

Top-down recognizers for MCFGs and MGs

Edward P. Stabler

stabler@ucla.edu

Abstract

This paper defines a normal form for MCFGs that includes strongly equivalent representations of many MG variants, and presents an incremental priority-queue-based TD recognizer for these MCFGs. After introducing MGs with overt phrasal movement, head movement and simple adjunction are added without change in the recognizer. The MG representation can be used directly, so that even rather sophisticated analyses of properly non-CF languages can be defined very succinctly. As with the similar stack-based CF-methods, finite memory suffices for the recognition of infinite languages, and a fully connected left context for probabilistic analysis is available at every point.

1 Introduction

In the years after Joshi (1985) proposed that human languages are weakly and strongly “mildly context sensitive” (MCS), it was discovered that many independently proposed grammar formalisms define exactly the same MCS languages. The languages defined by Joshi’s tree adjoining grammars (TAGs) are exactly the same as those defined by a version of Steedman’s combinatory categorial grammars, and the same as those defined by head wrapping grammars (Vijay-Shanker and Weir, 1994). A slightly larger class of languages is defined by another variant of TAGs (set-local multicomponent), by a version of Pollard’s generalized phrase structure grammars called multiple context free grammars (MCFGs), and by a wide range of minimalist

grammar (MG) formalizations of Chomskian syntax (Seki et al., 1991; Michaelis, 1998; Michaelis, 2001b; Harkema, 2001a; Stabler, 2011). These remarkable convergences provide evidence from across grammatical traditions that something like these MCS proposals may be approximately right, and so it is natural to consider psychological models that fit with these proposals. With a range of performance models for a range of MCS grammars, it becomes possible to explore how grammatical dependencies interact with other factors in the conditioning of human linguistic performance.

For context free grammars (CFGs), perhaps the simplest parsing model is top-down: beginning with the prediction of a sentence, rules are applied to the leftmost predicted category until a terminal element is reached, which is then checked against the input. This parsing method is of interest in psychological modeling not only because it uses the grammar in a very transparent way, but also it is because it is predictive in a way that may be similar to human parsing. At every point in analyzing a sentence from left to right, the structure that has been constructed is fully connected: grammatical relationships among the elements that have been heard have been guessed, and there are no pieces of structure which have not been integrated. Consequently, this structure can be interpreted by a standard compositional semantics and may be appropriate for “incremental” models of sentence interpretation (cf. Haddock, 1989; Chambers et al., 2004; Shen and Joshi, 2005; Altmann and Mirković, 2009; Demberg and Keller, 2009; Kato and Matsubara, 2009; Schuler, 2010). And like human parsing, when used with

backtracking or a beam search, TD memory demands need not continually increase with sentence length: a fixed bound on stack depth and on back-track or beam depth suffices for infinitely many sentences. Furthermore, TD parsing provides explicit, relevant “left contexts” for probabilistic conditioning (Roark and Johnson, 1999; Roark, 2001; Roark, 2004). But it has not been clear until recently how to apply this method to Chomskian syntax or any of the other MCS grammar formalisms. There have been some proposals along these lines, but they have either been unnecessarily complex or applicable to only a restricted range of grammatical proposals (Chesi, 2007; Mainguy, 2010).

This paper extends TD parsing to minimalist context free grammars (MCFGs) in a certain normal form and presents minimalist grammars (MGs) as a succinct representation for some of those MCFGs. With this extension, the TD parsing method handles an infinite range of MCFGs that encompasses, strongly and weakly, an infinite range of (many variants of) MGs in a very transparent and direct way. The parsing method can be defined in complete detail very easily, and, abstracting away from limitations of time and memory, it is provably sound and complete for all those grammars.

The TD recognizer for MCFGs is presented in §4, generalizing and adapting ideas from earlier work (Mainguy, 2010; Villemonte de la Clergerie, 2002). Instead of using a stack memory, this recognizer uses a “priority queue,” which just means that we can access all the elements in memory, sorting them into left-to-right order. Then it is easy to observe: (§3.2) while the reference to MCFG is useful for understanding the recognizer, an MG representation can be used directly without explicitly computing out its MCFG equivalent; (§5.1) the extensions for head movement and simple adjunction allow the recognizer of §4 to apply without change; (§5.2) like its stack-based CF counterpart, the MG recognizer requires only finite memory to recognize certain infinite subsets of languages – that is, memory demands do not always strictly increase with sentence length; and (§5.3) the TD recognizer provides, at every point in processing the input, a fully connected left context for interpretation and probabilistic conditioning, unlike LC and other familiar methods. Since a very wide range of grammatical proposals can be

expressed in this formalism and parsed transparently by this method, it is straightforward to compute fully explicit and syntactically sophisticated parses of the sorts of sentences used in psycholinguistic studies.

2 MCFGs

MCFGs are first defined by Seki et al. (1991), but here it will be convenient to represent MCFGs in a Prolog-like Horn clause notation, as in Kanazawa (2009). In this notation, the familiar context free rule for sentences would be written

$$S(x_{0_1}x_{1_1}) :- NP(x_{0_1}), \\ VP(x_{1_1}).$$

Reading $:-$ as “if”, this formula says that a string formed by concatenating any string x_{0_1} with string x_{1_1} is an S, if x_{0_1} is an NP, and x_{1_1} is a VP. We number the variables on the right side in such a way as to indicate that each variable that appears on the right side of any rule appears exactly once on the right and once on the left. Lexical rules like

$$NP(\text{Mary}) \\ VP(\text{sings}),$$

have empty “right sides” and no variables in this notation.

MCFGs allow categories to have multiple string arguments, so that, for example, a VP with a wh-phrase that is moving to another position could be represented with two string arguments, one of which holds the moving element. In general, each MCFG rule for building an instance of category A from categories $B_0 \dots B_n$ ($n \geq 0$) has the form,

$$A(t_1, \dots, t_{d(A)}) :- B_0(x_{0_1}, \dots, x_{0_{d(B_0)}}), \\ \dots, \\ B_n(x_{n_1}, \dots, x_{n_{d(B_n)}}),$$

where each t_i is a term (i.e. a sequence) over the (finite nonempty) vocabulary Σ and the variables that appear on the right; no variable on the right occurs more than once on the left (no copying); and the designated ‘start’ category S has ‘arity’ or ‘dimension’ $d(S) = 1$. For any such grammar, the language $L(G)$ is the set of strings $s \in \Sigma^*$ such that we can derive $S(s)$.

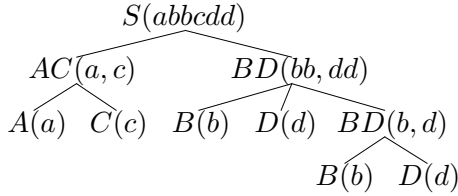
Here, we restrict attention to a normal form in which (i) each MCFG rule is *nondeleting* in the sense that every variable x_{ij} on the right occurs exactly once on the left, and (ii) each rule is either lexical or nonlexical, where a *lexical rule* is one in which $n = 0$ and $d(A) = 1$ and $t_1 \in \Sigma \cup \{\epsilon\}$, and a *nonlexical rule* is one in which $n \geq 0$ and each $t_i \in Var^*$. Clearly these additional restrictions do not affect the expressive power of the grammars.

2.1 Example 1

Consider this MCFG for $\{a^i b^j c^i d^j \mid i, j > 0\}$, with 5 non-lexical rules, 4 lexical rules, and start category S . We letter the rules for later reference:

- a. $S(x_0 x_1 x_2 x_3) :- AC(x_0, x_2), BD(x_1, x_3)$
- b. $AC(x_0 x_2, x_1 x_3) :- A(x_0), C(x_1), AC(x_2, x_3)$
- c. $AC(x_0, x_1) :- A(x_0), C(x_1)$
- d. $BD(x_0 x_2, x_1 x_3) :- B(x_0), D(x_1), BD(x_2, x_3)$
- e. $BD(x_0, x_1) :- B(x_0), D(x_1)$
- f. $A(a)$
- g. $B(b)$
- h. $C(c)$
- i. $D(d)$

With this grammar we can show that $abcdd$ has category S with a derivation tree like this:



See, for example, Kanazawa (2009) for a more detailed discussion of MCFGs in this format.

3 MGs as MCFGs

Michaelis (1998; 2001a) shows that every MG has a ‘strongly equivalent’ MCFG, in the sense that the MG derivation trees are a relabeling of the MCFG derivation trees. Here we present MGs as finite sets of lexical rules that define MCFGs. MG categories contain finite tuples of feature sequences, where the features include *categories* like N,V,A,P,..., *selectors* for those categories =N,=V,=A,=P,..., *licensors* +case,+wh,..., and *licensees* -case,-wh,... In our MCFG representation, a category is a tuple

$$\langle x, \delta_0, \delta_1, \dots, \delta_j \rangle$$

where (i) $j \geq 0$, (ii) $x = 1$ if the element is lexical and 0 otherwise, (iii) each δ_i is a nonempty feature sequence, and (iv) the category has dimension $j + 1$. An MG is then given by a specified start category and a finite set of lexical rules

$$\langle 1, \delta_0 \rangle(a).$$

for some $a \in \Sigma$. The MG defines the language generated by its lexicon together with MCFG rules determined by the lexicon, as follows. Let $\pi_2(Lex)$ be the set of feature sequences δ_0 contained in the lexical rules, and let k be the number of different types of licensees f that occur in the lexical rules. For all $0 \leq i, j \leq k$, all $x, y \in \{0, 1\}$, all $\alpha, \beta, \delta_i, \gamma_i \in \text{suffix}(\pi_2(Lex))$, and $\beta \neq \epsilon$, we have these ‘merge’ rules, broken as usual into the cases where (i) we are merging into complement position on the right, (ii) merging into specifier position on the left, or (iii) merging with something that is moving:

$$\begin{aligned}
 \langle 0, \alpha, \delta_1, \dots, \delta_j \rangle(s_0 t_0, t_1, \dots, t_j) :- \\
 \langle 1, =f\alpha \rangle(s_0), \\
 \langle x, f, \delta_1, \dots, \delta_j \rangle(t_0, \dots, t_j)
 \end{aligned}$$

$$\begin{aligned}
 \langle 0, \alpha, \delta_1, \dots, \delta_i, \gamma_1, \dots, \gamma_j \rangle(t_0 s_0, s_1, \dots, s_i, t_1, \dots, t_j) :- \\
 \langle 0, =f\alpha, \delta_1, \dots, \delta_i \rangle(s_0, \dots, s_i), \\
 \langle x, f, \gamma_1, \dots, \gamma_j \rangle(t_0, \dots, t_j)
 \end{aligned}$$

$$\begin{aligned}
 \langle 0, \alpha, \delta_1, \dots, \delta_i, \beta, \gamma_1, \dots, \gamma_j \rangle(s_0, \dots, s_i, t_0, \dots, t_j) :- \\
 \langle x, =f\alpha, \delta_1, \dots, \delta_i \rangle(s_0, \dots, s_i), \\
 \langle y, f\beta, \gamma_1, \dots, \gamma_j \rangle(t_0, \dots, t_j)
 \end{aligned}$$

And we have these ‘move’ rules, broken as usual into the cases where the moving element is landing, when $\delta_i = -f$,

$$\begin{aligned}
 \langle 0, \alpha, \delta_1, \dots, \delta_{i-1}, \delta_{i+1}, \dots, \delta_j \rangle \\
 (s_i s_0, s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_j) :- \\
 \langle 0, +f\alpha, \delta_1, \dots, \delta_j \rangle(s_0, \dots, s_j),
 \end{aligned}$$

and cases where the moving element must move again, when $\delta_i = -f\beta$,

$$\begin{aligned}
 \langle 0, \alpha, \delta_1, \dots, \delta_{i-1}, \beta, \delta_{i+1}, \dots, \delta_j \rangle(s_0, \dots, s_i) :- \\
 \langle 0, +f\alpha, \delta_1, \dots, \delta_j \rangle(s_0, \dots, s_i),
 \end{aligned}$$

where none of $\delta_1, \dots, \delta_{i-1}, \delta_{i+1}, \dots, \delta_j$ begin with $-f$. The language of the MG is the MCFL defined by the lexicon and all instances of these 5 rule schemes (always a finite set).

By varying the lexicon, MGs can define all the MCFLs (Michaelis, 2001b; Harkema, 2001b), i.e.,

the set-local multi-component tree adjoining languages (MCTALs) (Weir, 1988; Seki et al., 1991). TALs are a proper subset, defined by ‘well-nested 2-MCFGs’ (Seki et al., 1991; Kanazawa, 2009).

3.1 Example 2

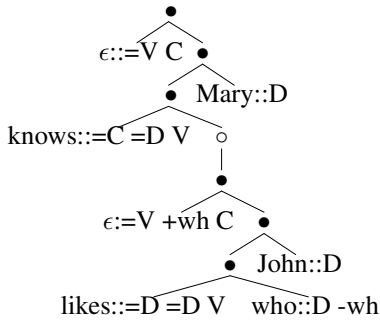
Consider the following lexicon containing 7 items, with the ‘complementizer’ start category C,

$\langle 1, =D =D V \rangle(\text{likes})$	$\langle 1, D \rangle(\text{Mary})$
$\langle 1, =C =D V \rangle(\text{knows})$	$\langle 1, D \rangle(\text{John})$
$\langle 1, =V C \rangle(\epsilon)$	$\langle 1, D -\text{wh} \rangle(\text{who})$
$\langle 1, =V +\text{wh} C \rangle(\epsilon)$	

Using the definition given just above, this determines an MG. This is a derivation tree for one of the infinitely many expressions of category C:

$\langle 0, C \rangle(\text{Mary knows who John likes})$	
$\langle 1, =V C \rangle(\epsilon)$	$\langle 0, V \rangle(\text{Mary knows who John likes})$
$\langle 0, =D V \rangle(\text{knows who John likes})$	$\langle 1, D \rangle(\text{Mary})$
$\langle 1, =C =D V \rangle(\text{knows})$	$\langle 0, C \rangle(\text{who John likes})$
	$\langle 0, +\text{wh} C, -\text{wh} \rangle(\text{John likes, who})$
$\langle 1, =V +\text{wh} C \rangle(\epsilon)$	$\langle 0, V, -\text{wh} \rangle(\text{John likes, who})$
$\langle 0, =D V, -\text{wh} \rangle(\text{likes, who})$	$\langle 1, D \rangle(\text{John})$
$\langle 1, =D =D V \rangle(\text{likes})$	$\langle 1, D -\text{wh} \rangle(\text{who})$

If we relabel this tree so that each instance of merge is labeled Merge or \bullet , and each instance of move is labeled Move or \circ , the result is the corresponding MG derivation tree, usually depicted like this:



In fact, the latter tree fully specifies the MCFG derivation above, because, in every MG derivation, for *every* internal node, the categories of the children determine which rule applies. This is easily verified by checking the 5 schemes for non-lexical rules on the previous page; the left side of each rule is a function of the right. Consequently the MCFG categories at the internal nodes can be regarded as specifying

the states of a deterministic finite state bottom-up tree recognizer for the MG derivation trees (Kobele et al., 2007; Graf, 2011; Kobele, 2011).

3.2 MCFGs need not be computed

We did not explicitly present the nonlexical MCFG rules used in the previous section §3.1, since they are determined by the lexical rules. The first rule used at the root of the derivation tree is, for example, an instance of the first rule scheme in §3, namely:

$$\langle 0, C \rangle(s_0 t_0) :- \langle 1, =V C \rangle(s_0), \langle 0, V \rangle(t_0).$$

Generating these non-lexical MCFG rules from the MG lexicon is straightforward, and has been implemented in (freely available) software by Guillaumin (2004). But the definition given in §3 requires that all feature sequences in all rules be suffixes of lexical feature sequences, and notice that in any derivation tree, like the one shown in §3.1, for example, feature sequences increase along the left branch from any node to the leaf which is its ‘head.’ Along any such path, the feature sequences increase one feature at a time until they reach the lexical leaf. So in effect, if we are building the derivation top-down, each step adds or ‘unchecks’ features in lexical sequences one at a time, and obviously the options for doing this can be seen without compiling out all the MCFG nonlexical rules.

4 The top-down recognizer

For any sequence s of elements of S , let $|s|$ =the length of s and $\text{nth}(i, s) = a$ iff $a \in S$, and for some $u, v \in S^*$, $s = uav$ and $|u| = i$. Adapting basic ideas from earlier work (Mainguy, 2010; Villemonte de la Clergerie, 2002) for TD recognition, we will instantiate variables not with strings but with indices $i \in \mathbb{N}^*$ to represent linear order of constituents, to obtain *indexed atoms* $A(i_1, \dots, i_{d(A)})$.

Consider any nonlexical rule $\alpha :- \gamma$ and any indexed atom β where

$$\begin{aligned} \alpha &= A(t_1, \dots, t_{d(A)}) \\ \beta &= A(i_1, \dots, i_{d(A)}) \\ \gamma &= B_0(x_{0_1}, \dots, x_{0_{d(B_0)}}), \dots, B_n(x_{n_1}, \dots, x_{n_{d(B_n)}}). \end{aligned}$$

For each variable x_{i_j} in γ , define

$$\text{index}_{\alpha, \beta}(x_{i_j}) = \begin{cases} i_k & \text{if } t_k = x_{i_j} \\ i_{kp} & \text{if } |t_k| > 1, x_{i_j} = \text{nth}(p, t_k). \end{cases}$$

Let $\text{index}_{\alpha,\beta}(\gamma)$ be the result of replacing each variable x_{i_j} in γ by $\text{index}_{\alpha,\beta}(x_{i_j})$. Finally, let $\text{trim}(\gamma)$ map γ to itself except in the case when every index in γ begins with the same integer n , in which case that initial n is deleted from every index.

Define a total order on the indices \mathbb{N}^* as follows. For any $\alpha, \beta \in \mathbb{N}^*$,

$$\alpha < \beta \text{ iff } \begin{cases} \alpha = \epsilon \neq \beta, \text{ or} \\ \alpha = i\alpha', \beta = j\beta', i < j, \text{ or} \\ \alpha = i\alpha', \beta = i\beta', \alpha' < \beta'. \end{cases}$$

For any atom α , let $\mu(\alpha)$ be the least index in α . So, for example, $\mu(AB(31, 240)) = 240$. And for any indexed atoms α, β , let $\alpha < \beta$ iff $\mu(\alpha) < \mu(\beta)$. We use this order to sort categories into left-to-right order in the ‘expand’ rule below.

We now define TD recognition in a deductive format. The state of the recognition sequence is given by a (remaining input, priority queue) pair, where the queue represents the memory of predicted elements, sorted according to $<$ so that they can be processed from left to right. We have 1 *initial axiom*, which predicts that input s will have start category S , where S initially has index ϵ :

$$\overline{(s, S(\epsilon))}$$

The main work is done by the *expand rule*, which pops atom α off the queue, leaving sequence Θ underneath. Then, for any rule $\beta :- \gamma$ with β of the same category as α , we compute $\text{index}_{\alpha,\beta}(\gamma)$, append the result and Θ , then sort and trim:

$$\frac{(s, \alpha\Theta)}{(s, \text{sort}(\text{trim}(\text{index}_{\alpha,\beta}(\gamma)\Theta)))} \beta :- \gamma$$

(We could use ordered insertion instead of sorting, and we could trim the indices much more aggressively, but we stick to simple formulations here.) Finally, we have a *scan rule*, which scans input a if we have predicted an A and our grammar tells us that $A(a)$. For all $a \in (\Sigma \cup \epsilon)$, $s \in \Sigma^*$, $n \in \mathbb{N}^*$:

$$\frac{(as, A(n)\Theta)}{(s, \Theta)} A(a)$$

A string s is accepted if we can use these rules to get from the start axiom to (ϵ, ϵ) . This represents the fact that we have consumed the whole input and there are no outstanding predictions in memory.

4.1 Example 1, continued.

Here is the sequence of recognizer states that accepts *abbccd*, using the grammar presented in §2.1:

- initial axiom:
 init. (abbccd, S(ϵ))
 expand with rule a:
 1. (abbccd, AC(0,2),BD(1,3))
 expand with rule c (note sort):
 2. (abbccd, A(0),BD(1,3),C(2))
 scan with rule f:
 3. (bbcdd, BD(1,3),C(2))
 expand with rule d:
 4. (bbcdd, B(10),BD(11,31),C(2),D(30))
 scan with rule g:
 5. (bcdd, BD(11,31),C(2),D(30))
 expand with rule e:
 6. (bcdd, B(11),C(2),D(30),D(31))
 scan with rule g:
 7. (cdd, C(2),D(30),D(31))
 scan with rule h (note trim removes 3):
 8. (dd, D(0),D(1))
 scan with rule i: (note trim removes 1)
 9. (d, D())
 scan with rule i:
 10. (ϵ , ϵ)

The number of recognizer steps is always exactly the number of nodes in the corresponding derivation tree; compare this accepting sequence to the derivation tree shown in §2.1, for example.

5 Properties and extensions

5.1 Adding adjunction, head movement

Frey and Gärtner (2002) propose that adjunction be added to MGs by (i) allowing another kind of selecting feature $\approx f$, which selects but does not ‘check and delete’ the feature f of a phrase that it modifies, where (ii) the head of the result is the selected, ‘modified’ phrase that it combines with, and (iii) the selecting ‘modifier’ cannot have any constituents moving out of it. We can implement these ideas by adding a rule scheme like the following (compare the first rule scheme in §3):

$$\langle 0, f\alpha, \delta_1, \dots, \delta_j \rangle (t_0 s_0, t_1, \dots, t_j) :- \\ \langle y, f\alpha, \delta_1, \dots, \delta_j \rangle (t_0, \dots, t_j), \\ \langle x, \approx f \rangle (s_0).$$

Note this rule ‘attaches’ the modifier on the right. We could also allow left modifiers, but in the examples below will only use this one.

Some analyses of simple tensed sentences say that tense affixes ‘hop’ onto the verb after the verb has combined with its object. Affix hopping and head movement are more challenging than adjunction, but previous approaches can be adapted to the present perspective by making two changes: (i) we keep the head separate from other material in its phrase until that phrase is merged with another phrase, so now every non-lexical category A has $d(A) \geq 3$ and (ii) we add diacritics to the selection features to indicate whether hopping or head movement should apply in the merge step. To indicate that a head A selects category f we give A the feature $=f$, but to indicate that the head of A should hop onto the head of the selected constituent, we give A the feature $f=>$. Essentially this representation of MGs with head movement and affix hopping as MCFGs is immediate from the formalization in Stabler (2001) and the automated translation by Guillaumin (2004). The examples in this paper below will use only affix hopping which is defined by the following modified version of the first rule in §3:

$$\begin{aligned} \langle 0, \alpha, \delta_1, \dots, \delta_j \rangle (\epsilon, \epsilon, t_s t_h s_h t_c, t_1, \dots, t_j) :- \\ \langle 1, f=>\alpha \rangle (s_h), \\ \langle x, f, \delta_1, \dots, \delta_j \rangle (t_s, t_h, t_c, t_1, \dots, t_j) \end{aligned}$$

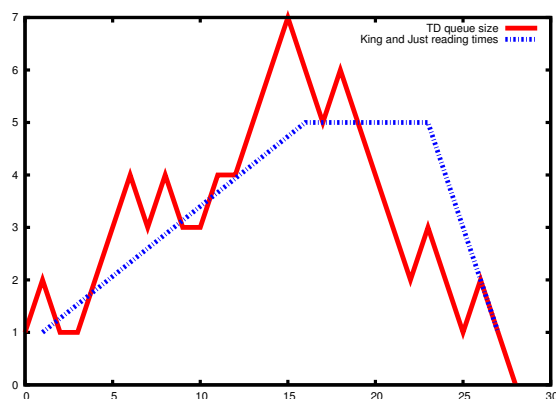
The first atom on the right side of this rule, the ‘selector’, is a lexical head with string s_h . The second atom on the right of the rule has string components t_s, t_h, t_c (these are the specifier, head, and complement strings) together with $j \geq 0$ moving elements t_1, \dots, t_j . In the result on the left, we see that the lexical selector s_h is ‘hopped’ to the right of the selected head t_h , where it is sandwiched between the other concatenated parts of the selected phrase, leaving ϵ in the head position. Since the usual start category C now has 3 components, like every other head, we begin with a special category S that serves only to concatenate the 3 components of the matrix complementizer phrase, by providing the recognizer with this additional initializing rule:

$$S(s_s s_h s_c) :- \langle x, C \rangle (s_s, s_h, s_c).$$

The nature of adjunction is not quite clear, and there is even less consensus about whether head movement or affix hopping or both are needed in grammars of human languages, but these illustrate

how easily the MCFG approach to MGs can be extended. Like many of the other MG variants, these extensions do not change the class of languages that can be defined (Stabler, 2011), and the recognizer defined in §4 can handle them without change.

With head movement and adjunction we can, for example, provide a roughly traditional analysis of the famous example sentence from King and Just (1991) shown in Figure 1. Note again that the derivation tree in that figure has lexical items at the leaves, and these completely determine the non-lexical rules and the structure of the derivation. Various representations of the ‘derived trees’, like the X-bar tree shown in this figure, are easily computed from the derivation tree (Kobelet al., 2007). And Figure 2 shows the recognizer steps accepting that sentence. Plotting queue size versus recognizer step, and simply overlaying the King and Just self-paced reading times to see if they are roughly similar, we see that, at least in sentences like these, readers go more slowly when the queue gets large:



Recent work has challenged the claim that reading times are a function of the number of predictions in memory, (e.g., Nakatani and Gibson, 2008, p.81) but preliminary studies suggest that other performance measures may correlate (Bachrach, 2008; Brennan et al., 2010; VanWagenen et al., 2011). Exploring these possibilities is beyond the scope of this paper. The present point is that any analysis expressible in the MG formalism can be parsed transparently with this approach, assessing its memory demands; partially parallel beam search models for ambiguity, used in natural language engineering, can also be straightforwardly assessed.

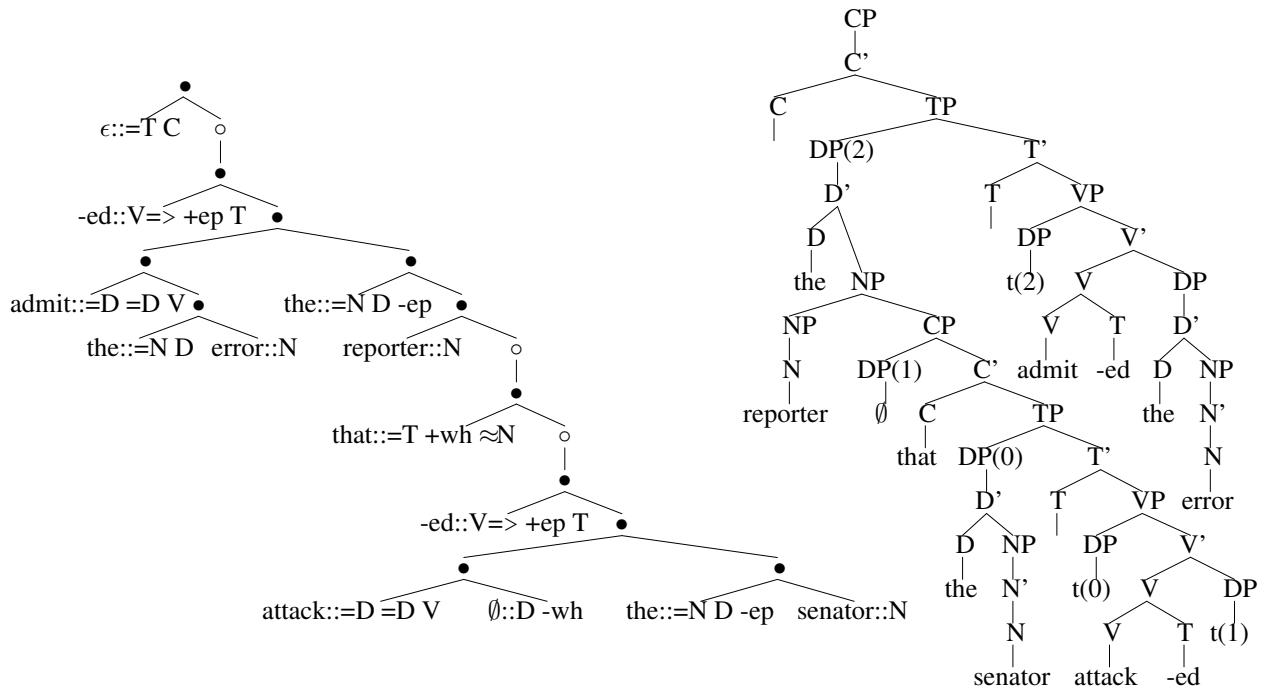


Figure 1: 28 node derivation tree and corresponding X-bar tree for King and Just (1991) example

5.2 Infinite languages with finite memory

Although memory use is not the main concern of this paper, it is worth noting that, as in stack-based CF models, memory demands do not necessarily increase without bound as sentence length increases. So for example, we can extend the naive grammar of Figure 2 to accept *this is the man that kiss -ed the maid that milk -ed the cow that toss -ed the dog that worry -ed the cat that chase -ed the rat*, a sentence with 6 clauses, and use no more memory at any time than is needed for the 2 clause King and Just example. Dynamic, chart-based parsing methods usually require more memory without bound as sentence length grows, even when there is little or no indeterminacy.

5.3 Connectedness

More directly relevant to incremental models is the fact that the portions of the derivation traversed at any point in TD recognition are all connected to each other, their syntactic relations are established. As we see in all our examples, the TD recognizer is always traversing the derivation tree on paths connected to the root; while the indexing and sorting ensures that the leaves are scanned in the order of their appear-

ance in the derived X-bar tree. Left corner traversals do not have this property. Consider a sentence like *the reporter poured the egg in the bowl over the flour*. In a syntax in the spirit of the one we see in Figure 1, for example, *in the bowl* could be right adjoined to the direct object, and *over the flour* right adjoined to VP. Let VP1 be the parent of *over the flour*, and VP2 its sister. With LC, VP1 will be predicted right after the subject is completed. But the verb is the left corner of VP2, and VP2 will not be attached to VP1 – and so the subject and verb will not be connected – until VP1 is completed. This delay in the LC attachment of the subject to the verb can be extended by adding additional right modifiers to the direct object or the verb phrase, but the evidence suggests that listeners make such connections immediately upon hearing the words, as the TD recognizer does.

6 Future work

Standard methods for handling indeterminacy in top-down CF parsers, when there are multiple ways to expand a derivation top down, are easily adapted to the MCFG and MG parsers proposed here. With backtracking search, left recursion can cause non-

termination, but a probabilistic beam search can do better. For $\alpha = (i, \Theta)$ any recognizer state, let $\text{step}(\alpha)$ be the (possibly empty) sequence of all the next states that are licensed by the rules in §3 (always finitely many). A probabilistic beam search uses the rules,

$$\frac{}{\langle (s, S(\epsilon)) \rangle} \text{init} \quad \frac{\alpha\Theta}{\text{prune}(\text{sort}_C(\text{step}(\alpha)\Theta))} \text{search},$$

popping a recognizer state α off the top of the queue $\alpha\Theta$, appending $\text{step}(\alpha)$ and Θ , then sorting and pruning the result. The sort in the search steps is done according to the probability of each parser

state in context C , where the context may include a history of previous recognizer steps – i.e. of each derivation up to this point – but also possibly extrasentential information of any sort. The pruning rule acts to remove highly improbable analyses, and success is achieved if a step puts (ϵ, ϵ) on top of the queue. Roark shows that this ability to condition on material not in parser memory – indeed on anything in the left context – can allow better estimates of parse probability. On small experimental grammars, we are finding that TD beam search performance can be better than our chart parsers using the same grammar. Further feasibility studies are in

1	init.	(trttsa-a-te, S(ϵ))
1	init.	(trttsa-a-te, $\langle 0, C \rangle (0, 1, 2)$)
2	1.	(trttsa-a-te, $\langle 1, =TC \rangle (1), \langle 0, T \rangle (20, 21, 22)$)
1	2.	(trttsa-a-te, $\langle 0, T \rangle (0, 1, 2)$)
1	3.	(trttsa-a-te, $\langle 0, +epT, -ep \rangle (01, 1, 2, 00)$)
2	4.	(trttsa-a-te, $\langle 0, V, -ep \rangle (20, 21, 23, 00), \langle 1, V=>+epT \rangle (22)$)
3	5.	(trttsa-a-te, $\langle 0, D, -ep \rangle (000, 001, 002), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
4	6.	(trttsa-a-te, $\langle 1, =ND, -ep \rangle (001), \langle 0, N \rangle (0020, 0021, 0022), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
3	7.	(trttsa-a-te, $\langle 0, N \rangle (0020, 0021, 0022), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
4	8.	(trttsa-a-te, $\langle 1, N \rangle (0021), \langle 0, \approx N \rangle (00220, 00221, 00222), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
3	9.	(ttsa-a-te, $\langle 0, \approx N \rangle (00220, 00221, 00222), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
3	10.	(ttsa-a-te, $\langle 0, +wh \approx N, -wh \rangle (002201, 00221, 00222, 002200), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
4	11.	(ttsa-a-te, $\langle 0, T, -wh \rangle (002220, 002221, 002222, 002200), \langle 1, =T+wh \approx N \rangle (00221), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
4	12.	(ttsa-a-te, $\langle 0, +epT, -ep, -wh \rangle (0022201, 002221, 002222, 0022200, 002200), \langle 1, =T+wh \approx N \rangle (00221), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
5	13.	(ttsa-a-te, $\langle 0, V, -ep, -wh \rangle (0022220, 0022221, 0022223, 0022200, 002200), \langle 1, =T+wh \approx N \rangle (00221), \langle 1, V=>+epT \rangle (002222), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
6	14.	(ttsa-a-te, $\langle 0, =DV, -wh \rangle (0022220, 0022221, 0022223, 002200), \langle 1, =T+wh \approx N \rangle (00221), \langle 0, D, -ep \rangle (0022200, 0022201, 0022202), \langle 1, V=>+epT \rangle (002222), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
7	15.	(ttsa-a-te, $\langle 1, D, -wh \rangle (002200), \langle 1, =T+wh \approx N \rangle (00221), \langle 0, D, -ep \rangle (0022200, 0022201, 0022202), \langle 1, =D=DV \rangle (0022221), \langle 1, V=>+epT \rangle (002222), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
6	16.	(ttsa-a-te, $\langle 1, =T+wh \approx N \rangle (00221), \langle 0, D, -ep \rangle (0022200, 0022201, 0022202), \langle 1, =D=DV \rangle (0022221), \langle 1, V=>+epT \rangle (002222), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
5	17.	(tsa-a-te, $\langle 0, D, -ep \rangle (0022200, 0022201, 0022202), \langle 1, =D=DV \rangle (0022221), \langle 1, V=>+epT \rangle (002222), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
6	18.	(tsa-a-te, $\langle 1, =ND, -ep \rangle (00222001), \langle 1, N \rangle (00222002), \langle 1, =D=DV \rangle (0022221), \langle 1, V=>+epT \rangle (002222), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
5	19.	(sa-a-te, $\langle 1, N \rangle (00222002), \langle 1, =D=DV \rangle (0022221), \langle 1, V=>+epT \rangle (002222), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
4	20.	(a-a-te, $\langle 1, =D=DV \rangle (0022221), \langle 1, V=>+epT \rangle (002222), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
3	21.	(-a-te, $\langle 1, V=>+epT \rangle (002222), \langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
2	22.	(a-te, $\langle 0, =DV \rangle (20, 21, 23), \langle 1, V=>+epT \rangle (22)$)
3	23.	(a-te, $\langle 1, =D=DV \rangle (1), \langle 1, V=>+epT \rangle (2), \langle 0, D \rangle (30, 31, 32)$)
2	24.	(-te, $\langle 1, V=>+epT \rangle (2), \langle 0, D \rangle (30, 31, 32)$)
1	25.	(te, $\langle 0, D \rangle (30, 31, 32)$)
2	26.	(te, $\langle 1, =ND \rangle (1), \langle 1, N \rangle (2)$)
1	27.	(e, $\langle 1, N \rangle (2)$)
0	28.	(ϵ , ϵ)

Figure 2: 28 step TD recognition of derivation in Figure 1, abbreviating input words by their initial characters. The left column indicates queue size, plotted in §5.1.

progress.

The recognizer presented here simplifies Mainguy's (2010) top-down MG recognizer by generalizing it to handle an MCFG normal form, so that a wide range of MG extensions are immediately accommodated. This is made easy when we adopt Kanazawa's Horn clause formulation of MCFGs where the order of variables on the left side of the rules so visibly indicates the surface order of string components. With the Horn clause notation, the indexing can be string-based and general rather than tree-based and tied to particular assumptions about how the MGs work. Transparently generalizing the operations of CF TD recognizers, the indexing and operations here are also slightly simpler than 'thread automata' (Villemonde de la Clergerie, 2002). Compare also the indexing, sometimes more or less similar, in chart-based recognizers of MCF and closely related systems (Burden and Ljunglöf, 2005; Harkema, 2001c; Boullier, 1998; Kallmeyer, 2010).

Mainguy shows that when the probability of a derivation is the product of the rule probabilities, as usual, and when those rule probabilities are given by a consistent probability assignment, a beam search without pruning will always find a derivation if there is one. When there is no derivation, though, an unpruned search can fail to terminate; a pruning rule can guarantee termination in such cases. Those results extend to the MCFG recognizers proposed here. Various applications have found it better to use a beam search with top-down recognition of left- or right-corner transforms of CF grammars (Roark, 2001; Roark, 2004; Schuler, 2010; Wu et al., 2010); those transforms can (but need not always) disrupt grammatical connectedness as noted in §5.3. Work in progress explores the possibilities for such strategies in incremental MCFG parsing. It would also be interesting to generalize Hale's (2011) "rational parser" to these grammars.

Acknowledgments

Thanks to Thomas Mainguy, Sarah VanWagenen and Éric Villemonde de la Clergerie for helpful discussions of this material.

References

- Gerry T. M. Altmann and Jelena Mirković. 2009. Incrementality and prediction in human sentence processing. *Cognitive Science*, 33:583–809.
- Asaf Bachrach. 2008. *Imaging Neural Correlates of Syntactic Complexity in a Naturalistic Context*. Ph.D. thesis, Massachusetts Institute of Technology.
- Pierre Boullier. 1998. Proposal for a natural language processing syntactic backbone. Technical Report 3242, Projet Atoll, INRIA, Rocquencourt.
- Jonathan Brennan, Yuval Nir, Uri Hasson, Rafael Malach, David J. Heeger, and Liinay Pylkkänen. 2010. Syntactic structure building in the anterior temporal lobe during natural story listening. *Forthcoming*.
- Håkan Burden and Peter Ljunglöf. 2005. Parsing linear context-free rewriting systems. In *Ninth International Workshop on Parsing Technologies, IWPT'05*.
- Craig G. Chambers, Michael K. Tanenhaus, Kathleen M. Eberhard, Hana Filip, and Greg N. Carlson. 2004. Actions and affordances in syntactic ambiguity resolution. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 30(3):687–696.
- Cristiano Chesi. 2007. An introduction to phase-based minimalist grammars: Why *move* is top-down from left-to-right. Technical report, Centro Interdipartimentale di Studi Cognitivi sul Linguaggio.
- Vera Demberg and Frank Keller. 2009. A computational model of prediction in human parsing: Unifying locality and surprisal effects. In *Proceedings of the 29th meeting of the Cognitive Science Society (CogSci-09)*, Amsterdam.
- Werner Frey and Hans-Martin Gärtner. 2002. On the treatment of scrambling and adjunction in minimalist grammars. In *Proceedings, Formal Grammar'02*, Trento.
- Thomas Graf. 2011. Closure properties of minimalist derivation tree languages. In *Logical Aspects of Computational Linguistics, LACL'11*, *Forthcoming*.
- Matthieu Guillaumin. 2004. Conversions between mildly sensitive grammars. UCLA and École Normale Supérieure. <http://www.linguistics.ucla.edu/people/stabler/epssw.htm>.
- Nicholas J. Haddock. 1989. Computational models of incremental semantic interpretation. *Language and Cognitive Processes*, 4((3/4)):337–368.
- John T. Hale. 2011. What a rational parser would do. *Cognitive Science*, 35(3):399–443.
- Henk Harkema. 2001a. A characterization of minimalist languages. In *Proceedings, Logical Aspects of Computational Linguistics, LACL'01*, Port-aux-Rocs, Le Croisic, France.
- Henk Harkema. 2001b. A characterization of minimalist languages. In Philippe de Groote, Glyn Morrill, and

- Christian Retoré, editors, *Logical Aspects of Computational Linguistics*, Lecture Notes in Artificial Intelligence, No. 2099, pages 193–211, NY: Springer.
- Henk Harkema. 2001c. *Parsing Minimalist Languages*. Ph.D. thesis, University of California, Los Angeles.
- Aravind Joshi. 1985. How much context-sensitivity is necessary for characterizing structural descriptions. In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Processing: Theoretical, Computational and Psychological Perspectives*, pages 206–250. Cambridge University Press, NY.
- Laura Kallmeyer. 2010. *Parsing beyond context-free grammars*. Springer, NY.
- Makoto Kanazawa. 2009. A pumping lemma for well-nested multiple context free grammars. In *13th International Conference on Developments in Language Theory, DLT 2009*.
- Yoshihide Kato and Shigeki Matsubara. 2009. Incremental parsing with adjoining operation. *IE-ICE Transactions on Information and Systems*, E92.D(12):2306–2312.
- Jonathan King and Marcel Adam Just. 1991. Individual differences in syntactic processing: the role of working memory. *Journal of Memory and Language*, 30:580–602.
- Gregory M. Kobele, Christian Retoré, and Sylvain Salvati. 2007. An automata-theoretic approach to minimalism. In J. Rogers and S. Kepser, editors, *Model Theoretic Syntax at 10, ESSLLI'07*.
- Gregory M. Kobele. 2011. Minimalist tree languages are closed under intersection with recognizable tree languages. In *Logical Aspects of Computational Linguistics, LACL'11*, Forthcoming.
- Thomas Mainguy. 2010. A probabilistic top-down parser for minimalist grammars. <http://arxiv.org/abs/1010.1826v1>.
- Jens Michaelis. 1998. Derivational minimalism is mildly context-sensitive. In *Proceedings, Logical Aspects of Computational Linguistics, LACL'98*, pages 179–198, NY: Springer.
- Jens Michaelis. 2001a. *On Formal Properties of Minimalist Grammars*. Ph.D. thesis, Universität Potsdam. *Linguistics in Potsdam 13*, Universitätsbibliothek, Potsdam, Germany.
- Jens Michaelis. 2001b. Transforming linear context free rewriting systems into minimalist grammars. In P. de Groote, G. Morrill, and C. Retoré, editors, *Logical Aspects of Computational Linguistics, LNCS 2099*, pages 228–244, NY: Springer.
- Kentaro Nakatani and Edward Gibson. 2008. Distinguishing theories of syntactic expectation cost in sentence comprehension: Evidence from Japanese. *Linguistics*, 46(1):63–86.
- Brian Roark and Mark Johnson. 1999. Efficient probabilistic top-down and left-corner parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 421–428.
- Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.
- Brian Roark. 2004. Robust garden path parsing. *Natural Language Engineering*, 10(1):1–24.
- William Schuler. 2010. Incremental parsing in bounded memory. In *Proceedings of the 10th International Workshop on Tree Adjoining Grammars and Related Frameworks, TAG+10*.
- Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.
- Libin Shen and Aravind Joshi. 2005. Incremental LTAG parsing. In *Proceedings, Human Language Technology Conference and Conference on Empirical Methods in Human Language Processing*.
- Edward P. Stabler. 2001. Recognizing head movement. In Philippe de Groote, Glyn Morrill, and Christian Retoré, editors, *Logical Aspects of Computational Linguistics*, Lecture Notes in Artificial Intelligence, No. 2099, pages 254–260. Springer, NY.
- Edward P. Stabler. 2011. Computational perspectives on minimalism. In Cedric Boeckx, editor, *Oxford Handbook of Linguistic Minimalism*, pages 617–641. Oxford University Press, Oxford.
- Sarah VanWagenen, Jonathan Brennan, and Edward P. Stabler. 2011. Evaluating parsing strategies in sentence processing. In *Proceedings of the CUNY Sentence Processing Conference*.
- K. Vijay-Shanker and David Weir. 1994. The equivalence of four extensions of context free grammar formalisms. *Mathematical Systems Theory*, 27:511–545.
- Éric Villemonte de la Clergerie. 2002. Parsing MCS languages with thread automata. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks, TAG+6*.
- David Weir. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania, Philadelphia.
- Stephen Wu, Asaf Bachrach, Carlos Cardenas, and William Schuler. 2010. Complexity metrics in an incremental right-corner parser. In *Proceedings of the 48th Annual Meeting of the Association for Computer Linguistics*, pages 1189–1198.