Memory Resource Allocation in Top-Down Minimalist Parsing*

Gregory M. Kobele[1], Sabrina Gerth[2], and John Hale[3]

[1] University of Chicago, Chicago, Illinois, USA
[2] Universität Potsdam, Potsdam, Germany
[3] Cornell University, Ithaca, New York, USA

Abstract. This paper provides a linking theory between the minimalist grammar formalism and off-line behavioural data. We examine the transient stack states of a top-down parser for Minimalist Grammars as it analyzes embedded sentences in English, Dutch and German. We find that the number of time steps that a derivation tree node persist on the parser's stack derives the observed contrasts in English center embedding, and the difference between German and Dutch embedding. This particular stack occupancy measure formalizes the leading idea of "memory burden" in a way that links predictive, incremental parsing to specific syntactic analyses.

1 Introduction

An important goal of linguistics is to account for human linguistic behavior. A cognitive scientist might, following David Marr [25], attempt to analyze behavioral data by viewing the syntactician's grammatical analysis as a high level description of a parser (cf. [20]). Deviations from the categoricity 'predicted' by the grammar could be given a natural explanation using the more refined vocabulary of parser states, and memory resource consumption.

A number of theoretical proposals within psycholinguistics over the years [13, 42, 6, 22] have been built around the idea that our capacity to remember words and phrases plays an important role in normal human language comprehension. These theories link observed processing contrasts between sentences to differences in theorized memory requirements. These contrasts are observed empirically in sentence types whose grammatical analysis is a matter of active research within the field of syntax. This presents a problem: ideally, the same analysis that is supported by comparative linguistic research ought to derive observed processing contrasts. But in some previous memory burden theories, syntactic assumptions were left implicit or oversimplified (but cf. [32]). Under these conditions, it becomes difficult to say exactly which aspect or aspects of sentence structure motivates the memory burdens to which the theory appeals as explanations. Yet an emerging body of work suggests that the choice of syntactic analysis may matter greatly [9, 40] (cf. section 5.2).

* An anonymous reviewer provided very helpful comments, which have greatly improved the quality of this paper.

This work strives to address this problem. We proceed by defining grammar fragments whose syntactic commmitments are clear. Starting from a mildly context sensitive grammar [37] we assume a top-down parsing strategy [36]. This additional assumption is motivated by evidence for predictive human sentence comprehension [26, 39]. Using independently motivated syntactic analyses, we derive complexity profiles consistent with two key contrasts that have been traditionally acknowledged in the literature. An additional benefit of this is that the pathway to semantic interpretation is entirely standard [30, 18].

2 Methodology

We investigate a measure of parsing complexity for minimalist grammars [37], a formalism based on transformational generative grammar [4]. Parsing, like other nondeterministic algorithms, is naturally viewed as a *search* problem [14], where the nondeterminicity is resolved by an independent search strategy such as depth-first, breadth-first or A*. In an ideal parser, heuristics would perfectly guide its actions at every choice point. In this optimal case, the amount of resources consumed during a parse of a sentence is identical to the amount of resources needed to traverse its parse tree (no additional resources are consumed by working on ultimatey 'incorrect' parses, and backtracking). We adopt this simplifying assumption here, noting that something like it (beam search with a narrow beam) has been proposed as a natural way to capture the 'dual nature – generally good and occasionally pathological – of human linguistic performance' [5].

Our methodology extends [12]. The main idea is to advance a particular automaton model of sentence processing, examining certain aspects of the automaton's configuration as it parses strings whose abstract structure mirrors that of the sentences used in comprehension experiments with humans. We explore a measure of parsing complexity based on the allocation of memory resources during a successful parse. One memory unit is considered allocated per item on the parser stack, where the parser stack holds predictions yet to be verified. Following Joshi and Rambow [12, 33], our complexity metric reflects the amount of "time" that an item is retained in memory; we call this *stack tenure* (even though the actual data structure is a priority queue, cf. §4.1). From the state-trajectory of the top-down automaton over the correct derivation tree, we calculate the number of time steps that an item is retained. The length of longest sequence of automaton transitions for which the same item remains in memory we identify as the *maximal tenure* of the parse.

The remainder of this paper is structured as follows. Section 3 briefly introduces the processing phenomena that this modelling work addresses. Section 4 then introduces Minimalist Grammars, and a top-down parser for them. Section 5 reports the stack tenure measures obtained by simulating the parser on the sentences in question. Section 6 takes up the relationship between this approach and other related proposals in the literature. Section 7 concludes, and mentions some directions for future work.

3 Embedding phenomena

The notion of tenure (to be introduced in §4.1) can be thought of as a way of formalising intuitions about dependency length [6]. The following phenomena are naturally understood in these terms, and we will see (§5) that this intuitive understanding can in fact be cashed out rigorously in terms of tenure.

3.1 English center embedding, as compared to right-branching

Center embedding in English is notoriously difficult to comprehend, in comparison to a right-branching alternative [28]. Center-embedded examples like 1 below can be made more and more difficult by interposing additional relative clause modifiers after each new subject noun phrase. This embedded material intervenes between the subject and the verb. By contrast, in right-branching examples like 2, it is the object that is modified. In these cases, the distance between subject and verb remains the same.

(1) The boy that the girl that the cat licked loves laughed.
(2) The cat licked the girl that loves the boy that laughed.

Resnik [34] (cf. [11, 1]) expresses what has become the standard account of this contrast. He proposes that an explanation for the radical unacceptability of center embedded sentences can be made to follow elegantly from constraints on memory resources in a parser. He shows that a left-corner, but not a bottom-up or a top-down, parser for a context-free grammar requires more memory to process center embedded structures than peripherally embedded ones.

It is, however, widely accepted [35] that context-free grammars are unable to assign structures to sentences which allow for a transparant description of their meaning. Unfortunately, once we move to more sophisticated grammar formalisms, which *do* seem able to assign semantically appropriate structures to sentences, the notion of left-corner parsing is either ill-defined or not yet discovered. In other words, the explanation of the processing difficulty of center embedded structures based on imposing memory restrictions on a left-corner parser does not transfer directly to linguistically plausible grammar formalisms. To reconcile the evidence motivating mild context-sensitivity with the selectively greater difficulty of center-embedding requires some alternative automaton model, such as the one to be presented in section 4.

3.2 Embedded vs cross-serial verb clusters

Bach et al. [2] observe that the rated comprehensibility of German embeddings with sentence-final verb clusters (3) increases more sharply than does the corresponding rating of Dutch cross-serial items (4), as the number of verbs in the cluster grows.

(3) daß Hans Peter Marie schwimmen lassen sah
that Hans Peter Mary swim let saw
"that Hans saw Peter let Mary swim"

(4) dat Jan Piet Marie zag laten zwemmen
that Jan Peter Mary saw let swim
"that Jan saw Peter let Mary swim"

In other words, German examples such as 3 are more difficult to process than Dutch examples such as 4.

From a formal perspective, this is surprising, as the Dutch cross serial pattern, under the semantically appropriate pairing of nouns and verbs, is mildly context sensitive ($N_1\ N_2\ N_3\ V_1\ V_2\ V_3 \in ww$), whereas the German nested pattern ($N_1\ N_2\ N_3\ V_3\ V_2\ V_1 \in ww^R$) can be generated by a less-expressive context-free grammar. This is thus an example where language theoretic complexity does not coincide with 'behavioural complexity'. A natural intuition is to link this contrast to the length of the dependencies between nouns and their selecting verbs – in the Dutch case, the longest dependency is checked first, whereas in German it is checked last (N_1 and V_1).

4 Minimalist Grammars

Minimalist grammars make use of two syntactic structure building operations; binary **merge** and unary **move**. Whether a structure building operation is defined on a particular object in its domain (a pair of expressions or a single expression) is determined solely by the syntactic categories of these objects. In minimalist grammars, syntactic categories are finite sequences of 'features'. The currently accessible feature is the feature at the beginning (leftmost) position of the list. In order for **merge** to apply, the heads of its arguments must have matching first features. These features are eliminated in the derived structure which results from their merger. In the case of **move**, the head of its argument must have a feature matching a feature of the head of one of its subconstituents. In the result, both features are eliminated. Each feature type has an attractor and an attractee variant, and for two features to match, one must be an attractor and the other an attractee. For **merge**, the attractee feature is a simple categorial feature, written x. There are two kinds of attractor features, =x and x=, depending on whether the selected expression is to be merged on the right (=x) or on the left (x=). For the **move** operation, there is a single attractor feature, written +y, and a single attractee, −y.

We write lexical items using the notation $\langle \sigma, \delta \rangle$, where σ is a (phonological) string, and δ is a feature bundle. Complex expressions are written using the notation of [37] for the 'bare phrase structure' trees of [4]. These trees are essentially X-bar trees without phrase and category information represented at internal nodes. Instead, internal nodes are labeled with 'arrows' > and <, which point to the head of their phrase. A tree of the form $[< \alpha\ \beta]$ indicates that the head is to be found in the subtree α, and we say that α projects over β, while one of the form $[> \alpha\ \beta]$ that its head is in β, and we say that β projects over α. Leaves are labeled with lexeme/feature pairs (and so a lexical item $\langle \alpha, \delta \rangle$ is a special case of a tree with only a single node). The head of a tree t is the leaf one arrives at from the root by following the arrows at the internal nodes. If t is a bare phrase structure tree with head H, then I will write $t[\mathrm{H}]$ to indicate this. (This means we can write lexical items $\langle \alpha, \delta \rangle$ as $\langle \alpha, \delta \rangle[\langle \alpha, \delta \rangle]$.) The **merge** operation is defined on a pair of trees t_1, t_2 if and only if the head of t_1 has a feature bundle which begins with

either =x or x=, and the head of t_2 has a feature bundle beginning with the matching x feature. The bare phrase structure tree which results from the merger of t_1 and t_2 has t_1 projecting over t_2, which is attached either to the right of t_1 (if the first feature of the head was =x) or to the left of t_1 (if the first feature of the head was x=). In either case, both selection features are checked in the result.

$$\textbf{merge}(t_1[\langle \alpha, \texttt{=x}\delta\rangle], t_2[\langle \beta, \texttt{x}\gamma\rangle]) = \begin{array}{ccc} & < & \\ t_1[\langle \alpha, \delta\rangle] & & t_2[\langle \beta, \gamma\rangle] \end{array}$$

$$\textbf{merge}(t_1[\langle \alpha, \texttt{x=}\delta\rangle], t_2[\langle \beta, \texttt{x}\gamma\rangle]) = \begin{array}{ccc} & > & \\ t_2[\langle \beta, \gamma\rangle] & & t_1[\langle \alpha, \delta\rangle] \end{array}$$

If the selecting tree is both a lexical item and an affix (which I notate by means of a hyphen preceding/following the lexeme in the case of a suffix/prefix), then head movement is triggered from the head of the selected tree to the head of the selecting tree.

$$\textbf{merge}(\langle \text{-}\alpha, \texttt{=x}\delta\rangle, t_2[\langle \beta, \texttt{x}\gamma\rangle]) = \begin{array}{ccc} & < & \\ \langle \beta\text{-}\alpha, \delta\rangle & & t_2[\langle \epsilon, \gamma\rangle] \end{array}$$

The operation **move** applies to a single tree $t[\langle \alpha, \texttt{+y}\delta\rangle]$ only if there is *exactly one* leaf ℓ in t with matching first feature $-\texttt{y}$ or $\ominus\texttt{y}$. This is a radical version of the shortest move constraint [4], and will be called the SMC – it requires that an expression move to the first possible landing site. If there is competition for that landing site, the derivation crashes (because the losing expression will have to make a longer movement than absolutely necessary). If it applies, **move** moves the maximal projection of ℓ to a newly created specifier position in t, and deletes both licensing features. To make this precise, let $t\{t_1 \mapsto t_2\}$ denote the result of replacing all subtrees t_1 in t with t_2, for any tree t, and let ℓ_t^M denote the maximal projection of ℓ in t, for any leaf ℓ.

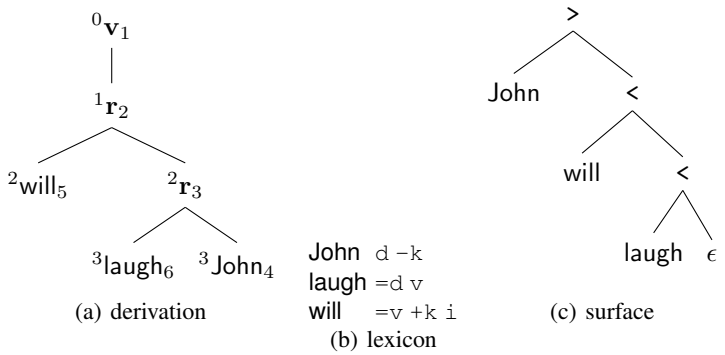
$$\textbf{move}(t[\langle \alpha, \texttt{+y}\delta\rangle]) = \begin{array}{ccc} & > & \\ t'[\langle \beta, \gamma\rangle] & & t[\langle \alpha, \delta\rangle]\{t' \mapsto \langle \epsilon, \epsilon\rangle\} \end{array}$$

(where $t' = \langle \beta, -\texttt{y}\gamma\rangle_t^M$)

A derivation tree is an element of the term language over the ranked alphabet $A_0 \cup A_1 \cup A_2$, where $A_0 = Lex$ is the set of nullary symbols, $A_1 = \{\mathbf{v}\}$ is the set of unary symbols, and $A_2 = \{\mathbf{r}\}$ the set of binary symbols. As a consequence of the translation of minimalist grammars into multiple context free grammars [27, 10], the set of derivation trees in a minimalist grammar of an expression with unchecked feature string γ at the root and no features anywhere else is regular.

As an example, the lexical item *John* in figure 3(b) has the feature sequence 'd -k', which indicates that it must first be the second argument to the merge operation (where the first argument has a matching =d feature), and then it will, as part of a larger expression whose head has a +k feature, be targeted by the move operation. Similarly,

the lexical item *laugh* with feature sequence '=d v' indicates that it must first be the first argument to the merge operation (where the second argument has a matching d feature), and then it may be the second argument to the merge operation (where the first argument has matching =v feature). The sequence of rule applications used in the construction of



(a) derivation (b) lexicon (c) surface

Fig. 1. Structures for "John will laugh."

a sentence can itself be viewed as a tree, as in figure 1(a), which describes the derivation of the surface structure in 1(c). In figure 1(a), internal nodes are labeled either **v** (for move) or **r** (for merge), and leaves are labeled with lexical items (we have suppressed the features for reasons of space). The internal node immediately dominating the leaves *laugh* and *John* is labeled **r**, which indicates that the lexical items *laugh* and *John* were merged together. The parent of this node is also labeled **r**, and indicates that the lexical item *will* was merged together with the result of merging together *laugh* and *John*. The derived tree in figure 1(c) is therefore the result of applying the move rule (**v**) to the result of merging *will* together with the result of merging together *laugh* and *John*.

The nodes of the derivation tree in figure 1(a) are superscripted (on the left) and subscripted (on the right). Derivation trees marked up in this way are a very condensed yet *complete* representation of a parse of a sentence. Nodes represent parser items, the superscript indicates at which parsing step that node is put into the parser's stack and the subscript indicates at which parsing step it is removed from the parser's stack.[4] The order in which predicted items are expanded is determined by their order in the surface tree, which is only computed implicitly by the parser. Compare the order of terminals in the surface tree 1(c) with the order the leaves of the derivation tree are expanded. Table 1 reconstructs the parser stack at each step from the marked-up derivation tree in 1(a). The underlining in table 1 indicates which item is operated on in the subsequent step. Items are represented in the table as terms, where **S** is the initial item, and for α an item, $\mathbf{v}(\alpha)$, $\mathbf{r}(\alpha)_1$, and $\mathbf{r}(\alpha)_2$ is the result of applying an unmove rule to α, the first element of the pair resulting from applying an unmerge rule to α, and the second element of

[4] A more precise characterization of the relation between a marked up derivation tree and the sequence of parser states in a successful top down parse of that derivation is that an item corresponding to a node $^i\alpha_j$ is in the stack at time t iff $i \leq t < j$.

0	—	$\{\underline{\mathbf{S}}\}$
1	—	$\{\underline{\mathbf{v}(\mathbf{S})}\}$
2	—	$\{\underline{\mathbf{r}(\mathbf{v}(\mathbf{S}))_1},\ \underline{\mathbf{r}(\mathbf{v}(\mathbf{S}))_2}\}$
3	—	$\{\underline{\mathbf{r}(\mathbf{v}(\mathbf{S}))_1},\ \mathbf{r}(\mathbf{r}(\mathbf{v}(\mathbf{S}))_2)_1,\ \underline{\mathbf{r}(\mathbf{r}(\mathbf{v}(\mathbf{S}))_2)_2}\}$
4	—	$\{\underline{\mathbf{r}(\mathbf{v}(\mathbf{S}))_1},\ \mathbf{r}(\mathbf{r}(\mathbf{v}(\mathbf{S}))_2)_1\}$
5	—	$\{\underline{\mathbf{r}(\mathbf{r}(\mathbf{v}(\mathbf{S}))_2)_1}\}$
6	—	$\emptyset$

Table 1. The sequence of parser configurations corresponding to figure 1(a)

the same, respectively. (See §4.1 for more details.) Observe that in, for example, step 2, all and only items corresponding to nodes in figure 1(a) with a superscript less than or equal to 2 and a subscript greater than 2 appear in the stack – at parsing step two, there are two items in the parser state; one corresponding to a prediction of the lexical item *will* ($\mathbf{r}(\mathbf{v}(\mathbf{S}))_1$), and one to a prediction of a VP-like constituent (($\mathbf{r}(\mathbf{v}(\mathbf{S}))_2$ – the node labelled ${}^2\mathbf{r}_3$). In the next step, instead of scanning a word from the input as would a context-free parser, the prediction ${}^2\mathbf{r}_3$ is expanded. The prediction of *will* remains in the parser state until the fifth step.

Because the marked up derivation tree concisely encodes the entire parse history of an expression, we use it exclusively in the remainder of this paper.

4.1 Parsing

A top down minimalist parser explores a search space defined by inverting the operations of merge and move (and called in [10] *unmerge* and *unmove*). As mentioned in section 2, we assume that the parser is equipped with a perfect oracle, and will ignore the bookkeeping necessary to perform backtracking.[5] This assumption amounts to claiming that the asymmetries in acceptability judgements in the constructions we examine here are not due to (local) ambiguities; either because there are none, as we assume here, or because all sentences involved have roughtly the same amount.

As in the case of context-free parsing, a minimalist parser item corresponds to a node in a derivation tree, and a minimalist parser state is a sequence of minimalist parser items. Just like with context-free parsing, a (top-down) parser state represents the set of derivation tree contexts with the same open positions; the parser items it contains correspond to the categories of the open positions, and their order the order in which these open positions might correspond to the input string. Differences between them, however, stem from the following difference between the grammar formalisms: the language of a given nonterminal in a minimalist grammar consists of *tuples* of derived objects [38], as opposed to single derived object as in the case of context-free grammars. Accordingly, the minimalist parser's 'stack' needs to take the form of a priority queue. As shown by [24], an ordering on derivation tree nodes reflecting the corresponding node in the surface tree can be computed efficiently by the parser online

[5] This decision is motivated also by the fact that search strategy and oracle are highly underdetermined by the search space, and that we do not know how to select among the alternatives in a principled way.

(a finite copying transduction relates the two [19, 29]). The order on queue elements is given by this relation. In some sense, the crucial difference between minimalist parsing and context-free parsing is that the ordering relation between nodes in the minimalist derivation tree (defined as per the above in terms of the surface string position of the leftmost component) is *not* inherited through dominance, whereas that in the context-free derivation tree *is*.[6] This allows the priority queue for context-free parsing to behave as a simple stack.

We do not present the full set of parser rules here, for reasons of space and simplicity (we leave out the rules for head movement, as this complicates things even more; see footnote 8); see [24] and [36] for more optimized versions hereof.[7] The parser rules are presented as (upside down) inference rules, such as the below.

$$\frac{P}{R_1 \dots R_n}$$

This rule is to be understood as saying that the items $R_1, \dots, R_n$ are derivable in one step from item P. Given a stack (or some similar data structure) whose top item is P, applying this rule removes P from the stack, and adds $R_1 \dots R_n$. If $n < 1$, this means that P is simply removed from the stack.

A parser item takes the form $\langle \gamma_0, \alpha_0; \gamma_1, \alpha_1; \dots; \gamma_n, \alpha_n \rangle$, where for $0 \leq i \leq n$, α_i is the feature sequence of the i^{th} moving expression, and γ_i is the gorn address of its position in the derived tree.[8] The parser items are totally ordered, where $p < p'$ iff the leftmost γ_i in p is to the left of the leftmost γ'_j in p'.[9]

The **scan** rule is given below. Here, a parser item is assumed to correspond to a lexical item, and is removed from the stack.

$$\frac{\langle \gamma, \alpha \rangle}{} \quad \text{where } u \text{ is the next word and } \langle \mathsf{u}, \alpha \rangle \in Lex \qquad \textbf{(scan)}$$

For conciseness, we write an item $\langle \gamma_0, \alpha_0; \gamma_1, \alpha_1; \dots; \gamma_n, \alpha_n \rangle$ as $\langle \gamma_0, \alpha_0; A \rangle$. We use the notation $\langle \gamma_0, \alpha_0; A[\phi] \rangle$ to indicate that A contains ϕ. If $A[\psi]$ occurs in the antecedent of a rule, then $A[\phi]$ in the consequent indicates that ψ has been replaced by ϕ in A. We write $A[-]$ to indicate that ψ has been removed. We write $A = B \oplus C$ to indicate that A can be partitioned into B and C.

[6] An ordering ($<$) over derivation tree nodes is inherited through dominance in this sense just in case for any two nodes a and b, $a < b$ implies that, for any children c_a and c_b of a and b, $c_a < c_b$.

[7] In particular, we should restrict the feature sequences in predicted items to lexical feature suffixes.

[8] To incorporate head movement, we need to decompose γ_0 into the triple $\gamma_0^0, \gamma_0^1, \gamma_0^2$; see e.g. [16].

[9] This is a total order because the gorn addresses assigned to items in the stack are unique, corresponding as they do to positions in the derived tree.

unmerge1s

$$\frac{\langle \gamma, \alpha; A \oplus B \rangle}{\langle \gamma 1, \texttt{x=}\alpha; A \rangle \quad \langle \gamma 0, \texttt{x}; B \rangle}$$

unmerge2s

$$\frac{\langle \gamma_0, \alpha_0; A \oplus B[\gamma, \alpha] \rangle}{\langle \gamma_0 1, \texttt{x=}\alpha; A \rangle \quad \langle \gamma, \texttt{x}\alpha; B[-] \rangle}$$

unmerge1c

$$\frac{\langle \gamma, \alpha; A \oplus B \rangle}{\langle \gamma 0, \texttt{=x}\alpha; A \rangle \quad \langle \gamma 1, \texttt{x}; B \rangle}$$

unmerge2c

$$\frac{\langle \gamma_0, \alpha_0; A \oplus B[\gamma, \alpha] \rangle}{\langle \gamma_0 0, \texttt{=x}\alpha; A \rangle \quad \langle \gamma, \texttt{x}\alpha; B[-] \rangle}$$

In the case of the **unmerge** rules, the moving components (described above as $A \oplus B$) must be split among the two newly predicted items. Illustrative are the gorn addresses of the predicted items. In the case of **unmerge1s** we are assuming that the second argument to the **merge** operation was a specifier (merged on the left – x=) and was pronounced there (i.e. it did not later move as it has no further features) – we can therefore conclude that the item representing the second argument to **merge** is the left sister of the item representing the first argument in the derived surface tree.

unmove1

$$\frac{\langle \gamma_0, \alpha_0; A[-] \rangle}{\langle \gamma_0 1, \texttt{+x}\alpha; A[\gamma_0 0, \texttt{-x}] \rangle}$$

unmove2

$$\frac{\langle \gamma_0, \alpha_0; A[\gamma, \alpha] \rangle}{\langle \gamma_0 1, \texttt{+x}\alpha; A[\gamma, \texttt{-x}\alpha] \rangle}$$

In **unmove2**, which corresponds to an application of **move** where the moving expression has not moved its last (i.e. it has movement features left over), the predicted item only updates the gorn address of the head, as the moving item's surface position is not changed by this movement. This contrasts with **unmove1**, which corresponds to an application of **move** where the moving expression moves to its final resting place. Here we know that the moving expression is the left daughter of the head of the popped item, and the head of the predicted item is the right daughter (as movement is to the left).

5 Modeling

Here we report the results of our complexity measures on sentences of the kind in sections 3.1 and 3.2. We begin (in §5.1) with an analysis of verb clusters, and show that a simple but linguistically motivated analysis of Dutch and German predicts maximum tenure differences which line up with the behavioural data. Then (in §5.2) we consider the stark constrast in English between center embedded and peripherally embedded structures. We provide two syntactic analyses of this phenomenon, which differ from one another only on their analysis of verbal inflection. Somewhat surprisingly, this matters, and highlights the degree to which tenure is dependent upon the particulars of a syntactic analysis.

5.1 Verb Clusters

We assume a verb-raising analysis, depicted in figure 2.[10] On this analysis, verbal complexes in both German and in Dutch have the same deep structure 2(a), one in which verbs and their objects are in a strict sisterhood relation [43]. The different surface word orders arise as a consequence of verbal head movement up and to the left (as shown in 2(c) in the case of German) or to the right (as shown in 3(c) in the case of Dutch). The only difference between the two grammar fragments is that the verb cluster is formed via *leftward* head movement in German, and via *rightward* head movement in Dutch.

The lexical items in both figures, which are identical but for the direction of head movement (indicated by means of the hyphen attached to the lexeme), are schematic representations of either nominal phrases ('DPs'), sequence initiating verbs (Vi), serial verbs (V) and inflectional heads (I).

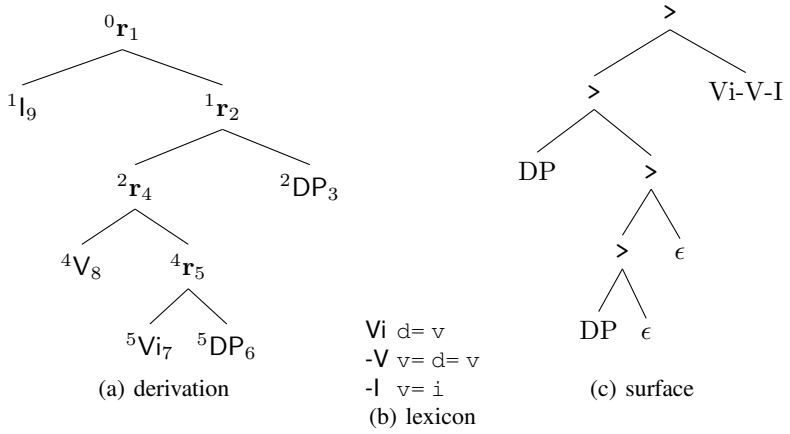


Fig. 2. Structures for German

	Max Tenure	
Embeddings	**German**	**Dutch**
1	8	6
2	12	9
3	16	12
4	20	15

Figures 2 and 3 correspond to the first row in the table above. In the German derivation 2(a) and surface structure 2(c), the maximal tenure is had by the parser item corresponding to the node labelled $^1\mathsf{I}_9$ in the figure, which is predicted at the first step, and which is removed from the queue only at the ninth and last step. The reason why this item is predicted already in the first step is because it is a child of the root/starting item. It is

[10] This particular analysis is a variation of [31].

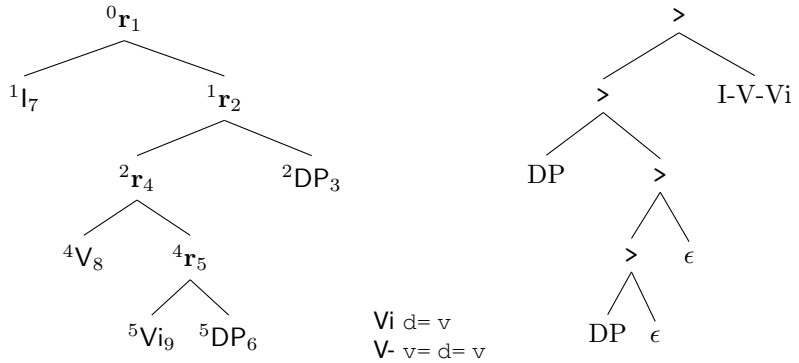


Fig. 3. Structures for Dutch

removed from the queue so late because every other parser state contains an item which can be expanded into items which correspond to words in the input which come before this one.

The Dutch deep structure is traversed identically to the German one up until step 7, where, instead of operating on the prediction for a *Vi*, the prediction of an inflectional element is operated on.

5.2 English Center and Right Embeddings

We adopt a promotion-style analysis of relative clauses, according to which the relative clause head is an argument of the embedded verb, and then moves to a clause-peripheral position [15, 8, 17]. We report the results of applying our complexity metric to two grammar fragments which differ in their analyses of verbal inflection. One (4(c)) relies

-s	=v +k s	that	=s +w n	s	=v +z +k s
laugh	=d v	ϵ	=n d −k −w	laugh	=d v −z
-ϵ	=V +k d= v	the	=n d −k	ϵ	=V +k d= v
praise	=d V	boy	n	praise	=d V −z
	(a) head		(b) shared		(c) phrasal

Fig. 4. Lexica

on phrasal movement which conspires to position the verb before the inflectional ending as suggested by [23]. The other (4(a)) uses head movement to build a complex head consisting of a verb and its inflections, an analysis which has its roots in [3].

	Head Mvt		Phrasal Mvt	
Depth	**Right**	**Center**	**Right**	**Center**
1	11	23	18	24
2	11	40	36	42
3	11	57	54	60
4	11	74	72	78
5	11	91	90	96
6	11	108	108	114

Under the phrasal movement analyis of inflection, although sentences with right branching embedding have a lower maximal tenure than do those with center embeddings, they have the same rate of growth – in other words, there is no bound on maximal tenure which will correctly rule out (as unacceptable) center embedded sentences of nesting degree greater than (say) 3, but allow peripheral embedding to (much) higher degrees.

Under the head movement analysis of inflection (4(a)), however, sentences with right branching embedding (2) have a lower maximal tenure than those with center embedding (1). Indeed, the maximal tenure of right branching sentences remains constant up to 6 embeddings, whereas the maximal tenure of center embedded structures continues to increase (by seventeen steps) with each additional embedding. Derivation trees for center and peripherally embedded sentences under the head movement analysis of inflection are given in figures 5(a) and 6(a) respectively. In both cases, it is the matrix clause inflectional head (the present tense suffix *-s*) which is the parser item with the maximal tenure. The crucial aspect of the minimalist analysis is that the inflectional head is predicted very early, due to the fact that inflection is assumed to merge with the verb only after all its arguments have been merged with it.

6 General Discussion

Our proposal is that the maximal tenure of an item on the stack reflects a memory requirement that burdens human comprehenders. This is related to different aspects of previous work in psycholinguistics. Since it views a memory cell being occupied as opposed to unoccupied, our proposal can be viewed as a generalization of the HOLD hypothesis [13, 42]. One could also view long tenure as an approximation to working memory decay [22].

We have focussed here on off-line difficulty measures. However, various on-line notions of stack tenure are easily derivable, such as maximal tenure up to a particular point in the sentence, or average tenure of stack items. In particular, we derive interesting predictions for on-line judgements in the Dutch versus German data. In figure 9, we report for each of the last six parser steps (steps 13 to 18) of a parse of a sentence with four DPs the maximal tenure of a current stack item,, the average tenure of items on the stack , and the sum tenure of all items on the stack. Note that the steps taken by the parser on the Dutch and German sentences are identical up to and including step 13, at which point the verbal cluster begins to be parsed.

step	max	average	sum
13	12	6.4	32
14	13	8.5	34
15	14	11	33
16	15	13.5	27
17	16	16	16
18	(empty)		

(a) German

step	max	average	sum
13	12	6.4	32
14	10	6	24
15	8	5.7	17
16	6	5.5	11
17	6	6	6
18	(empty)		

(b) Dutch

$$\max(\mathsf{stack}) = \mathsf{max}_{x \in \mathsf{stack}}\, \mathsf{tenure}(x)$$
$$\text{average}(\mathsf{stack}) = \text{sum}(\mathsf{stack}) / \, |\mathsf{stack}|$$
$$\text{sum}(\mathsf{stack}) = \Sigma_{x \in \mathsf{stack}} \mathsf{tenure}(x)$$

Fig. 5. The parser on the German/Dutch sentence with 3 embeddings

6.1 Difficulties with maximal tenure

Although simple and well-defined, the notion of maximal tenure (or incremental versions thereof, cf. §7) in minimalist grammars is difficult to relate to geometric properties of a derivation. (As, for example, it is possible to relate memory burden in left-corner parsing to the geometric property of center embedding in the context-free parse tree.) Still, a high tenure will obtain whenever an *unmerge* rule introduces derivational sisters, which are seperated on the surface by a large number of derived tree leaves. In particular, as pointed out by a reviewer, the top-down minimalist parser should assign high tenure to left-branching structures (as in 5), just as would a top-down context-free parser, which does not seem to accurately reflect the behavioural data.

(5) John's neighbor's dentist's uncle laughed.

Additionally, the more words which intervene between a *wh*-word and its extraction site, the higher the tenure. Sentence 7 has a higher tenure than 6, despite the fact that the paths between the *wh*-words and their extraction sites are of the same length.

(6) Who did the boy kiss.
(7) Who did the very tall boy kiss.

To the extent that this is not reflected in the behavioural data, one potential (but radical) solution could be to change the way tenure is measured.

There are three currently existing proposals in this regard. First, the Dependency Locality Theory [6] can be thought of in the present terms as measuring not the number of steps taken by the parser while a given item is on the stack, but the number of discourse referents introduced. This change would then equate the tenures of 6 and 7, but would still predict a high tenure score for structures like example 5. In the computational literature, Joshi [12] (see also [33]) presents an account of the observed processing contrast between German and Dutch verbal complexes in the context of tree adjoining grammar (TAG). Using the fact that a TAG can be compiled into an equivalent (embedded) push down automaton (EPDA), he notes that the total amount of time the EPDA for

the Dutch sentences stores any symbol on its stack is less than that of the EPDA for the German sentences.[11] The complexity measure of 'sum total tenure of stack items' is equivalent to the sum of the sizes of the stack after each step (the sum of the number of symbols on stack one plus the number of symbols on stack two plus ...). However, the notion of a step is taken in [12] to be a scan of an overt word, and intermediate steps are not taken into account. In [33], a step is indeed taken to be a single operation of the automaton, but symbols corresponding to empty nodes in an elementary tree are crucially not counted toward stack size.

7 Conclusion

We have shown that longstanding and influential psycholinguistic ideas about memory resources can be connected with specific and explicit syntactic analyses in rigorous ways. The results of section 5.2 show that the syntactic analysis can indeed play a significant role in the memory requirements of parsing.

The notion of tenure, while useful (at least to a first approximation), is not able to account for all aspects of psycholinguistic data. In particular, we have idealized non-determinism in the parsing process away, while complexity measures which focus on the resolution of non-determinism, such as entropy reduction [8] or surprisal [7], have been demonstrated to have explanatory value [21, 41]. Furthermore, although tenure is a measure related to memory burden, we have made very weak assumptions about the nature of memory — incorporating psychological insights into the nature and limitations of human memory [22] may allow a reductive explanation of our tenure measure, or at least for a more refined and nuanced theory.

Future work will investigate how best to relate tenure to online data, how to integrate various indepenent notions of psycholinguistic 'difficulty', and how to most appropriately account for examples like those presented in §6.1.

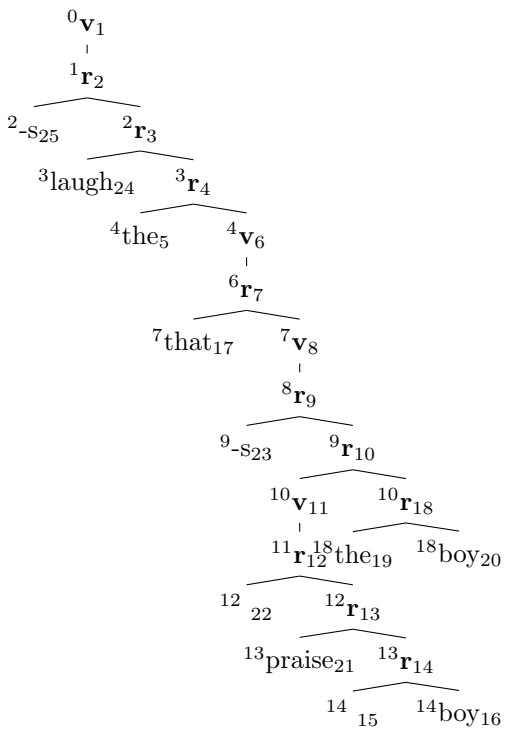
References

1. Abney, S., Johnson, M.: Memory requirements and local ambiguities of parsing strategies. Journal of Psycholinguistic Research 20(3), 233–249 (1991)
2. Bach, E., Brown, C., Marslen-Wilson, W.: Crossed and nested dependencies in German and Dutch: A psycholinguistic study. Language and Cognitive Processes 1(4), 249–262 (1986)
3. Chomsky, N.: Syntactic Structures. Mouton, The Hague (1957)
4. Chomsky, N.: The Minimalist Program. MIT Press, Cambridge, Massachusetts (1995)
5. Crocker, M.W., Brants, T.: Wide-coverage probabilistic sentence processing. Journal of Psycholinguistic Research 29(6), 647–669 (2000)
6. Gibson, E.: The dependency locality theory: A distance-based theory of linguistic complexity. In: Miyashita, Y., Marantz, A., O'Neil, W. (eds.) Image, Language, Brain, pp. 95–126. MIT Press, Cambridge, Massachusetts (2000)
7. Hale, J.: A Probabilistic Earley Parser as a Psycholinguistic Model. In: Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics (2001)

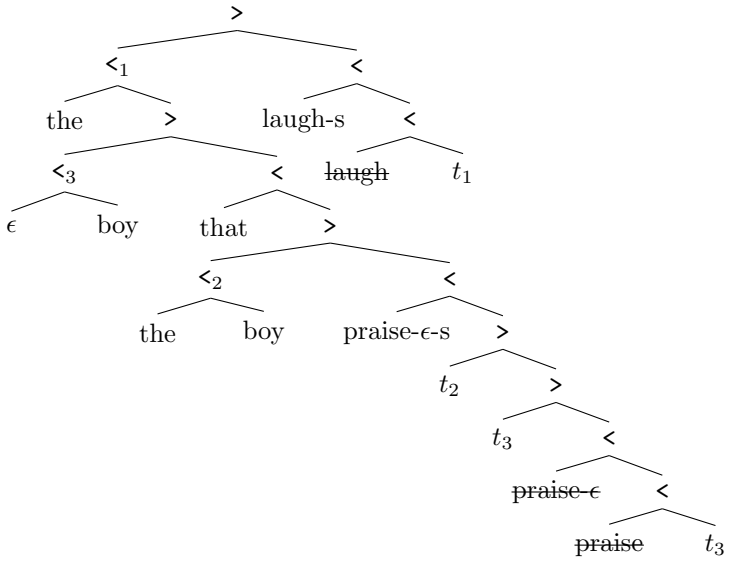
[11] He also observes that the maximal stack size is lower for the Dutch EPDA than for the German one.

8. Hale, J.T.: Grammar, Uncertainty and Sentence Processing. Ph.D. thesis, The Johns Hopkins University (2003)
9. Hale, J.T.: Uncertainty about the rest of the sentence. Cognitive Science 30, 643–672 (2006)
10. Harkema, H.: Parsing Minimalist Languages. Ph.D. thesis, University of California, Los Angeles (2001)
11. Johnson-Laird, P.N.: Mental Models. Cambridge University Press (1983)
12. Joshi, A.K.: Processing crossed and nested dependencies: An automation perpsective on the psycholinguistic results. Language and Cognitive Processes 5(1), 1–27 (1990)
13. Kaplan, R.M.: Transient Processing Load in Relative Clauses. Ph.D. thesis, Harvard (1975)
14. Kay, M.: Algorithm schemata and data structures in syntactic processing. In: Grosz, B.J., Jones, K.S., Webber, B.L. (eds.) Readings in Natural Language Processing. Morgan Kaufman (1986)
15. Kayne, R.: The Antisymmetry of Syntax. MIT Press, Cambridge, Massachusetts (1994)
16. Kobele, G.M.: Formalizing mirror theory. Grammars 5(3), 177–221 (2002)
17. Kobele, G.M.: Generating Copies: An investigation into structural identity in language and grammar. Ph.D. thesis, University of California, Los Angeles (2006)
18. Kobele, G.M.: Importing montagovian dynamics into minimalism. In: Béchet, D., Dikovsky, A. (eds.) Logical Aspects of Computational Linguistics. Lecture Notes in Computer Science, vol. 7351, pp. 103–118. Springer, Berlin (2012)
19. Kobele, G.M., Retoré, C., Salvati, S.: An automata theoretic approach to minimalism. In: Rogers, J., Kepser, S. (eds.) Proceedings of the Workshop Model-Theoretic Syntax at 10; ESSLLI '07. Dublin (2007)
20. Kowalski, R.: Algorithm = logic + control. Communications of the ACM 22(7), 424–436 (1979)
21. Levy, R.: Expectation-based syntactic comprehension. Cognition 106, 1126–1177 (2008)
22. Lewis, R.L., Vasishth, S.: An activation-based model of sentence processing as skilled memory retrieval. Cognitive Science 29, 375–419 (2005)
23. Mahajan, A.: Word order and (remnant) VP movement. In: Karimi, S. (ed.) Word Order and Scrambling, chap. 10. Blackwell (2003)
24. Mainguy, T.: A probabilistic top-down parser for minimalist grammars. CoRR abs/1010.1826 (2010)
25. Marr, D.: Vision. W. H. Freeman and Company, New York (1982)
26. Marslen-Wilson, W.: Linguistic structure and speech shadowing at very short latencies. Nature 244, 522–523 (1973)
27. Michaelis, J.: On Formal Properties of Minimalist Grammars. Ph.D. thesis, Universität Potsdam (2001)
28. Miller, G.A., Chomsky, N.: Finitary models of language users. In: Luce, R.D., Bush, R.R., Galanter, E. (eds.) Handbook of mathematical psychology, chap. 13, pp. 419–491. John Wiley, New York (1963)
29. Mönnich, U.: Minimalist syntax, multiple regular tree grammars and direction preserving tree transductions. In: Rogers, J., Kepser, S. (eds.) Proceedings of the Workshop Model-Theoretic Syntax at 10; ESSLLI '07. Dublin (2007)
30. Montague, R.: The proper treatment of quantification in ordinary English. In: Hintikka, J., Moravcsik, J., Suppes, P. (eds.) Approaches to Natural Language, pp. 221–242. D. Reidel, Dordrecht (1973)
31. Morawietz, F.: Two-Step Approaches to Natural Language Formalisms, Studies in Generative Grammar, vol. 64. Mouton de Gruyter (2003)
32. Rambow, O., Joshi, A.K.: A processing model for free word order languages. In: Clifton, C., Frazier, L., Rayner, K. (eds.) Perspectives on sentence processing, pp. 267–301. Lawrence Erlbaum (1994)

33. Rambow, O., Joshi, A.K.: A processing model for free word order languages. Tech. Rep. IRCS-95-13, University of Pennsylvania (1995)
34. Resnik, P.: Left-corner parsing and psychological plausibility. In: Proceedings of the Fourteenth International Conference on Computational Linguistics. Nantes, France (1992)
35. Shieber, S.M.: Evidence against the context-freeness of natural language. Linguistics and Philosophy 8, 333–343 (1985)
36. Stabler, E.: Top-down recognizers for mcfgs and mgs. In: Proceedings of CMCL (2011)
37. Stabler, E.P.: Derivational minimalism. In: Retoré, C. (ed.) Logical Aspects of Computational Linguistics, Lecture Notes in Computer Science, vol. 1328, pp. 68–95. Springer-Verlag, Berlin (1997)
38. Stabler, E.P., Keenan, E.L.: Structural similarity within and among languages. Theoretical Computer Science 293, 345–363 (2003)
39. Tanenhaus, M., Spivey-Knowlton, M., Eberhard, K., Sedivy, J.: Integration of visual and linguistic information in spoken language comprehension. Science 268, 1632–1634 (1995)
40. VanWagenen, S., Brennan, J., Stabler, E.P.: Evaluating parsing strategies in sentence processing. Poster presented at CUNY 2011 (2011)
41. Vasishth, S., Drenhaus, H.: Locality in German. Dialogue and Discourse 1(2), 59–82 (2011)
42. Wanner, E., Maratsos, M.: An ATN approach to comprehension. In: Halle, M., Bresnan, J., Miller, G.A. (eds.) Linguistic Theory and Psychological Reality, chap. 3, pp. 119–161. MIT Press, Cambridge, Massachusetts (1978)
43. Wurmbrand, S.: How complex are complex predicates. Syntax 10(3), 243–288 (2007)

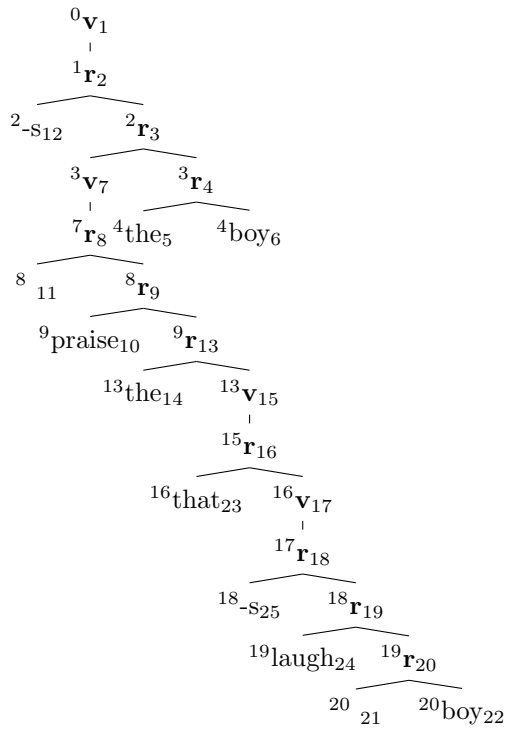


(a) derivation tree

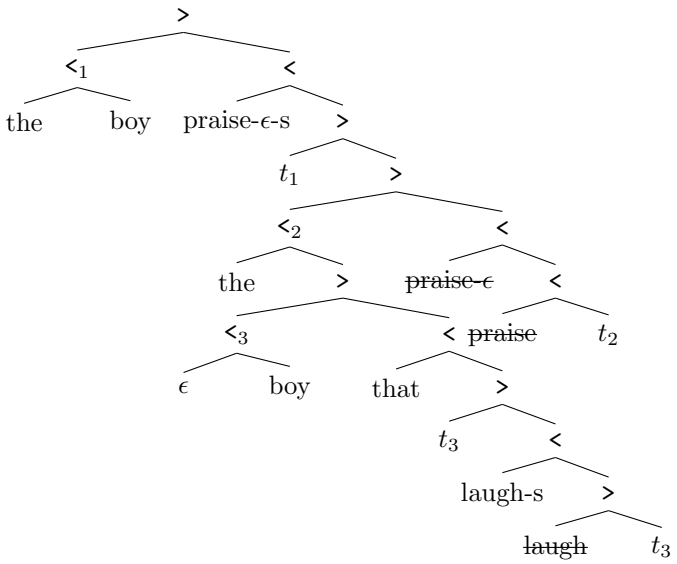


(b) derived tree

Fig. 6. Center Embedding with Head Movement

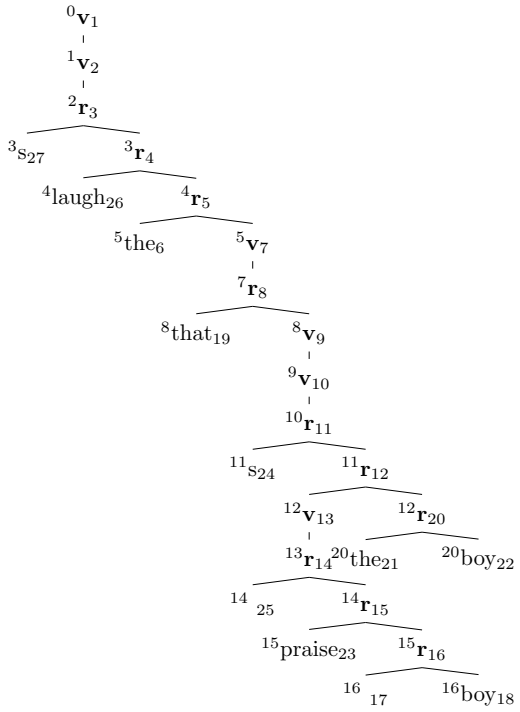


(a) derivation tree

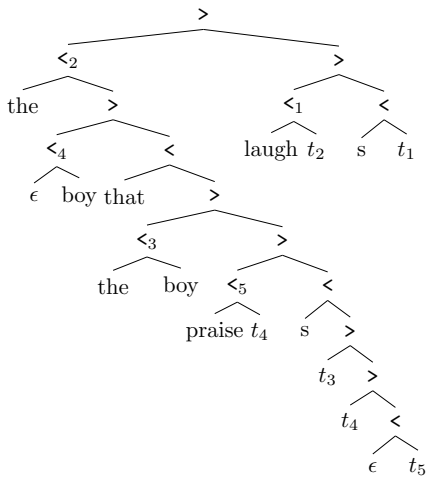


(b) derived tree

Fig. 7. Peripheral Embedding with Head Movement

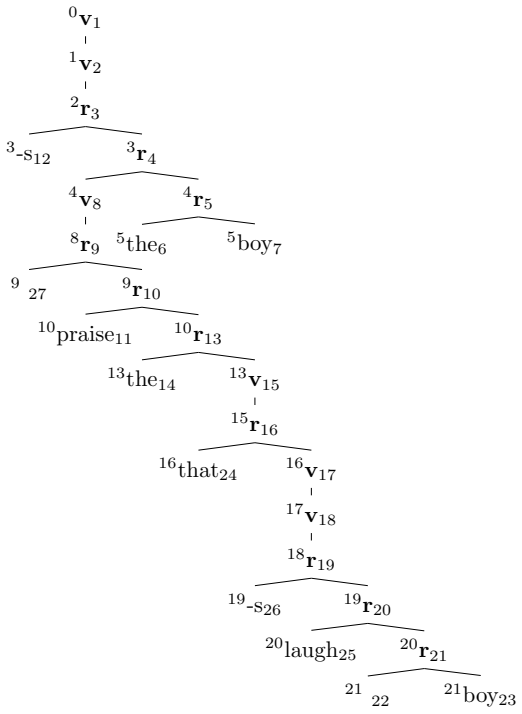


(a) derivation tree

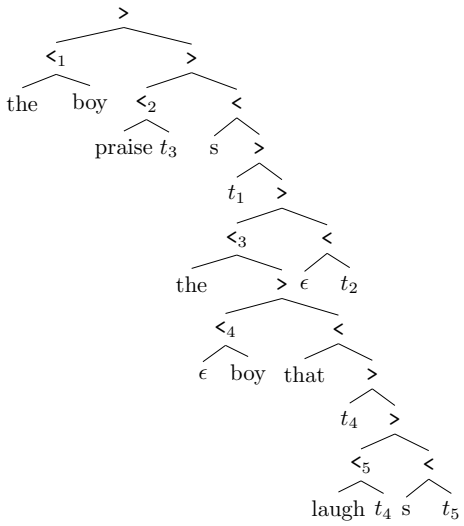


(b) derived tree

Fig. 8. Center Embedding with Phrasal Movement



(a) derivation tree



(b) derived tree

Fig. 9. Peripheral Embedding with Phrasal Movement