will consider here could be construed as instances of Head ($X^0$) Movement.

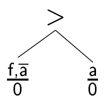
3.1. **Merge.** In both Chomsky's and Stabler's systems Merge is essentially free. It may result in checking of features, but it need not. In Stabler's resource-oriented terms, Merge does not necessarily consume any resources.[3] However, it is still sensitive to syntactic features in the following sense. Merge is a two-place operation which takes two trees as arguments, one of which *selects* the other.

**Definition 1.** *A tree $t_1$ selects another tree $t_2$ if the head of $t_1$ has as its first selectional feature $\overline{\mathsf{x}}$ or $\overline{\mathsf{X}}$ and the head of $t_2$ contains a corresponding categorial feature $\mathsf{x}$. We refer to $\mathsf{x}$ as the active selection feature. We refer to $t_1$ as the selecting tree and $t_2$ as the selected tree.*

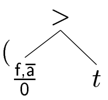
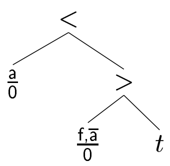
**Definition 2.** *A tree $t$ is* selectionally inert *if it contains no uncancelled selection features.*

Merge is *directional* in a sense familiar from GB theory: an $X^0$ takes its complement to the right and its specifier to the left. In our terms, if the root of the selecting tree is $<$ or $>$ then it becomes the right-hand subtree after merger; otherwise it becomes the left-hand subtree. The resulting tree is labeled $<$ in the former case, $>$ in the latter case; that is, the selecting tree becomes the head of the resulting construction.

So, for example:

(1) Merge($\frac{\mathsf{f},\bar{\mathsf{a}}}{\mathsf{0}}, \frac{\mathsf{a}}{\mathsf{0}}$) $=$ 

On the other hand,

(2) Merge(  , $\frac{\mathsf{a}}{\mathsf{0}}$) $=$ 

[3]Note that this is obvious only in Stabler's earliest grammars. Later Merge and Move (for heads) are combined into a single operation, which *does* consume features, presumably as a consequence of the movement component of the operation.

**Definition 3.** *Given a pair of trees $t_1, t_2$, such that $t_1$ selects $t_2$, $t_2$ selectionally inert. Then*

$$\text{Merge} : (t_1, t_2) \mapsto \begin{cases} \overset{>}{\widehat{t_1 \qquad t_2}} & \textit{if } t_1 \textit{ is } \mathrm{X}^0, \\ \overset{<}{\widehat{t_2 \qquad t_1}} & \textit{if } t_1 \textit{ is not } \mathrm{X}^0. \end{cases} \tag{3}$$

Note that adjunction structures are never created by Merge in this system. (So, in particular, the action of Merge on two $\mathrm{X}^0$'s is determinate.)

Observe also the following. Suppose we have a head $\frac{\mathsf{f},\overline{\mathsf{F}},\bar{\mathsf{a}}}{\mathsf{0}}$, that is, a transitive head. Since Merge is constrained by the current value of the leftmost selection feature ($\overline{\mathsf{F}}$ in this case), it will not be possible to merge in a tree of category $\frac{\mathsf{a}}{\mathsf{0}}$ until the $\overline{\mathsf{F}}$ feature has been cancelled. This is a very strong condition which forces the derivation to develop along very narrowly constrained lines. I do not know if it is absolutely necessary, but it will certainly make proofs of grammar-language correspondence easier, so I will retain it here. (It comes originally from Stabler (1996a).)

3.2. **Move.** Move as defined here is just head movement. The essential idea is as usual: applying Move to a tree has the effect of detaching some subtree of the input, and then combining it again with the tree from which it has been detached, but now in another position. Following the general bent of the minimalist program, we treat movement as driven by the *attraction* of a categorial feature by a selection feature, where the attracted categorial feature *pied-pipes* its orthography if it is strongly attracted, not if weakly.

Normally, head movement is treated as a species of adjunction, and supposed thereby to be just like any other kind of movement. However, by the time Chomsky is finished with chapter 4 of *The Minimalist Program*, it would appear that this is the *only* species of adjunction left in the grammar, so in fact head movement is not just like any other sort of movement. So "head movement" and "adjunction" will be synonymous for the remainder of this paper. This leaves us with four kinds of movement: strong XP movement, weak XP movement, strong $\mathrm{X}^0$ movement and weak $\mathrm{X}^0$ movement. Of these only strong $\mathrm{X}^0$ movement is exemplified in the derivation of strings in $\mathcal{L}_{ww}$.

We stipulate here that all movement is upward. That is, given a node including a selection feature, it can only look into its sister to find something to attract. So we define a head $t_2$ as being *accessible* to another head $t_1$ just in case it is a subtree of the sister of $t_1$ (that is, $t_1$ *i-commands*

$t_2$). Obviously, for constructing reasonable facsimiles of natural language, more complex notions of accessibility may be entertained. We introduce the concept here so as to retain a hook accessible [sic] for later extensions and elaborations.

Normally Move is subjected to some sort of locality condition. Here it appears we can get by with the optimally simple condition: move only the closest tree whose head you attract. That is, if the attractor feature is $\overline{\chi}$, then the closest attractand is a tree marked $\mathsf{x}$ such that no other $\mathsf{x}$-tree c-commands it. Note that if there is no closest such tree (there is more than one equidistant tree, or there are no such trees at all) then Move is undefined. Again, like accessibility this notion of nearness is too trivial to stand up for long against the tidal pull of linguistic data, but it suffices for $\mathcal{L}_{ww}$, and it gives us a convenient hook accessible for later work.

**Definition 4.** *A tree $t_1$ attracts another tree $t_2$ if there is some selection feature $\overline{\chi} \in cat(t_1)$ such that $x \in cat(t_2)$. (Where $\chi$ ranges over $\mathsf{x}$ and $\mathsf{X}$.)*

Let $t_1/t_2$ denote the result of removing $t_2$ from $t_1$. A little more formally, suppose we have a collection of special labels (call them *variables*) $(\frac{\emptyset}{\epsilon})_x$ and $(\frac{\emptyset}{\mathsf{w}})_x$, with indices drawn from some index set $X$. Then, given some $x \in X$, $t_1/_x t_2$ is that tree such that replacing every occurrence of a variable indexed by $x$ with $t_2$ would yield $t_1$. We will refer to this division operation as *detachment*, distinguishing weak detachment (in which $t_2$ corresponds to a "variable" of the form $(\frac{\emptyset}{\mathsf{w}})_x$) from strong detachment (in which $t_2$ corresponds to a variable of the form $(\frac{\emptyset}{\epsilon})_x$). We limit detachment to heads, and insist that weak detachment remove all features from a given position and leave behind all the orthography associated with that position.

So, weak movement will be weak detachment of the nearest attracted head followed by adjunction to the head of the whole tree, and strong movement will be strong detachment of a head followed by adjunction to the head of the whole tree. Weak movement is triggered by a weak attractor feature in the tree's head, while strong movement is triggered by a strong attractor feature in the tree's head. Note that there is in practice no need to keep the "variable"-tag in the vacated "trace" position. In this system we need neither indices, chains nor traces. (These annotations could be required again if we wanted to build a logical calculus around the algebraic calculations we are employing here.)

**Definition 5.** *Given trees*

$$t_1 = \frac{\{\overline{\chi}\} \cup \mathsf{G}}{\mathsf{v}}$$

*and*

$$t_2 = \frac{\{\chi\} \cup \mathsf{H}}{\mathsf{w}}$$

*and a tree*

$$t = \begin{array}{ccc} & > & \\ / & & \backslash \\ t_1 & & t'[t_2] \end{array}$$

*and in particular* $t_2$ *is that subtree which is* closest *to* $t_1$ *among those heads marked* $\{\chi\}$*. Then*

$$\text{Move}(t) = \begin{array}{ccc} & > & \\ / & & \backslash \\ \begin{array}{ccc} & \circ & \\ / & & \backslash \\ \frac{\mathsf{G}}{\mathsf{v}} & & \frac{\mathsf{H}}{\mathsf{w}} \end{array} & & t'/t_2 \end{array}$$

In this definition, we present feature cancellation as actually being part of the Move operation. In fact, Move as we have defined it invariably feeds feature cancellation. In this way we have built the Last Resort Principle into the definition of the Move operation, with the effect that Move only applies if their is no other way for the derivation to proceed. Since the derivation is always driven by the need to cancel features, this means that Move will in principle only apply if it can lead to feature cancellation. But whether that feature cancellation is actually a built-in sub-operation of Move or a separate operation which Move feeds is not really determined by the ideas laid out in Chomsky (1995).

## 4. A Derivation of the String 1010

In this section we present and describe the derivation in our formalism of the string 1010. We present it as a derivation tree (a *T-marker*, to revive Chomsky's old terminology), though, spread out as it is over a number of pages, the overall structure of this meta-tree may be less than transparent. The T-marker is constructed deliberately to resemble a proof tree, but bear in mind we are only tracing algebraic calculations here, not (yet!) doing proofs in a logic.

**Definition 6.** *A* $\chi$*-tree is a tree containing* $\chi$ *as its only uncancelled feature.*

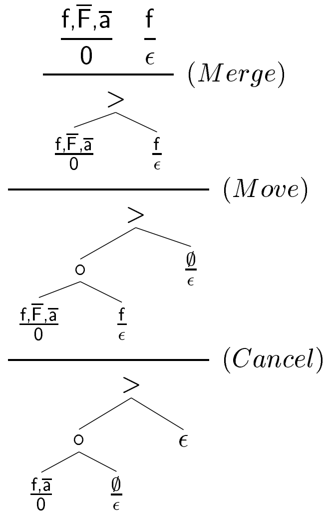



FIGURE 1. Complement selection and subsequent $X^0$ movement.

Our goal will be to derive an s-tree. In fact we will later define the language generated by our grammar to be just those strings which are the yields of s-trees.

The numeration we begin from is as follows.

$$\frac{\mathsf{s},\overline{\mathsf{F}}}{\epsilon}, \; \frac{\mathsf{b}}{1}, \; \frac{\mathsf{f},\overline{\mathsf{F}},\overline{\mathsf{b}}}{1}, \; \frac{\mathsf{a}}{0}, \; \frac{\mathsf{f},\overline{\mathsf{F}},\overline{\mathsf{a}}}{0}, \; \frac{\mathsf{f}}{\epsilon} \tag{4}$$

Using our abbreviated notation, this becomes the following.

$$\mathsf{s},\ \mathsf{b},\ \mathsf{f_b},\ \mathsf{a},\ \mathsf{f_a},\ \mathsf{f_0} \tag{5}$$

Our first move cannot be a Move, since the numeration contains only atoms, so it must be a Merge. Furthermore, to be a legal application of Merge, it must combine a functor category which selects an inert (non-functor) category. All three functor categories select for f, and the only selectionally inert element of that category is $\mathsf{f_0}$, so it must be the second argument to Merge. We can begin with any of the three functor categories as the first argument to Merge. Note that if we choose s, it will trigger the raising of $\mathsf{f_0}$, cancelling all uncancelled features but the s, ending the derivation with an s-tree which yields the empty string, which is an element of $\mathcal{L}_{ww}$ (though not derivable from the full numeration in (4)). However, we will avoid this trivial option and choose to apply Merge to $\mathsf{f_a}$ and $\mathsf{f_0}$.

Figure 1 shows this first step and the subsequent raising of $\mathsf{f}_0$ to adjoin to $\mathsf{f}_\mathsf{a}$. Note that here we have illustrated feature cancellation as a separate step; this is done for clarity of the presentation and not out of principle. Recall that we have in fact defined head movement as involving feature cancellation.

Note that when we first empty out the content of a given position (either by Move, Cancel or both), we will leave a syntactically empty symbol in the relevant position, but after that we will retain only the orthography associated with each position. Note that structure dominating only the empty string could be simply removed at this point, "garbage-collected away," so to speak. Items with no syntactic features will play no role in the remaining derivation.

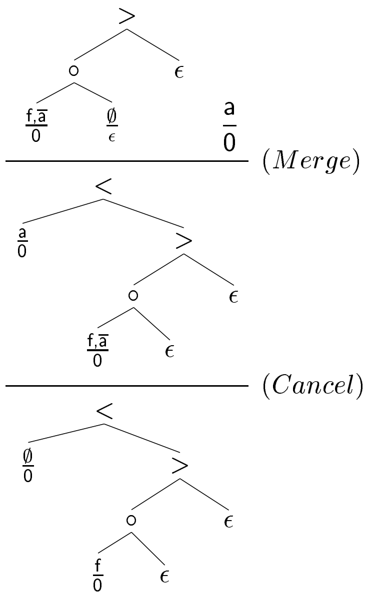



FIGURE 2. Specifier selection, spec/head configuration.

Once head movement has led to the cancellation of the $\overline{\mathsf{F}}$ feature, the next selection feature ($\overline{\mathsf{a}}$) is exposed, and we are allowed to Merge in the $\mathsf{a}$ from our numeration. Note that no other operation is possible: the $\overline{\mathsf{a}}$ feature cannot be satisfied by Move, in particular, and nothing else is allowed to Merge.

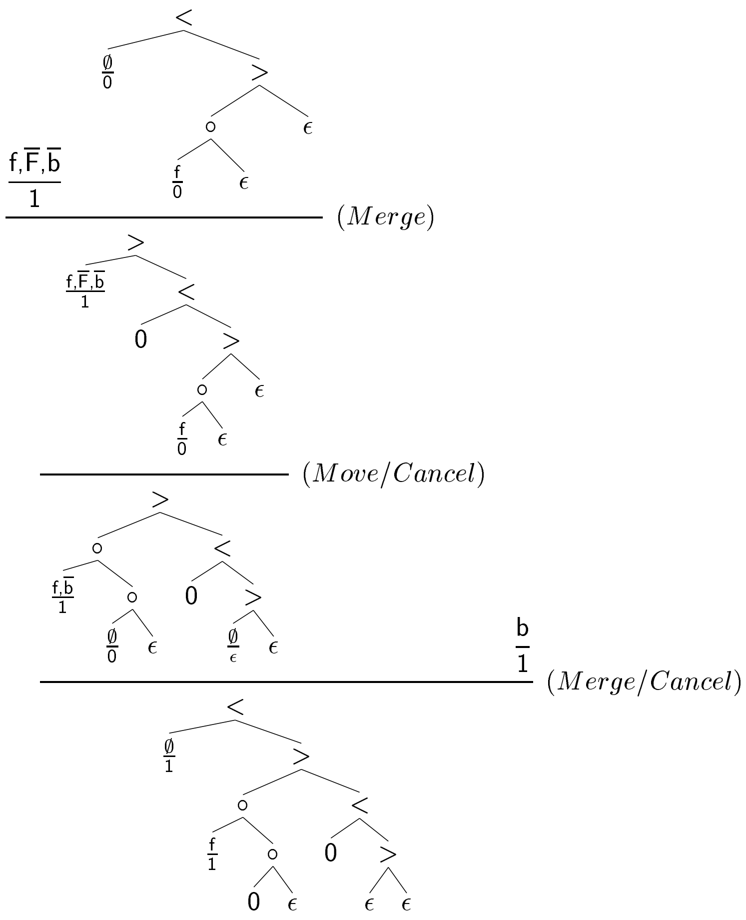



FIGURE 3. Raising a complex head.

Once we attach the a specifier and cancel the a feature we have derived an f-tree, a tree in which f is the only "live" feature. This tree is illustrated as the last step of Figure 2 (also the first step of Figure 3). Again, we could merge this with s, effectively completing the derivation. After the raising of the downstairs head, we would derive an s-tree yielding the string 00, also a member of $\mathcal{L}_{ww}$. But as before we continue by merging in another f-head, this time $\mathsf{f_b}$.

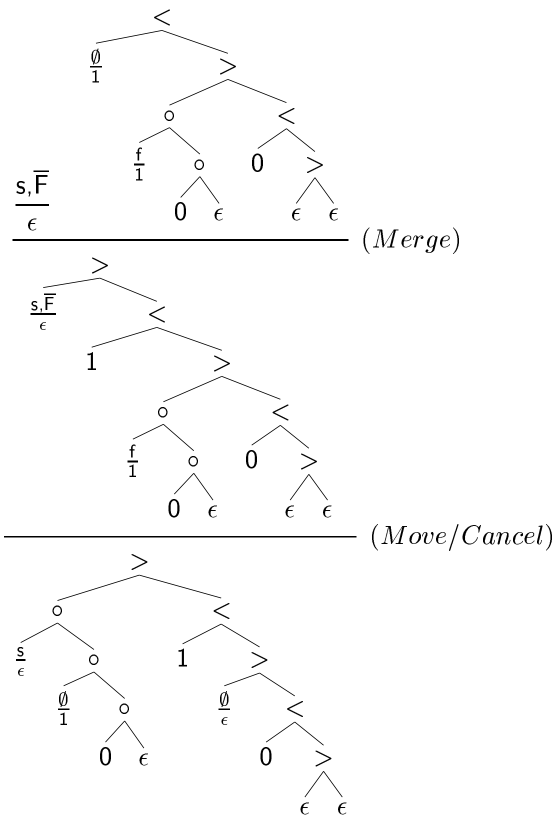



FIGURE 4. Final steps.

This leads to the raising of the complex head, canceling the $\overline{\mathsf{F}}$ feature and exposing the $\overline{\mathsf{b}}$ feature. Merging in a b specifier leaves us with yet another f-tree, the last step in the sub-derivation in Figure 3.

This time we do cap off the derivation by merging in our s. This triggers raising of the complex head and, after feature cancelation, the derivation of an s-tree is complete. As required, it yields the string 1010.

## 5. All and Only $\mathcal{L}_{ww}$

It should be clear that we can derive any string in $\mathcal{L}_{ww}$ with this system. Abstractly, what we are doing is creating a purely right-branching "main" tree, which, again abstracting away from the timing of our derivation steps, represents a string with strictly serial dependencies, consisting of pairs of 0's and 1's. Then head movement strips out the lower members of each of these pairs, constructing a parallel "head" tree which preserves the order of the original tree. So, roughly, from underlying [[11][00]] we derive [[10][10]] by separating all the heads from all the specifiers.

**Proposition 1.** *For any string in the* $\{0,1\}^*$ *copy language, there is an* $\mathsf{s}$*-tree which yields it.*

That we derive *only* strings in $\mathcal{L}_{ww}$ is less clear, and we take some time to prove this in this section. We state the theorem now, and then develop the proof in several steps.

**Theorem 1.** *Every* $\mathsf{s}$*-tree yields a string from the* $\{0,1\}^*$ *copy language.*

We begin with a simple observation about $\mathsf{f}$-trees: every (finite!) $\mathsf{f}$-tree must contain at its base an instance of $\frac{\mathsf{f}}{\epsilon}$ (or, more accurately, a position from which an instance of $\frac{\mathsf{f}}{\epsilon}$ was removed by movement). This is a direct consequence of the recursive structure of the $\mathsf{f}$-categories.

**Lemma 1.** *For all categorial features other than* $\mathsf{s}$*, the numeration must contain an equal number of clashing selection features, and vice-versa.*

This is guaranteed by the linearity of our operations: outside of $\mathsf{s}$, we must establish a bijection between categorial and selection features, so the sets involved must be of the same size.

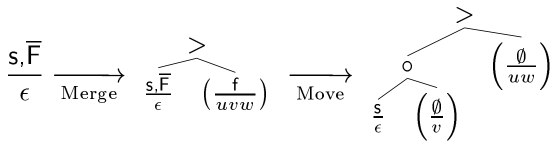
**Lemma 2.** *There can be exactly one instance each of* $\frac{\mathsf{s},\overline{\mathsf{F}}}{\epsilon}$ *and* $\frac{\mathsf{f}}{\epsilon}$ *in any derivation of an* $\mathsf{s}$*-tree.*

Note first that nothing selects $\mathsf{s}$, so by the definition of an $\mathsf{s}$-tree it can only have a single $\frac{\mathsf{s},\overline{\mathsf{F}}}{\epsilon}$ in it. Next note that this one $\frac{\mathsf{s},\overline{\mathsf{F}}}{\epsilon}$ atom that we need to derive an $\mathsf{s}$-tree itself selects $\overline{\mathsf{F}}$. So, by Lemma 1, there must be something in the derivation which introduces a corresponding $\mathsf{f}$ feature. All the recursive $\mathsf{f}$-categories introduce an $\mathsf{f}$ category feature, but also another $\overline{\mathsf{F}}$ feature. So only an instance of nonrecursive $\frac{\mathsf{f}}{\epsilon}$ can balance out the $\overline{\mathsf{F}}$ feature introduced by $\frac{\mathsf{s},\overline{\mathsf{F}}}{\epsilon}$. So we must have at least one $\frac{\mathsf{f}}{\epsilon}$ in our derivation.

The argument works the other way as well. Suppose we have an $\frac{\mathsf{f}}{\epsilon}$ in our derivation. It introduces an $\mathsf{f}$ category feature which must be balanced against an $\overline{\mathsf{F}}$ (or $\overline{\mathsf{f}}$) selection feature. But once again the recursive $\mathsf{f}$ atoms introduce their own $\mathsf{f}$ features for each $\overline{\mathsf{F}}$ that they introduce. So only an $\frac{\mathsf{s},\overline{\mathsf{F}}}{\epsilon}$ can balance the $\mathsf{f}$ feature introduced by an $\frac{\mathsf{f}}{\epsilon}$. Since $\frac{\mathsf{s},\overline{\mathsf{F}}}{\epsilon}$ is unique, $\frac{\mathsf{f}}{\epsilon}$ must be as well. $\square$

Now consider what must happen for us to derive an $\mathsf{s}$-tree. Obviously, we will have to Merge an instance of $\frac{\mathsf{s},\overline{\mathsf{F}}}{\epsilon}$ with an appropriate complement. That hypothetical complement will have to be an $\mathsf{f}$-tree, by the definition of Merge applied to the lexical item $\frac{\mathsf{s},\overline{\mathsf{F}}}{\epsilon}$. Suppose that this $\mathsf{f}$-tree yields the string $uvw$, where $v$ is the part of the string dominated by the $\mathrm{X}^{0\max}$. Then the derivation of an $\mathsf{s}$-tree must proceed as follows.

(6)




Note that there is one alternative possibility in this derivation. Since Merge does not use up the selection feature $\overline{\mathsf{F}}$, it would be possible to apply Merge yet again, merging another inert $\mathsf{f}$-tree as specifier. In fact, you could do this as many times as you liked. However, doing it even once is too much: every $\mathsf{f}$-tree contains an instance of $\frac{\mathsf{f}}{\epsilon}$, and we know already from Lemma 2 that we can only have one of these in a derivation, and we already have one in the complement.

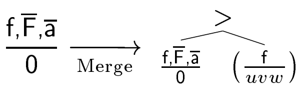
So now we see what we need to prove to establish Theorem 1.

**Lemma 3.** *Every* $\mathsf{f}$*-tree yields a string of the form* $uvw$*, where the* $\mathrm{X}^{0\max}$ *yields* $v$*, and* $v = uw$*.*

The base is trivial: $\frac{\mathsf{f}}{\epsilon}$ dominates the empty string.

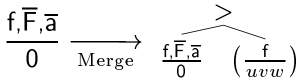
For the inductive step we use the same "partial evaluation" technique that we used above. Suppose for concreteness that we are building an $\mathsf{f}$-tree with $\frac{\mathsf{f},\overline{\mathsf{F}},\overline{\mathsf{a}}}{0}$ as its head. Again, the first thing we must do is to merge $\frac{\mathsf{f},\overline{\mathsf{F}},\overline{\mathsf{a}}}{0}$ with an appropriate complement, namely another $\mathsf{f}$-tree.

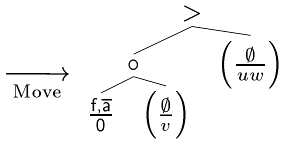
(7)

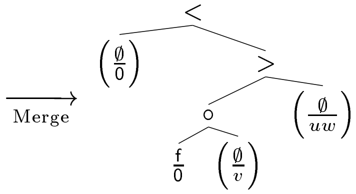



By the inductive hypothesis, this tree yields a string of the form $uvw$, where $v$ is the substring dominated by the head, and it equals $uw$.

Again, note that we could consider merging another $\mathsf{f}$-tree as specifier, but that would introduce too many $\frac{\mathsf{f}}{\epsilon}$'s into the derivation, as we have already observed. So we must move, to cancel the $\overline{\mathsf{F}}$ feature, before the derivation can proceed any further.

(8) 

(9) 

(10) 

Once we have cancelled the $\overline{\mathsf{F}}$ feature we can consider how to cancel the $\overline{\mathsf{a}}$ feature. Theoretically, we could now move an $\mathsf{a}$-tree from within the complement, but since, by assumption, the complement is an $\mathsf{f}$-tree, there can be no such $\mathsf{a}$-tree within it: it contains no live features other than $\mathsf{f}$. Note that we cannot merge more than one $\mathsf{a}$-tree here, since we use up the $\overline{\mathsf{a}}$ feature of the head as soon as we Merge an $\mathsf{a}$-category into its checking domain. So we merge exactly one $\frac{\mathsf{a}}{\mathsf{0}}$, and we are done: we have derived a new $\mathsf{f}$-tree in the only way possible. Checking that our inductive hypothesis is verified, we note that our $\mathsf{f}$-tree dominates a string of the form $u'v'w'$, where $u' = \mathsf{0}$, $v' = \mathsf{0}v$, and $w' = uw$. So we have to show that

(11) $$v = uw \Rightarrow \mathsf{0}v = \mathsf{0}uw$$

which is obvious. □

Theorem 1 now follows immediately, given (6).

## References


Chomsky, N. (1995). *The Minimalist Program*, MIT Press.

Huybregts, R. (1984). The weak inadequacy of context-free phrase structure grammars, *in* de Haan, Trommelen and Zonneveld (eds), *Van Periferie naar Kern*, Foris, Dordrecht, pp. 81–99.

Kracht, M. (1995). Syntactic codes and grammar refinement, *Journal of Logic, Language, and Information* **4**: 41–60.

Kurtonina, N. and Moortgat, M. (1996). Structural control, *in* P. Blackburn and M. de Rijke (eds), *Specifying Syntactic Structures*, CSLI Publications, Stanford, CA.

Moortgat, M. (1996). In situ binding: A modal analysis, *in* Dekker and Stokhof (eds), *Proceedings 10th Amsterdam Colloquium*, ILLC, Amsterdam, pp. 539–549.

Rogers, J. (1994). *Studies in the Logic of Trees with Applications to Grammar Formalisms*, PhD thesis, University of Delaware, Department of Computer & Information Sciences.

Shieber, S. M. (1985). Evidence against the context-freeness of natural language, *Linguistics and Philosophy* **8**: 333–343.

Stabler, E. P. (1996a). *Acquiring and parsing languages with movement*, Basil Blackwell, Oxford. (Drafts available at http://128.97.8.34/utrecht/utrecht.html.).

Stabler, E. P. (1996b). Computing quantifier scope, *in* A. Szabolcsi (ed.), *Ways of Scope Taking*, Kluwer, Boston.



Seminar für Sprachwissenschaft, University of Tübingen

*E-mail address*: `cornell@sfs.nphil.uni-tuebingen.de`