Title: Top-Down Recognition of Minimalist Languages

Author: Henk Harkema

Affiliation: Département de Linguistique
Université de Québec à Montréal
Montréal, Québec, Canada, H3C 3P8

E-mail: harkema.hendrik@courrier.uqam.ca

Abstract:

This paper describes a top-down recognition method for languages generated by Minimalist Grammars. Minimalist Grammars are formal grammars that incorporate certain aspects of current transformational linguistic theories: phrases are derived by applying structure building functions to lexical items and intermediate structures, and the applicability of these functions is determined by the syntactic features of the structures involved. The recognition method presented in this paper reduces phrase structures to simple expressions that encode the behavior of these structures with regard to the structure building functions of the grammar.

Top-Down Recognition of Minimalist Languages

1 Introduction

In this paper, we will describe a method for top-down recognition of languages generated by Minimalist Grammars (Stabler, 1997). Minimalist Grammars are a rigorous formalization of the kind of grammars proposed in the linguistic framework of Chomsky's Minimalist Program (Chomsky, 1995). The grammar formalism will be introduced in section 2.

Harkema (2000) presents a chart-based, bottom-up recognizer for Minimalist Grammars. A bottom-up parser has no predictive power. Hence, the chart will contain numerous items that are not involved in the derivation of the sentence to be parsed. In particular, in order to be complete, a bottom-up parser has to assume that any phonetically empty category can intervene between any two adjacent words in a sentence. A top-down parser, however, has the ability to predict the presence of phonetically empty categories in a sentence based on the structure built so far. Phonetically empty functional projections are ubiquitous in present-day transformational theories of language, so this issue is an important motivation for the formulation of a top-down recognizer for Minimalist Grammars. The design of the top-down recognizer will be discussed in sections 3 and 4. The accuracy of the predictions of the recognizer can be improved by adding a mechanism for look-ahead. This will be discussed in section 5.

The paper will close with some final remarks.

2 Minimalist Grammars

In this section, we will present the Minimalist Grammar formalism, following Stabler (1997). A Minimalist Grammar defines a set of trees. These trees are derived by closing the lexicon, which is a set of trees itself, under two structure building functions, *merge* and *move*. Consider for example Minimalist Grammar G_1 with a lexicon containing the following 6 elements:

(1) `d -k` *Lavinia*

(2) `d -k` *Titus*

(3) `=d vt -v` *praise*

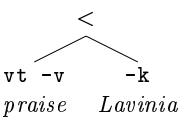
(4) `=pred +v +k i` *s*

(5) `=i c` ϵ

(6) `=vt +k =d pred` ϵ

Each lexical item is a 1-node tree labeled with syntactic features of various kinds and a phonetic form. (note 1) The last two lexical items are phonetically empty. The features determine how the structure building functions will apply to the lexical items and the trees derived from these. Two trees, one of which has a feature `=x` and the other of which has a feature `x`, will trigger an application of the structure building function *merge*. For example, the lexical items `=d vt -v` *praise* and `d -k` *Lavinia* will merge to form the tree below:

(7)



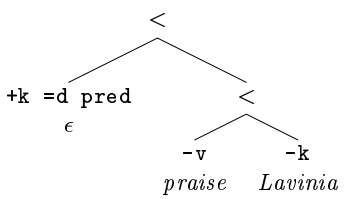
The resulting tree includes the original expressions `=d vt -v` *praise* and `d -k` *Lavinia*, minus the pair of features `=d` and `d` that triggered the function *merge*. These features have been checked and deleted. The '<' points to the head of the tree. For *merge*, the head of the tree that has the feature `=x` (before this feature is deleted by *merge*) will be the head of the resulting tree. The features of the head of the tree determine what other structure building functions will apply. The order of the features matters: the application of the structure building functions is triggered by the left-most features of the sequences of features of the expressions involved. Thus, *merge* does not apply to the lexical items `=vt +k =d pred` ϵ and `=d vt -v` *praise*, because the feature `vt` is not left-most in the lexical item `=d vt -v` *praise*.

A tree that consists of more than one node, such as the tree in 7, is called a complex tree. Lexical items are simple trees, as they consist of just one node. We will use the following notation for complex trees: $[_{<}\tau,\ \upsilon]$ denotes a complex tree with immediate subtrees τ and υ, τ preceding υ, whose head is the head of τ; similarly, $[_{>}\tau,\ \upsilon]$ denotes a complex tree with immediate subtrees τ and υ, τ preceding υ, whose head is the head of υ. For example, in this notation the tree in 7 is written as $[_{<}$`vt -v` *praise*, `-k` *Lavinia*$]$. The head of a simple tree is the single node making up the simple tree.

The structure building function *merge* is defined in the following way. A pair of trees τ, υ is in the domain of *merge* if the left-most feature of the head of τ is `=x` and the left-most feature of the head of υ is `x`. Then $merge(\tau,\ \upsilon) = [_{<}\tau',\ \upsilon']$ if τ is simple, and $merge(\tau,\ \upsilon) = [_{>}\upsilon',\ \tau']$ if τ is complex, where τ' is like τ except that feature `=x` is deleted, and υ' is like υ except that feature `x` is deleted. So simple trees take sisters to their right, and complex trees take sisters to their left.

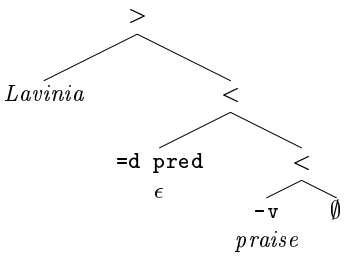
The derivation will continue with merging the lexical item `=vt +k =d pred` ϵ and the tree in 7. The result is given in 8.

(8)



The left-most feature of the head of the tree in 8 is `+k`. The tree also contains a node whose left-most feature is `-k`. In this situation, the structure building function *move* will apply. It will move the maximal subtree whose head has the feature `-k` to the specifier position of the original tree, as in 9 below. A subtree ϕ is maximal if any subtree τ properly containing ϕ has a head other than ϕ. In this case, the subtree that moves is the tree consisting of the single node `-k` *Lavinia*. As in the case of *merge*, the features triggering the application of *move* are checked and deleted.

(9)

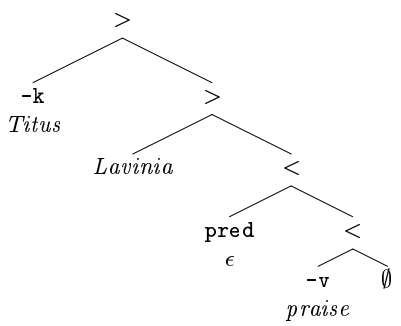


Formally, the structure building function *move* is defined as follows. A tree τ is in the domain of *move* if the left-most feature of the head of τ is `+f` and τ has exactly one maximal subtree τ_0 the left-most feature of the head of which is `-f`. Then $move(\tau) = [_> \tau_0', \tau']$, where τ_0' is like τ_0 except that feature `-f` is deleted, and τ' is like τ except that feature `+f` is deleted and subtree τ_0 is replaced by a single node without features. A node without features will be labeled $\emptyset$. (note 2)

All derivations in a Minimalist Grammar are subject to the Shortest Movement Constraint, which is built in into the definition of the domain of the structure building functions *move*: this function does not apply to a tree if the left-most feature of the head of the tree is `+f` and the tree contains more than one subtree whose head begins with the feature `-f`. In this case, all subtrees want to move to the same position, but moving any one subtree will deprive the other subtrees of their "shortest move", as they will now have to move to the specifier of some higher head which has the feature `+f`. (note 3)

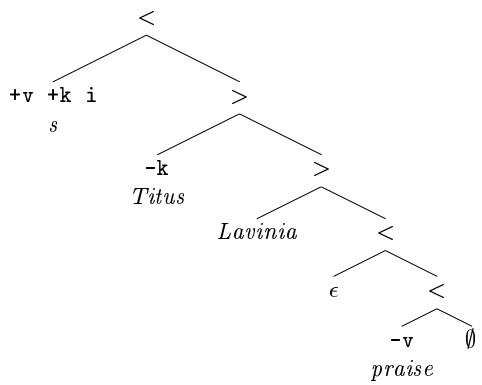
The left-most feature of its head being `=d`, the tree in 9 can merge with the lexical item `d -k` *Titus*. Because the tree in 9 is a complex tree, the second clause of the definition of *merge* will apply. The result is the tree in 10.

(10)



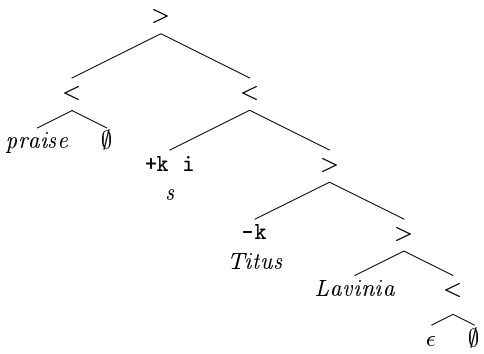
The head of the tree in 10 is labeled `pred`. The tree will be merged with the lexical item `=pred +v +k i` *s*. This will produce the tree in 11.

(11)



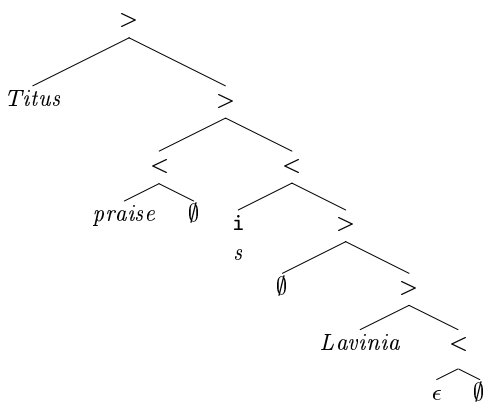
The next step of the derivation is another *move*, triggered by the feature `+v` on the head of the tree in 11 and the feature `-v` on the head of one of its subtrees. This is an instance of remnant movement: the object of the verb phrase that moves is no longer inside the verb phrase. The tree in 12 is the result.

(12)



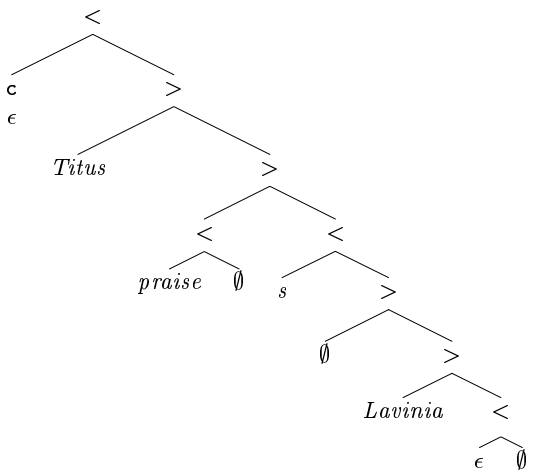
The pair of features `+k` and `-k` in the tree in 12 will trigger another application of *move*, which will yield the tree in 13.

(13)



Finally, the tree in 13, whose head has the feature `i`, will be selected by the lexical item `=i c` ϵ. The result of this final *merge* is the tree in 14. This tree contains only one unchecked syntactic feature, namely `c`, and this feature labels the head of the tree. We have thus established that the yield of the tree in 14, ϵ *Titus praise s Lavinia* ϵ, i.e., *Titus praises Lavinia*, is a string of category `c`, using an analysis along the lines of Mahajan (2000) for deriving simple SVO sentences without head movement. (note 4)

(14)

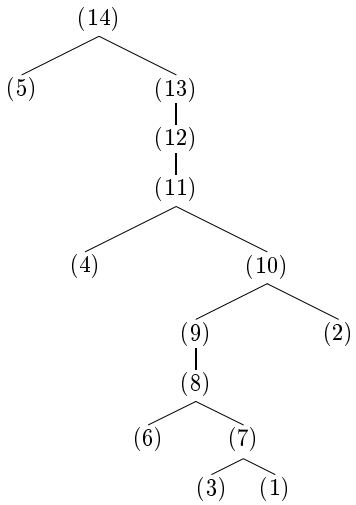


The total number of features occurring in the lexical items involved in the derivation of the sentence *Titus praises Lavinia* is 17. Therefore, the derivation of the tree in 14 takes exactly 8 steps, for each application of *merge* and *move* removes 2 features and the tree in 14 has one single feature left.

Formally, the language derivable by a Minimalist Grammar G consists of the yields of the complete trees in the closure of the lexicon under the structure building functions *merge* and *move*, where a complete tree is a tree without syntactic features, except for the distinguished feature `c`, which must label the head of the tree. The yield of a tree is the concatenation of the phonetic forms appearing at the leaves of the tree, ordered as in the tree.

Michaelis (1998) has demonstrated that the set of languages generated by Minimalist Grammars falls within the set of languages generated by Multiple Context-Free Grammars (Seki et al. 1991). Harkema (2001a) and Michaelis (2001) present proofs showing the reverse. Hence, Minimalist Grammars are equivalent to Multiple Context-Free Grammars, which, in turn, are known to be equivalent to Linear Context-Free Rewriting Systems (Vijay-Shanker et al., 1987), Multi-Component Tree-Adjoining Grammars (Weir, 1988), and Simple Positive Range Concatenation Grammars (Boullier, 1998).

(15)



The derivation of the sentence *Titus praises Lavinia* is summarized in the derivation tree given in 15. In this tree, the numbered nodes stand for the trees given above, including the lexical items. A derivation tree indicates how a sentence is derived from a set of lexical items via the structure building functions *merge* and *move*. The leaves of a derivation tree are lexical items, which are simple trees. Each internal node is a complex tree that is derived from its immediate daughters by one application of the structure building functions *merge* or *move*. The root of a derivation tree is a complete tree for the sentence being derived. A derivation tree is thus a tree whose nodes are trees themselves.

3 Context-Free Derivations

Michaelis (1998) has shown that the class of languages defined by Minimalist Grammars is included in the class of languages defined by Multiple Context-Free Grammars (Seki et al., 1991) by demonstrating how to construct a Multiple Context-Free Grammar G′ which defines the same language as a given Minimalist Grammar G. For every tree τ generated by grammar G, there will be a non-terminal symbol A in grammar G′ which encodes the purely syntactic properties of τ, that is, its behavior with regard to the structure building functions *merge* and *move*. These properties are completely determined by the syntactic

features appearing in the tree and whether the tree is simple or complex. The phonetic forms appearing in a tree are not relevant for the applicability of *merge* and *move*, nor does the geometry of the tree matter, except for the fact that the syntactic features of the head of the tree should be distinguished from any syntactic features appearing at other nodes in the tree. Hence, for syntactic purposes, a tree may be collapsed into a sequence of sequences of syntactic features. These are the non-terminal symbols of the Multiple Context-Free Grammar G′, which will be referred to as categories.

Formally, a category is a sequence of the form $[\gamma_0 \cdot \delta_0, \ldots, \gamma_n \cdot \delta_n]_t$, with γ_j, $\delta_j \in \mathrm{Cat}^*$, $0 \leq j \leq n$, $t \in \{c, s, l\}$, and Cat the set of syntactic features of Minimalist Grammar G. A category A represents a set of trees $\mathrm{T}_c(\mathrm{A})$. A tree τ is in $\mathrm{T}_c([\gamma_0 \cdot \delta_0, \ldots, \gamma_n \cdot \delta_n]_t)$ if, and only if, the following four conditions are met:

1. If $t = s$, τ is a simple tree; if $t = c$, τ is a complex tree; and if $t = l$, τ is a lexical tree. (note 5)
2. For every i, $0 \leq i \leq n$, there is a leaf l_i in τ such that:
   (a) The syntactic features of l_i are δ_i.
   (b) The maximal subtree headed by l_i is a projection of a lexical item with features $\gamma_i \delta_i$. (note 6)
3. Leaf l_0 is the head of τ.
4. Besides the leaves $l_0, \ldots, l_n$, there are no other leaves with syntactic features in τ.

For example, the tree in 4 is an element of $\mathrm{T}_c([\cdot \texttt{=pred +v +k i}]_s)$, the tree in 10 is an element of $\mathrm{T}_c([\texttt{=vt +k =d}\cdot\texttt{pred}, \texttt{d}\cdot\texttt{-k}, \texttt{=d vt}\cdot\texttt{-v}]_c)$, and the tree in 11 is an element of $\mathrm{T}_c([\texttt{=pred}\cdot\texttt{+v +k i}, \texttt{d}\cdot\texttt{-k}, \texttt{=d vt}\cdot\texttt{-v}]_c)$.

One can think of a category as an abbreviation of a possibly infinite set of trees. We will only be interested in relevant categories, that is, categories A for which $\mathrm{T}_c(\mathrm{A})$ includes a tree generated by grammar G which does not violate the Shortest Movement Constraint. Any tree τ that violates the Shortest Movement Constraint is useless in that it is not a complete tree by itself and cannot participate in the derivation of a complete tree. The structure building function *move* does not apply to such a tree τ, wherefore its outstanding `-f` features cannot be deleted. The crucial observation in Michaelis (1998) is that the set of relevant categories for Minimalist Grammar G is finite. This is essential for grammar G′ to be definable at all, since a Multiple Context-Free Grammar cannot have an infinite number of non-terminal symbols or categories.

With regard to the phonetic forms of the trees generated by G, any category A in G′ will be associated with a tuple of strings, rather than with just one string, as in plain Context-Free Grammars. For a category A corresponding to a tree τ generated by G, this tuple of strings includes all and only the phonetic forms appearing in τ. Phonetic forms that will move independently from one

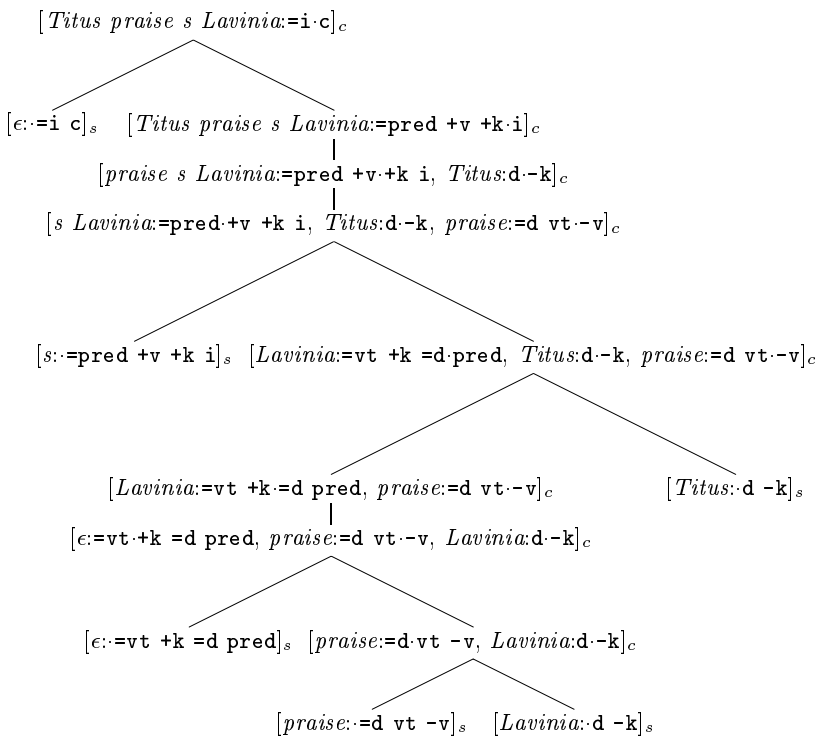
another will be kept separate. In order to formalize this notion, let the narrow yield Y_n of a tree be defined as follows. The narrow yield of a simple tree ϕ is its yield as defined in the previous section. If ϕ is a complex tree, then, in case $\phi = [_{>}\tau, \upsilon]$, $\mathrm{Y}_n(\phi) = \mathrm{Y}_n(\tau)^\frown\mathrm{Y}_n(\upsilon)$ if the left-most feature of the head of τ is not of the kind `-f`, and $\mathrm{Y}_n(\phi) = \mathrm{Y}_n(\upsilon)$ otherwise. (note 7) If $\phi = [_{<}\tau, \upsilon]$, then $\mathrm{Y}_n(\phi) = \mathrm{Y}_n(\tau)^\frown\mathrm{Y}_n(\upsilon)$ if the left-most feature of the head of υ is not of the kind `-f`, and $\mathrm{Y}_n(\phi) = \mathrm{Y}_n(\tau)$ otherwise. Then, a category $\mathrm{A} = [\gamma_0{\cdot}\delta_0, \ldots, \gamma_n{\cdot}\delta_n]_t$ as defined above will be associated with a tuple of strings $(s_0, \ldots, s_n)$ if, and only if, there is a tree $\tau \in \mathrm{T}_c(\mathrm{A})$ for which the narrow yield of the maximal projection of leaf l_i in τ labeled with syntactic features δ_i is s_i, $0 \leq i \leq n$.

For example, for the tree given in 10, the narrow yield of the maximal projection of the leaf labeled *pred*, i.e. the narrow yield of the tree itself, is *Lavina* ϵ = *Lavina*; for the leaf labeled `-k` it is *Titus*; and for the leaf labeled `-v` it is *praise*. Hence, the category [`=vt +k =d·pred`, `d·-k`, `=d vt·-v`]$_c$ to which tree 10 belongs will be associated with the triple (*Lavinia*, *Titus*, *praise*). Similarly, the category [·`=pred +v +k i`]$_s$ of tree 4 will be associated with the 1-tuple (*s*), and the category [`=pred·+v +k i`, `d·-k`, `=d vt·-v`]$_c$ of tree 11 will be associated with the triple (*s Lavinia*, *Titus*, *praise*). We can conveniently combine a category and its associated tuple of phonetic forms and write [*Lavinia*:`=vt +k =d·pred`, *Titus*:`d·-k`, *praise*:`=d vt·-v`]$_c$ for the tree in 10, for example.

In order to derive the categories and their associated tuples of strings, grammar G′ contains a set of context-free rewrite rules which mirror the effects of the structure building operations *merge* and *move* in grammar G. Since the set of relevant categories is finite, a finite set of rewrite rules suffices to describe all derivations. Thus the Multiple Context-Free Grammar G′ for Minimalist Grammar G_1 contains the context-free rewrite rule [$p^\frown q$:`=pred·+v +k i`, r:`d·-k`, t:`=d vt·-v`] → [p:·`=pred +v +k i`] [q:`=vt +k =d·pred`, r:`d·-k`, t:`=d vt·-v`]. This rule mirrors the step in the derivation in which the tree in 11 is derived from the tree in 4 and the tree in 10. The rule will derive the category with phonetic forms [*s Lavinia*:`=pred·+v +k i`, *Titus*:`d·-k`, *praise*:`=d vt·-v`]$_c$ from [*s*:·`=pred +v +k i`]$_s$ and [*Lavinia*:`=vt +k =d·pred`, *Titus*:`d·-k`, *praise*:`=d vt·-v`]$_c$.

Using categories to abbreviate trees, the derivation tree in 15, whose nodes are trees generated by grammar G_1, can be represented as the tree in 16.

(16)



The translation of a Minimalist Grammar G into an equivalent Multiple Context-Free Grammar G′ reveals that the derivation trees of Minimalist Grammars are context-free, even though the languages defined by Minimalist Grammars are beyond the power of Context-Free Grammars. (note 8) The next section will show how this insight can be used to obtain a transparent top-down recognizer for languages generated by Minimalist Grammars.

Note that the derivation tree in 16 illustrates a crucial difference between Minimalist Grammars and Context-Free Grammars: a depth-first, left-to-right traversal of a derivation tree for some sentence w generated by a Minimalist Grammar will in general not visit the lexical expressions at the leaves of the tree in the order in which their phonetic forms appear w, whereas for a Context-Free Grammar, the words at the leaves of a (derivation) tree, ordered according to the precedence relation of the tree, always constitute a grammatical sentence.

4 Top-Down Recognition

Like its bottom-up counterpart in Harkema (2000), the top-down recognizer for Minimalist Grammars that is presented in this section is based on the principle of parsing as deduction as described in Shieber et al. (1995). The recognizer uses a chart to store items, which embody predictions about the syntactic structure of the sentence to be recognized. Initially, the chart is filled with a set of axioms.

These axioms are then closed under a set of rules of inference. If the closure contains a distinguished goal item, the sentence is in the language defined by the grammar, otherwise it is not.

4.1 Items and Invariant

An item is a sequence $\Delta_1 + \ldots + \Delta_m$, where each subitem Δ_i, $1 \leq i \leq m$, is a prediction regarding the syntactic features and narrow yields of a tree involved in the derivation of the input sentence. A subitem is a category with position vectors and is of the general form $[(\mathrm{x}_0, \mathrm{y}_0){:}\gamma_0{\cdot}\delta_0, \ldots, (\mathrm{x}_n, \mathrm{y}_n){:}\gamma_n{\cdot}\delta_n]_t$, with γ_j, δ_j, and t as in the previous section, and x_j, $\mathrm{y}_j \in \mathbb{N}$, $0 \leq j \leq n$. A subitem Δ abbreviates a set of trees $\mathrm{T}_s(\Delta)$. For an input string $w = w_1 \ldots w_k$ and a subitem $\Delta = [(\mathrm{x}_0, \mathrm{y}_0){:}\gamma_0{\cdot}\delta_0, \ldots, (\mathrm{x}_n, \mathrm{y}_n){:}\gamma_n{\cdot}\delta_n]_t$, tree a τ is in $\mathrm{T}_s(\Delta)$ if, and only if, the following holds:

1. $\tau \in \mathrm{T}_c([\gamma_0{\cdot}\delta_0, \ldots, \gamma_n{\cdot}\delta_n]_t)$.
2. The narrow yield of the maximal projection of leaf l_i in τ labeled δ_i is $w_{x_i+1} \ldots w_{y_i}$, $0 \leq i \leq n$.

On the assumption that sentences are strings of the distinguished category c, the items generated by the recognizer will be interpreted under the following invariant: given an item $\Delta_1 + \ldots + \Delta_m$, the sequence of trees $\tau_1, \ldots, \tau_m$, for arbitrary $\tau_i \in \mathrm{T}_s(\Delta_i)$, $0 \leq i \leq m$, is a cut of a (partial) derivation tree whose root is a complete tree ρ of category c with yield w. (note 9) (note 10) Recall that the nodes of a derivation tree are trees generated by a Minimalist Grammar (cf. the derivation tree in 15).

4.2 Axioms

If c is the distinguished feature of grammar G and k is the length of the input string w, then for any lexical item with the single syntactic feature c there will be an axiom $[(0, k){:}{\cdot}\mathsf{c}]_s$, and for any lexical item with syntactic features $\gamma\mathsf{c}$, there will be an axiom $[(0, k){:}\gamma{\cdot}\mathsf{c}]_c$, $\gamma \in \mathrm{Cat}^+$.

As is easily checked, any $\tau \in \mathrm{T}_s([(0, k){:}{\cdot}\mathsf{c}]_s)$ or $\tau \in \mathrm{T}_s([(0, k){:}\gamma{\cdot}\mathsf{c}]_c)$ constitutes a cut through the root of a partial derivation tree whose root is a complete tree of category c with yield w; the root is τ.

4.3 Rules of Inference

The rules of inference encode the context-free rewrite rules of the Multiple Context-Free Grammar G′ that is equivalent to Minimalist Grammar G. Grammar G′ contains a rewrite rule for every single application of the structure building functions *merge* and *move*, but the definitions of these functions allow us to condense these rewrite rules into five rules of inference: three Unmerge rules and two Unmove rules. Given a particular subitem Δ_i appearing in some item $\Delta_1 + \ldots + \Delta_m$, the rules of inference will predict how the trees in $\mathrm{T}_s(\Delta_i)$

can be 'unmerged' and 'unmoved' into smaller trees by specifying the particular subitems that these smaller trees belong to. In addition to the Unmerge and Unmove rules, there is also a Scan rule for reading the input string.

Unmerge-1: given an item $\Delta_1 + \ldots + \Delta_m$ such that $\Delta_i = [(p, q):\texttt{=x}\cdot\gamma, \mathrm{S}]_c$, $1 \leq i \leq m$, then, for any lexical item with syntactic features $\beta\texttt{x}$, add the items $\Delta_1 + \ldots + \Delta_{i-1} + [(p, v):\cdot\texttt{=x}\gamma]_s + [(v, q):\beta\cdot\texttt{x}, \mathrm{S}]_t + \Delta_{i+1} + \ldots + \Delta_m$ to the chart, for all possible values of v such that $p \leq v \leq q$; for all possible values of t such that $t = s$ if $\beta = \emptyset$, $t = c$ if $\beta \neq \emptyset$, and if $t = s$ then $\mathrm{S} = \emptyset$.

Unmerge-2: given an item $\Delta_1 + \ldots + \Delta_m$ such that $\Delta_i = [(p, q):\alpha\texttt{=x}\cdot\gamma, \mathrm{S}]_c$, $1 \leq i \leq m$, $\alpha \neq \emptyset$, then, for any lexical item with syntactic features $\beta\texttt{x}$, add the items $\Delta_1 + \ldots + \Delta_{i-1} + [(v, q):\alpha\cdot\texttt{=x}\gamma, \mathrm{U}]_c + [(p, v):\beta\cdot\texttt{x}, \mathrm{V}]_t + \Delta_{i+1} + \ldots + \Delta_m$ to the chart, for all possible values of v such that $p \leq v \leq q$; for all possible values of U, V such that $\mathrm{U} \cup \mathrm{V} = \mathrm{S}$; for all possible values of t such that $t = s$ if $\beta = \emptyset$, $t = c$ if $\beta \neq \emptyset$, and if $t = s$ then $\mathrm{V} = \emptyset$.

Unmerge-3: given an item $\Delta_1 + \ldots + \Delta_m$, such that $\Delta_i = [(p, q):\alpha\texttt{=x}\cdot\gamma, \mathrm{S}, (v, w):\beta\texttt{x}\cdot\delta, \mathrm{T}]_c$, $1 \leq i \leq m$, then add the items $\Delta_1 + \ldots + \Delta_{i-1} + [(p, q):\alpha\cdot\texttt{=x}\gamma, \mathrm{U}]_{t_1} + [(v, w):\beta\cdot\texttt{x}\delta, \mathrm{V}]_{t_2} + \Delta_{i+1} + \ldots + \Delta_m$ to the chart, for all possible values of U, V such that $\mathrm{U} \cup \mathrm{V} = \mathrm{S} \cup \mathrm{T}$; for all possible values of t_1 such that $t_1 = s$ if $\alpha = \emptyset$, $t_1 = c$ if $\alpha \neq \emptyset$, and if $t_1 = s$ then $\mathrm{U} = \emptyset$; for all possible values of t_2 such that $t_2 = s$ if $\beta = \emptyset$, $t_2 = c$ if $\beta \neq \emptyset$, and if $t_2 = s$ then $\mathrm{V} = \emptyset$.

Unmove-1: given an item $\Delta_1 + \ldots + \Delta_m$, such that $\Delta_i = [(p, q):\alpha\texttt{+y}\cdot\gamma, \mathrm{S}]_c$, $1 \leq i \leq n$, $\alpha \neq \emptyset$, then, for any lexical item with syntactic features $\beta\texttt{-y}$, $\beta \neq \emptyset$, add the items $\Delta_1 + \ldots + \Delta_{i-1} + [(v, q):\alpha\cdot\texttt{+y}\gamma, (p, v):\beta\cdot\texttt{-y}, \mathrm{S}]_c + \Delta_{i+1} + \ldots + \Delta_m$ to the chart, for all possible values of v such that $p \leq v \leq q$.

Unmove-2: given an item $\Delta_1 + \ldots + \Delta_m$, such that $\Delta_i = [(p, q):\alpha\texttt{+y}\cdot\gamma, \mathrm{S}, (v, w):\beta\texttt{-y}\cdot\delta, \mathrm{T}]_c$, $1 \leq i \leq m$, $\alpha \neq \emptyset$, $\beta \neq \emptyset$, add the items $\Delta_1 + \ldots + \Delta_{i-1} + [(p, q):\alpha\cdot\texttt{+y}\gamma, \mathrm{S}, (v, w):\beta\cdot\texttt{-y}\delta, \mathrm{T}]_c + \Delta_{i+1} + \ldots + \Delta_m$ to the chart.

Scan: given an item $\Delta_1 + \ldots + \Delta_m$, such that $\Delta_i = [(p, q):\cdot\gamma]_s$, $1 \leq i \leq m$, then, if there is a lexical item ℓ with syntactic features γ and phonetic features covering $w_{p+1} \ldots w_q$ of the input string, i.e. $\ell \in \mathrm{T}_s(\Delta_i)$, add the following item to the chart: $\Delta_1 + \ldots + \Delta_{i-1} + [(p, q):\cdot\gamma]_l + \Delta_{i+1} + \ldots + \Delta_m$.

The rules of inference Unmove-1 and Unmove-2 come with the restriction that no items violating the Shortest Movement Constraint should be added to the chart. An item violates the Shortest Movement Constraint if it contains a subitem $[(\mathrm{x}_0, \mathrm{y}_0):\gamma_0\cdot\delta_0, \ldots, (\mathrm{x}_n, \mathrm{y}_n):\gamma_n\cdot\delta_n]_t$ such that there are δ_i, δ_j whose left-most features are the same feature $\texttt{-f}$, $1 \leq i < j \leq n$.

4.4 Goal Items

Any item of the form $[(x_0, y_0){:}\cdot\gamma_0]_l + \ldots + [(x_m, y_m){:}\cdot\gamma_m]_l$ is a goal item. It follows from the definition of the Scan rule that there are lexical items $\ell_i \in T_s([(x_i, y_i){:}\cdot\gamma_i]_l)$, $0 \leq i \leq m$. According to the invariant, the sequence $\ell_1, \ldots, \ell_m$ is a cut of a partial derivation tree whose root is a complete tree ρ of category `c` with yield w. Since all ℓ_i, $0 \leq i \leq m$, are lexical items, the derivation tree is in fact a 'complete' derivation tree, which means that w is in the language defined by G.

4.5 Example

The recognition of the example sentence $w = {}_0\mathit{Titus}_1\mathit{praise}_2\mathit{s}_3\mathit{Lavinia}_4$ according to grammar G_1 given in section 2. will start with the assertion of the single axiom $[(0, 4){:}\texttt{=i}\cdot\texttt{c}]_c$. This item represents a cut through the root of the derivation tree in 15. Of course, for each cut of the tree in 15, there is a corresponding cut of the derivation tree in 16.

Rule Unmerge-1 will apply to the axiom, as there is a lexical item with features `=pred +v +k i`. One of the items produced is $[(0, 0){:}\cdot\texttt{=i c}]_s + [(0, 4){:}\texttt{=pred +v +k}\cdot\texttt{i}]_c$. This item corresponds to a cut through the nodes labeled 5 and 13 in derivation tree in 15. Rule Unmerge-1 also produces four other items: $[(0, p){:}\cdot\texttt{=i c}]_s + [(p, 4){:}\texttt{=pred +v +k}\cdot\texttt{i}]_c$, $1 \leq p \leq 4$. These four additional items also obey the invariant defined in section 4.1, but they do not correspond to a cut of the particular derivation tree in 15.

The item $[(0, 0){:}\cdot\texttt{=i c}]_s + [(0, 4){:}\texttt{=pred +v +k}\cdot\texttt{i}]_c$ can be rewritten in two ways. Either the Scan rule will apply to produce the item $[(0, 0){:}\cdot\texttt{=i c}]_l + [(0, 4){:}\texttt{=pred +v +k}\cdot\texttt{i}]_c$, or rule Unmove-1 will apply to produce the item $[(0, 0){:}\cdot\texttt{=i c}]_s + [(1, 4){:}\texttt{=pred +v}\cdot\texttt{+k i}, (0, 1){:}\texttt{d}\cdot\texttt{-k}]_c$ (plus another four items $[(0, 0){:}\cdot\texttt{=i c}]_s + [(p', 4){:}\texttt{=pred +v}\cdot\texttt{+k i}, (0, p'){:}\texttt{d}\cdot\texttt{-k}]_c$, $p' = 0$ or $2 \leq p' \leq 4$). Item $[(0, 0){:}\cdot\texttt{=i c}]_s + [(0, 4){:}\texttt{=pred +v}\cdot\texttt{+k i}\ (0, 0){:}\texttt{d}\cdot\texttt{-k}]_c$ corresponds to a cut through the nodes labeled 5 and 12 of the tree in 15. Rule Unmove-1 will also apply to the other four items produced in the previous step ($1 \leq p \leq 4$), but Scan will not.

Eventually, the item $[(0, 0){:}\cdot\texttt{=i c}]_l + [(2, 3){:}\cdot\texttt{=pred +v +k i}]_l + [(4, 4){:}\cdot\texttt{=vt +k =d pred}]_l + [(1, 2){:}\cdot\texttt{=d vt -v}]_l + [(3, 4){:}\cdot\texttt{d -k}]_l + [(0, 1){:}\cdot\texttt{d -k}]_l$ will be deduced. This is a goal item, corresponding to a cut through the leaves of the derivation tree in 15.

4.6 Correctness and Complexity

The recognizer presented above is sound and complete: every deducible item respects the invariant given in section 4.1, and for every string that is in the language defined by the grammar, there is a deduction of a goal item from the axioms. The correctness proofs are provided in Harkema (2001b). There it is also shown that the time complexity of the recognizer is polynomial in the length of the input sentence (considering the shortest deduction of a goal item for a sentence).

Unfortunately, since items produced by the recognizer correspond to cuts of derivation trees, the recognizer will fail to halt if the grammar is recursive. A Minimalist Grammar is recursive if there is a complete tree τ which is derived from a tree τ_1 via *merge* and *move*, and tree τ_1, in turn, is derived from a tree τ_2 such that τ_1 and τ_1 belong to the same category, i.e., there is a category A such that $\tau_1 \in \mathrm{T}_c(\mathrm{A})$ and $\tau_2 \in \mathrm{T}_c(\mathrm{A})$. (note 11)

5 Look-Ahead

The example in the previous section illustrates that the recognizer will produce items which do not represent cuts of a derivation tree for the sentence to be recognized. These superfluous items occur because the rules of inference will split the position vectors of subitems in every possible way. Another situation in which spurious items are deduced is when a category can be derived in more than one way. Consider for example grammar G_2, which is grammar G_1 of section 2 with two additional lexical items: `=i +wh c` ϵ and `d -k -wh` *who*. The language defined by G_2 will not only contain the declarative sentences defined by G_1, but also interrogative sentences such as *who Titus praise s*. (note 12) For grammar G_2 and any given sentence, the recognizer presented in the previous section will deduce items for either type of sentence, even though one can tell from looking at the first word of the sentence whether it is declarative or interrogative. In the remainder of this section we will develop a method of look-ahead for Minimalist Grammars, which will restrict the number of items produced by the inference rules without compromising the completeness of the recognizer.

5.1 $\mathbf{First}_k$

Given a Minimalist Grammar G and a relevant category $\mathrm{A} = [\gamma_0{\cdot}\delta_0, \ldots, \gamma_n{\cdot}\delta_n]_t$, the set $\mathrm{First}_k(\mathrm{A})$ is defined as follows: $(s_0, \ldots, s_n) \in \mathrm{First}_k(\mathrm{A})$ if, and only if, there is a tree $\tau \in \mathrm{T}_c(\mathrm{A})$ and for each leaf l_i in τ labeled with syntactic features δ_i, the first k symbols of the narrow yield of the maximal projection of l_i constitute the string s_i, $0 \leq i \leq n$.

We can compute the sets First_k using a slightly adapted version of the bottom-up recognizer from Harkema (2000). The items of the bottom-up recognizer are very similar to the subitems in the top-down method defined above: they are sequences of the form $[(\mathrm{x}_0, \mathrm{y}_0){:}\gamma_0{\cdot}\delta_0, \ldots, (\mathrm{x}_n, \mathrm{y}_n){:}\gamma_n{\cdot}\delta_n]_t$. To compute First_k, the position vectors $(\mathrm{x}_i, \mathrm{y}_i)$ will be replaced by strings s_i. A revised bottom-up item $[\gamma_0{\cdot}\delta_0/s_0, \ldots, \gamma_n{\cdot}\delta_n/s_n]_t$ is understood to assert the existence of a tree τ generated by G such that:

1. $\tau \in \mathrm{T}_c([\gamma_0{\cdot}\delta_0, \ldots, \gamma_n{\cdot}\delta_n]_t)$.
2. For each leaf l_i in τ labeled with syntactic features δ_i, the first k symbols of the narrow yield of the maximal projection of l_i constitute the string s_i, $0 \leq i \leq n$.

Thus, an item $[\gamma_0 \cdot \delta_0 / s_0, \ldots, \gamma_n \cdot \delta_n / s_n]_t$ claims that $(s_0, \ldots, s_n) \in \text{First}_k([\gamma_0 \cdot \delta_0, \ldots, \gamma_n \cdot \delta_n]_t)$.

The revised bottom-up recognizer will start with the following set of axioms: for each lexical item ℓ of G labeled with syntactic features $\gamma \in \text{Cat}^+$ and phonetic form p, there will be an axiom $[\cdot\gamma / k{:}p]_s$, where $k{:}s$ denotes a prefix of length k of string s; if $|s| < k$, $k{:}s = s$. The axioms will be closed under the following five rules of inference:

Merge-1: given two items $[\cdot\texttt{=x}\gamma / p]_s$ and $[\beta \cdot \texttt{x} / q, \text{S}]_t$, add the item $[\texttt{=x} \cdot \gamma / k{:}(p^\frown q), \text{S}]_c$ to the chart.

Merge-2: given two items $[\beta \cdot \texttt{=x}\gamma / p, \text{S}]_c$ and $[\mu \cdot \texttt{x} / q, \text{T}]_t$, add the item $[\beta \texttt{=x} \cdot \gamma / k{:}(q^\frown p), \text{S}, \text{T}]_c$ to the chart.

Merge-3: given two items $[\beta \cdot \texttt{=x}\gamma / p, \text{S}]_t$ and $[\mu \cdot \texttt{x}\delta / q, \text{T}]_t$ $(\delta \neq \emptyset)$, add the item $[\beta \texttt{=x} \cdot \gamma / p, \text{S}, \mu\texttt{x} \cdot \delta / q, \text{T}]_c$ to the chart.

Move-1: given an item $[\beta \cdot \texttt{+y}\gamma / p, \text{S}, \mu \cdot \texttt{-y} / q, \text{T}]_c$, add the item $[\beta \texttt{+y} \cdot \gamma / k{:}(q^\frown p), \text{S}, \text{T}]_c$ to the chart.

Move-2: given an item $[\beta \cdot \texttt{+y}\gamma / p, \text{S}, \mu \cdot \texttt{-y}\delta / q, \text{T}]_c$ $(\delta \neq \emptyset)$, add the item $[\beta \texttt{+y} \cdot \gamma / p, \text{S}, \mu \texttt{-y} \cdot \delta / q, \text{T}]_c$ to the chart.

The rules Move-1 and Move-2 are constrained so as not to apply to an item that violates the Shortest Movement Constraint.

The proof of soundness provided in Harkema (2000) is easily converted into a proof of soundness for the method for computing First_k proposed above. Thus, every deducible item $[\gamma_0 \cdot \delta_0 / s_0, \ldots, \gamma_n \cdot \delta_n / s_n]_t$ truthfully claims that $(s_0, \ldots, s_n) \in \text{First}_k([\gamma_0 \cdot \delta_0, \ldots, \gamma_n \cdot \delta_n]_t)$. Furthermore, one can prove that the method is complete up to permutation of the last n elements of an item, that is, for every true claim $(s_0, \ldots, s_n) \in \text{First}_k([\gamma_0 \cdot \delta_0, \ldots, \gamma_n \cdot \delta_n]_t)$, an item $[\gamma_0 \cdot \delta_0 / s_0, \gamma_{j_1} \cdot \delta_{j_1} / s_{j_1}, \ldots, \gamma_{j_n} \cdot \delta_{j_n} / s_{j_n}]_t$ will be deduced, where the sequence of indices $j_1, \ldots, j_n$ is a permutation of the sequence $1, \ldots, n$. Consequently, for any category $\text{A} = [\gamma_0 \cdot \delta_0, \ldots, \gamma_n \cdot \delta_n]_t$ such that $\text{T}_c(\text{A})$ contains some tree τ generated by G, the contents of set First_k can be established in the following way:

> $\text{First}_k([\gamma_0 \cdot \delta_0, \ldots, \gamma_n \cdot \delta_n]_t) = \{ (s_0, \ldots, s_n) \mid$ item $[\gamma_0 \cdot \delta_0 / s_0, \gamma_{j_1} \cdot \delta_{j_1} / s_{j_1}, \ldots, \gamma_{j_n} \cdot \delta_{j_n} / s_{j_n}]_t$ has been generated, and $j_1, \ldots, j_n$ is a permutation of $1, \ldots, n\}$.

Suppose tree ϕ participates in the derivation of a complete tree. Then it can be shown that the sequences of syntactic features labeling nodes in ϕ other than the head must be sequences of features of the kind `-f`. This implies that we can restrict our attention to relevant categories of the general form $[\gamma_0 \cdot \delta_0, \gamma_1 \cdot \texttt{-f}_1\delta'_1, \ldots, \gamma_m \cdot \texttt{-f}_n\delta'_n]$. The Shortest Movement Constraint implies that $\texttt{-f}_i \neq \texttt{-f}_j$, for $1 \leq i, j \leq n$ and $i \neq j$. Then, for a category $\text{A} = [\gamma_0 \cdot \delta_0, \gamma_1 \cdot \texttt{-f}_1\delta'_1, \ldots,$

$\gamma_m \cdot \texttt{-f}_n \delta'_n]$, define $\text{First}_k^0(\text{A}) = \{\text{z}_0 |\ (\text{z}_0, \ldots, \text{z}_n) \in \text{First}_k(\text{A})\}$ and $\text{First}_k^{-\texttt{f}_i}(\text{A}) = \{\text{z}_i |\ (\text{z}_0, \ldots, \text{z}_i, \ldots, \text{z}_n) \in \text{First}_k(\text{A})\}$. Thus, $\text{First}_k^0(\text{A})$ picks out the set containing the first k symbols of the narrow yield of the maximal projection of the head of any tree $\tau \in \text{T}_c(\text{A})$, (note 13) and $\text{First}_k^{-\texttt{f}_i}(\text{A})$ picks out the set containing the first k symbols of the narrow yield of the maximal projection of the node whose left-most syntactic feature is $\texttt{-f}_i$ for any tree $\tau \in \text{T}_c(\text{A})$.

For the categories in grammar G_2, we thus find, for example, $\text{First}_1([\texttt{=pred} \cdot \texttt{+v +k i}, \texttt{d} \cdot \texttt{-k}, \texttt{=d vt} \cdot \texttt{-v}]_c) = \{(s, \textit{Titus}, \textit{praise}), (s, \textit{Lavinia}, \textit{praise})\}$ and $\text{First}_1^{-\texttt{k}}([\texttt{=pred} \cdot \texttt{+v +k i}, \texttt{d} \cdot \texttt{-k}, \texttt{=d vt} \cdot \texttt{-v}]_c) = \{\textit{Titus}, \textit{Lavinia}\}$. Also, $\text{First}_1^0([\texttt{=i} \cdot \texttt{c}]_c) = \{\textit{Titus}, \textit{Lavinia}\}$ and $\text{First}_1^0([\texttt{=i +wh} \cdot \texttt{c}]_c) = \{\textit{who}\}$.

5.2 Recognition with Look-Ahead

The recognizer with look-ahead will check, for every new item that is about to be added to the chart according to the rules of inference given in section 4.3, whether it is harmonious with the input sentence by consulting the various sets First_k^0 and $\text{First}_k^{-\texttt{y}}$.

For a Minimalist Grammar G with distinguished category c, input string $w = w_1 \ldots w_n$, and look-ahead k, the item $[(0, n) : \cdot \texttt{c}]_s$ will only be an axiom if $k{:}w \in \text{First}_k^0([\cdot \texttt{c}]_s)$, and the item $[(0, n) : \gamma \cdot \texttt{c}]_c$ will only be an axiom if $k{:}w \in \text{First}_k^0([\gamma \cdot \texttt{c}]_c)$. The goal items are defined as before. The rules of inference are updated in the following way.

Unmerge-1 will add an item $\Delta_1 + \ldots + \Delta_{i-1} + [(p, v) : \cdot \texttt{=x}\gamma]_s + [(v, q) : \beta \cdot \texttt{x}, \text{S}]_t + \Delta_{i+1} + \ldots + \Delta_m$ to the chart provided that $k{:}(w_{p+1} \ldots w_v) \in \text{First}_k^0([\cdot \texttt{=x}\gamma]_s)$ and $k{:}(w_{v+1} \ldots w_q) \in \text{First}_k^0([\beta \cdot \texttt{x}, \text{S}]_t)$.

Unmerge-2 will add an item $\Delta_1 + \ldots + \Delta_{i-1} + [(v, q) : \alpha \cdot \texttt{=x}\gamma, \text{U}]_c + [(p, v) : \beta \cdot \texttt{x}, \text{V}]_t + \Delta_{i+1} + \ldots + \Delta_m$ to the chart provided that $k{:}(w_{v+1} \ldots w_q) \in \text{First}_k^0([\alpha \cdot \texttt{=x}\gamma, \text{U}]_c)$ and $k{:}(w_{p+1} \ldots w_v) \in \text{First}_k^0([\beta \cdot \texttt{x}, \text{V}]_t)$.

Unmerge-3 will add an item $\Delta_1 + \ldots + \Delta_{i-1} + [(p, q) : \alpha \cdot \texttt{=x}\gamma, \text{U}]_{t_1} + [(v, w) : \beta \cdot \texttt{x}\delta, \text{V}]_{t_2} + \Delta_{i+1} + \ldots + \Delta_m$ to the chart provided that $k{:}(w_{p+1} \ldots w_q) \in \text{First}_k^0([\alpha \cdot \texttt{=x}\gamma, \text{U}]_{t_1})$ and $k{:}(w_{v+1} \ldots w_w) \in \text{First}_k^0([\beta \cdot \texttt{x}\delta, \text{V}]_{t_2})$. (note 14)

Unmove-1 will add an item $\Delta_1 + \ldots + \Delta_{i-1} + [(v, q) : \alpha \cdot \texttt{+y}\gamma, (p, v) : \beta \cdot \texttt{-y}, \text{S}]_c + \Delta_{i+1} + \ldots + \Delta_m$ to the chart provided that $k{:}(w_{v+1} \ldots w_q) \in \text{First}_k^0([\alpha \cdot \texttt{+y}\gamma, \beta \cdot \texttt{-y}, \text{S}]_c)$ and $k{:}(w_{p+1} \ldots w_v) \in \text{First}_k^{-\texttt{y}}([\alpha \cdot \texttt{+y}\gamma, \beta \cdot \texttt{-y}, \text{S}]_c)$.

Unmove-2 will add an item $\Delta_1 + \ldots + \Delta_{i-1} + [(p, q) : \alpha \cdot \texttt{+y}\gamma, \text{S}, (v, w) : \beta \cdot \texttt{-y}\delta, \text{T}]_c + \Delta_{i+1} + \ldots + \Delta_m$ to the chart provided that $k{:}(w_{p+1} \ldots w_q) \in \text{First}_k^0([\alpha \cdot \texttt{+y}\gamma, \text{S}, \beta \cdot \texttt{-y}\delta, \text{T}]_c)$ and $k{:}(w_{v+1} \ldots w_w) \in \text{First}_k^{-\texttt{y}}([\alpha \cdot \texttt{+y}\gamma, \text{S}, \beta \cdot \texttt{-y}\delta, \text{T}]_c)$.

The Scan rule remains unchanged.

The reduction in the number of items brought about by the use of First_k depends on particularities of the grammar and the input sentence. When the grammar describes a natural language, beneficial effects are expected, because in natural languages most words serve as the first word of only a limited number of categories. For grammar G_2, the savings are significant. Without look-ahead, i.e., $k = 0$, recognition of the sentence ${}_0\mathit{Titus}_1\mathit{praise}_2\mathit{s}_3\mathit{Lavinia}_4$ involves the deduction of 340 items. For $k = 1$, it only takes 15 items. (note 15) The recognizer with look-ahead does not consider the item $[(0,\ 4)\text{:}\texttt{=i +wh·c}]_c$ an axiom, because $w_1 = \mathit{Titus} \notin \text{First}_1^0([\texttt{=i +wh·c}]_c)$. Hence, all items deducible from this item are suppressed. This example also illustrates the advantage of using dots in items. Without dots, categories $[\texttt{=i +wh·c}]_c$ and $[\texttt{=i ·c}]_c$ would be collapsed into one category $[\texttt{c}]_c$ for which $\text{First}_1^0([\texttt{c}]_c) = \{\mathit{Titus},\ \mathit{Lavinia},\ \mathit{who}\}$, whence the recognizer would not be able to immediately dismiss either the declarative or the interrogative analysis. Furthermore, the recognizer with look-ahead will split all position vectors correctly for G_2. For example, unmerging the axiom $[(0,\ 4)\text{:}\texttt{=i·c}]_c$, the recognizer will not posit the item $[(0,\ 1)\text{:}\texttt{·=i c}]_s + [(1,\ 4)\text{:}\texttt{=pred +v +k·i}]_c$, since $w_1 = \mathit{Titus} \notin \text{First}_1^0([\texttt{·=i c}]_s) = \emptyset$ and $w_2 = \mathit{praise} \notin \text{First}_1^0([\texttt{=pred +v +k·i}]_c) = \{\mathit{Lavinia},\ \mathit{Titus}\}$.

6 Conclusions

In this paper, we have motivated the usefulness of a top-down approach to recognizing languages defined by Minimalist Grammars. We described the design of a pure top-down recognizer and added a mechanism for look-ahead. The recognizer can be turned into a parser by extending items with a field for recording their immediate ancestors, and using this field to retrieve trees or forests from the chart of items produced by the algorithm.

Because of its complexity and potential non-termination, the recognizer *per se* is of limited practical value. However, the top-down perspective on Minimalist Grammars that is explored in this paper provides an understanding of the grammar formalism that is very useful for formulating an Earley-style recognizer for Minimalist Grammars, which will terminate for all grammar and sentence pairs (Harkema, 2001b).

Many interesting questions of a formal nature remain open, e.g. how to characterize Minimalist Grammars whose languages can be parsed deterministically in a top-down manner with at most k symbols of look-ahead, where, different from the mechanism described in section 5, the look-ahead string is a single contiguous substring of the input sentence, starting at the immediate right of the last (overt) word that was scanned.

The techniques outlined in this paper can also be used to design a recognizer for Asymmetry Grammars (e.g. Di Sciullo, 1999). The structure building functions of these grammars are sensitive to certain configurational properties of the trees they apply to, so these properties have to be encoded in the categories that represent trees. Since the rules of inference of the recognizer are very closely related to the structure building functions of the grammar, this

will show in what ways the asymmetry inherent in Asymmetry Grammars is computationally advantageous.

Notes

0. The parser of the human sentence processor also seems to have the correct prefix property: the ungrammaticality of a sentence is detected at the first word that makes the sentence ungrammatical, and for garden path sentences the human parser will generally hesitate at the first word that does not fit into the structure that is hypothesized for the sentence.

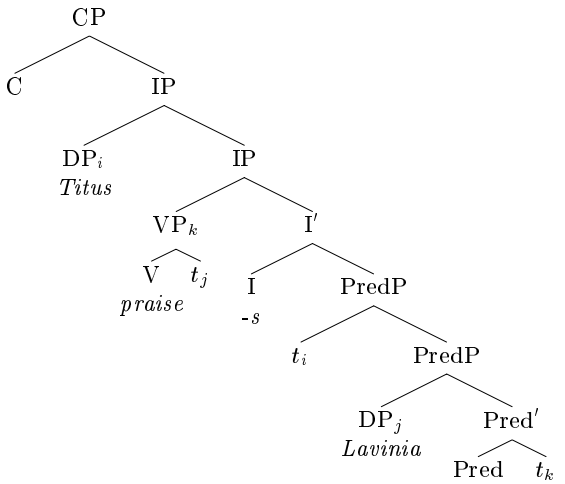
1. Lexical items also have semantic features, but we will not discuss these features in this paper.

2. The structure building function *move* defined here implements overt phrasal movement. The Minimalist Grammars specified in Stabler (1997) also include mechanisms for head movement and covert phrasal movement. We will not discuss these kinds of movement in this paper. However, the recognizer formulated in this paper can be extended to deal with Minimalist Grammars that allow head movement and covert movement; see Stabler (2001), Harkema (2001b).

3. The Shortest Movement Constraint formulated in Stabler (1997) is a rather strong condition. If the head of the tree has more than one `+f` features, the subtrees whose heads begin with the feature `-f` could all move to become specifiers of the same head, which would make these moves equally short from a linguistic point of view. However, under the current version of the Shortest Movement Constraint a tree with more than one exposed `-f` feature is not in the domain of the function *move*, so no movements will take place.

4. In traditional notation, the tree in 14 would be presented as in 14b.

(14b)



5. All lexical trees are assumed to be simple trees, but not all simple trees are lexical trees. For example, the tree `d -k` ϵ is a simple tree, but it is not a lexical tree in

grammar G_1 discussed in section 2.

6. Any lexical item ℓ is a projection of itself. If τ is a projection of ℓ and τ, υ are in the domain of *merge*, then $\phi = merge(\tau, \upsilon)$ is a projection of ℓ. If τ is a projection of ℓ and τ is in the domain of *move*, then $\phi = move(\tau)$ is a projection of ℓ.

7. For two strings p and q, $p^\frown q$ denotes their concatenation in the obvious order.

8. The derivation tree in 16 illustrates a crucial difference between Context-Free Grammars and Minimalist Grammars: a depth-first, left-to-right traversal of a derivation tree for some sentence w generated by a Minimalist Grammar will in general not visit the lexical expressions at the leaves of the tree in the order in which their phonetic forms appear in w, whereas for a Context-Free Grammar the words at the leaves of a (derivation) tree, ordered according to the precedence relation of the tree, always constitute a grammatical sentence.

9. A partial derivation tree is a derivation tree whose root is a complete tree, but whose leaves are not necessarily lexical items.

10. A sequence of nodes C of a tree T is a cut of T if (1) none of the nodes in C dominates another node in C, (2) every node in T and not in C either dominates or is dominated by a node in C, and (3) the nodes in C are ordered according to the precedence relation of T.

11. If the recognizer is made to scan the words of the sentence from left-to-right, the recognizer will fail to halt if the grammar is left-recursive, analogously to top-down recognizers for Context-Free Grammars. A Minimalist Grammar is left-recursive if it is recursive, i.e., there are trees τ, τ_1, and τ_2 with the properties mentioned in the text, and moreover the phonetic material of τ_2 contains a string that in the sentence w specified by τ is to the left of all the phonetic forms of τ_2 – see Harkema (2001b) for more discussion.

12. Grammar G_2 ignores *do*-support. See Stabler (2001) for a more complete Minimalist Grammar covering question formation in English.

13. Note that the maximal projection of the head of τ is τ itself.

14. Note that the application of rule Unmerge-3 must be preceded by an application of rule Unmove-1, because in a bottom-up derivation, any application of *move* must be preceded by an application of *merge* which will contribute the tree that will move. Therefore, the condition $k{:}(w_{v+1} \dots w_w) \in \text{First}_k^0([\beta{\cdot}\mathtt{x}\delta,\ \text{V}]_c)$ for Unmerge-3 will be evaluated after the condition $k{:}(w_{v+1} \dots w_w) \in \text{First}_k^{-\mathtt{y}}([\alpha{\cdot}\mathtt{+y}\gamma,\ \beta\mathtt{x}\delta'{\cdot}\mathtt{-y},\ \text{S}]_c)$, $\delta = \delta'\mathtt{-y}$, for Unmove-1 has been found true. A similar point can be made for the condition involving $\text{First}_k^{-\mathtt{y}}$ in rule Unmove-2. Hence, if in general $\text{First}_k^0([\beta{\cdot}\mathtt{x}\delta,\ \text{V}]_c) = \text{First}_k^{-\mathtt{y}}([\alpha{\cdot}\mathtt{+y}\gamma,\ \beta\mathtt{x}\delta'{\cdot}\mathtt{-y},\ \text{S}]_c)$, one of the conditions on either Unmerge-3 or Unmove-1 and Unmove-2 can be dropped. In that case an arrangement is possible in which all conditions involve checking the contents of First_k^0 rather than $\text{First}_k^{-\mathtt{y}}$.

15. This is the least number of items possible: 1 axiom, 8 items corresponding to the 8 intermediate trees in the derivation of the sentence, plus 6 scanned items.

References

Boullier, P. 1998. *Proposal for a Natural Language Processing Syntactic Backbone* [Research Report 3342]. Rocquencourt, France: INRIA-Rocquencourt.

Chomsky, N. 1995. *The Minimalist Program.* Cambridge, MA: MIT Press.

Di Sciullo, A.M. 1999. "The local asymmetry connection". In *MIT Working Papers in Linguistics*, 35: 25 - 47.

Harkema, H. 2000. "A recognizer for minimalist grammars". In *Proceedings of the 6th International Workshop on Parsing Technologies*, 111 - 122. Trento, Italy: ITC-IRST.

Harkema, H. 2001a. "A characterization of minimalist grammars". In *Logical Aspects of Computational Linguistics* [Lecture Notes in Artificial Intelligence 2099], P. de Groote, G.F. Morrill and C. Retoré (eds), 193 - 211. Berlin, Germany: Springer.

Harkema, H. 2001b. *Parsing Minimalist Languages.* PhD thesis. Los Angeles, CA: UCLA.

Mahajan, A. 2000. "Word order and remnant movement: Eliminating head movement". In *GLOW Newsletter* 44: 44 - 45. Tilburg, the Netherlands: Tilburg University.

Michaelis, J. 1998. "Derivational minimalism is mildly context-sensitive". In *Logical Aspects of Computational Linguistics* [Lecture Notes in Artificial Intelligence 2014], M. Moortgat (ed.), 179 - 198. Berlin, Germany: Springer.

Michaelis, J. 2001. "Transforming linear context-free rewriting systems into minimalist grammars". In *Logical Aspects of Computational Linguistics* [Lecture Notes in Artificial Intelligence 2099], P. de Groote, G.F. Morrill and C. Retoré (eds), 228 - 244. Berlin, Germany: Springer.

Seki, H, Matsumura, T., Fujii, M, and Kasami, T. 1991. "On multiple context-free grammars". In: *Theoretical Computer Science*, 88: 191 - 229.

Shieber, S.M., Schabes, Y., and Pereira, F.C.N. 1995. "Principles and implementation of deductive parsing". *Journal of Logic Programming* 24 (1, 2): 3 - 36.

Stabler, E.P. 1997. "Derivational minimalism". In *Logical Aspects of Computational Linguistics* [Lecture Notes in Artificial Intelligence 1328], C. Retoré (ed.), 68 - 95. Berlin, Germany: Springer.

Stabler, E.P. 2001. "Recognizing head movement". In *Logical Aspects of Computational Linguistics* [Lecture Notes in Artificial Intelligence 2099], P. de Groote, G.F. Morrill and C. Retoré (eds), 228 - 244. Berlin, Germany: Springer.

Vijay-Shanker, K., Weir, D.J., and Joshi, A.K. 1987. "Characterizing descriptions produced by various grammatical formalisms". In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, 104 - 111. Stanford, CA: Stanford University.

Weir, D.J. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms.* PhD thesis. Philadelphia, PA: University of Pennsylvania.