revised 2011-10-06 10:06 stabler@ucla.edu

Edward P. Stabler

# Recursion in grammar and performance

In the last 50 years of cognitive science, linguistic theory has proposed more and more articulated structures, while computer science has shown that simpler, flatter structures are more easily processed. If we are interested in adequate models of human linguistic abilities, models that explain the very rapid and accurate human recognition and production of ordinary fluent speech, it seems we need to come to some appropriate understanding of the relationship between these apparently opposing pressures for more and less structure. Here we show how the apparent conflict disappears when it is considered more carefully. Even when we regard the linguists' project as a psychological one, there is no pressure for linguists to abandon their rather deep structures in order to account for our easy production and recognition of fluent speech. The deeper, more recursive structures reflect insights into similarities among linguistic constituents and operations, but a processor can compute exactly these structures without the extra effort that deeper analyses might seem to require. To show how this works, we review §1 standard notions of recursive depth and some basic ideas about how computations can be implemented, §2 a consensus position about linguistic structure, and §3 some large classes of parsing models that compute exactly the consensus structures in such a way that the depth of the linguistic analysis does not correspond to processing depth. The apparent tension is resolved in this last step, with a proper understanding of how computations can be implemented. The needed perspective on what it is to represent and manipulate structures, to implement a computation, is completely standard. From this unifying perspective, we will argue that adequate performance models must be more superficial than adequate linguistic models, that this is unsurprising, and that the two perspectives are reconciled by a substantial theory of how linguistic computations are implemented. In a sense that will be made precise, the recursive depth of a structural analysis does not correspond in any simple way to depth of the calculation of that structure in linguistic performance.

UCLA Department of Linguistics

## 1. Recursion and depth

Ordinary fluent speech is perceived as having parts, in temporal sequence. Considering the natural patterns of parts found in these sequences, we find first that they do not have a principled upper bound on their length; in this sense, languages are infinite. So to define the patterns, we define an infinite set rather than imposing some arbitrary length bound. The standard definitions of infinite sets of sequences are recursive, in the sense that the set being defined is used in its own definition. This idea is familiar from the definitions of other infinite sets. For example, we can define the set of natural numbers as the smallest set containing 0 such that, for every natural number, its successor is also a natural number. This definition is sometimes presented with a notation like the following, in which we see that the set $\mathbb{N}$ of numbers that we are defining appears on the right, in the definition:

$$\mathbb{N} := 0 \mid s(\mathbb{N}).$$

This simply says that the set of natural numbers contains 0 and also the successor of every natural number (and nothing more). The successor function is also said to be recursive, since it applies to its own output. In a similar way, the theorems of standard logics are defined as the axioms together with everything derived from the theorems by the rules of inference $\mathcal{I}$,

$$\mathrm{T} := \mathrm{Axioms} \mid \mathcal{I}(\mathrm{T}).$$

Here again the definition of the set of theorems T is recursive, as are the inference rules if they can apply to their own output. Languages can be defined by rewrite rules, but it can be more convenient to use an inductive definition similar to the previous examples:

$$\mathrm{L} := \mathrm{Lex} \mid \mathcal{F}(\mathrm{L}).$$

This says that a language includes a lexicon and everything that can be built from lexical items by some structure building operations $\mathcal{F}$. Many generative grammars can be stated in this form, even many of those that are not explicitly presented this way [45].

Given a recursive definition of a set $S$ and a particular thing $a$, how can we tell whether $a$ is in the set $S$? Clearly, one way to do this is by proving from the definition that $a$ is included. For example, we can show that s(s(0)) is a number using our definition of $\mathbb{N}$ by observing first that s(s(0)) is a number if s(0) is, and that s(0) is a number if 0 is, and 0 is a number. In a sense, this establishes that s(s(0)) is a number by using the definition 3 times. We can present this proof as a 'derivation tree', where $\bullet$ is the successor function:

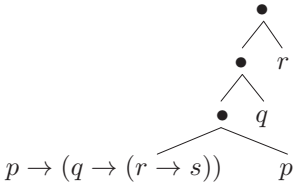
This tree has depth 3, since that is the number of steps from the root to the leaf. Returning to our question, how can we tell whether something is in a set defined by a recursive definition, the answer is: by finding a 'derivation' of that element, in this sense.

Suppose we have a propositional logic with this set of four axioms,

$$\text{Axioms} := \{p, q, r, (p \to (q \to (r \to s)))\},$$

and with the inference rule *modus ponens*, $\mathcal{I} = \{\text{mp}\}$. Then, with the definition $\text{T} := \text{Axioms} \mid \mathcal{I}(\text{T})$, we can show that $s$ is a theorem in T with the derivation standardly presented as shown on the left below, which is just another form of the derivation tree shown on the right, where $\bullet$ is mp:

$$\dfrac{\dfrac{\dfrac{p \to (q \to (r \to s)) \qquad p}{q \to (r \to s)} \qquad q}{r \to s} \qquad r}{s}$$



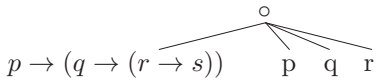

The depth of this derivation tree is 4, since that is the length of its longest path from root to leaf. In effect, on this path, we use the recursive definition 4 times to get this tree.

Now a first point of interest is that there are other definitions of the same set of theorems T, where many theorems have much shallower derivations. For example, suppose that in addition to mp, we have an inference rule which takes 4 premises instead of 2, and, in effect, applies mp 3 times at once:

$$\dfrac{\alpha \to (\beta \to (\gamma \to \psi)) \qquad \alpha \qquad \beta \qquad \gamma}{\psi}$$

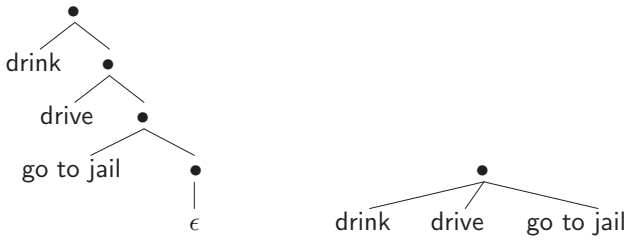
With this rule, even though the set T is not changed in any way, we can derive s in 1 step instead of 3 (here using $\circ$ to represent our new rule),

$$\dfrac{p \to (q \to (r \to s)) \qquad p \qquad q \qquad r}{s}$$




With this proof we have saved just a couple of steps, but obviously for longer proofs, the number of steps we save can increase without bound. That is, in the new system with two inference rules, many results can be established with much less recursive depth. There is a large literature in logic about how to 'speed up' proofs by adding new mechanisms like this [27,85,11].

Consider the sequence of actions: drink, drive, and go to jail. We might assume that that there is no recursion in a sequence like that, but now we see that the question depends on how the sequence is defined. For example, if at each decision point, a similar decision process $\bullet$ is operative, then we could have a recursive depth of 3 as shown on the left below. But if the three actions just follow one after the other, then there is just 1 step:




(T) (T')

In the absence of a theory of action or any other application, there may be no reason to prefer one description over the other. But turning now to human language, we argue for the following points: a methodological point M1, and two rather secure empirical hypotheses H1 and H2.

(M1) There can be strong reasons to favor a highly recursive definition like (T) over a less recursive one like (T'), or vice versa.

(H1) There is a significant consensus about the kinds of recursive operations needed to define human languages.

(H2) Performance models need to flatten where linguists don't.

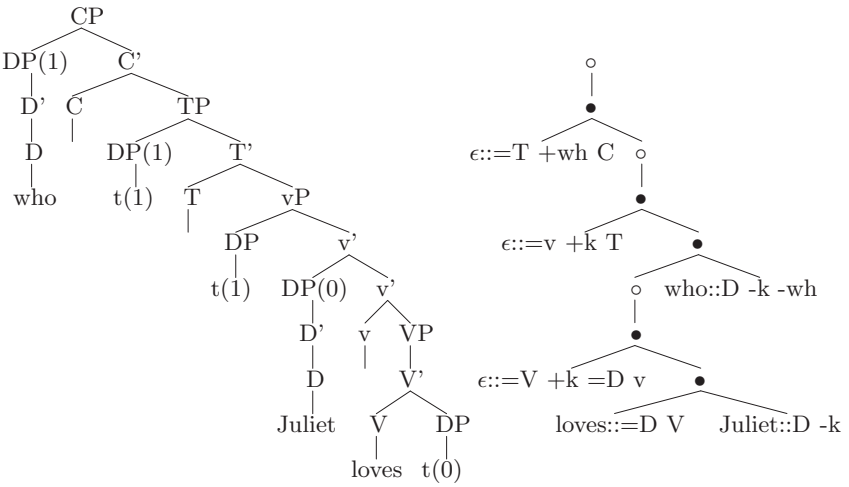
Before explaining and defending these points, two more quick preliminaries are needed: in §1.1 we define a couple of notions based on the previous discussion, and in §1.2 we briefly consider a computer science perspective on definitional complexity and implementation.

### *1.1. Complexity: size, depth , depth$_o$*

Let's say definition of L is recursive if the definition uses L, as in the examples above. And an operation (like the successor function, or *modus ponens*, or a rule of grammar) is recursive if it can apply to its own output. Taking a slightly more complex example, in a formal grammar inspired by recent Chomskian syntax, there is an operation called *merge* $\bullet$ and an operation called move $\circ$, so a language L is defined as follows [81,61,84]:

$$\mathrm{L} := \mathrm{Lex} \mid \mathcal{F}(\mathrm{L}), \text{ where } \mathcal{F} = \{\bullet, \circ\}.$$

A grammar of this kind can establish that *who Juliet loves* is a 'complementizer clause' (CP) with a derivation like the one shown below right, which is more conventionally shown with the corresponding derived tree on the left:




size 31, depth 11, $\text{depth}_{\text{CP}}$ 1, $\text{depth}_{\text{v'}}$ 2 size 14, depth 9, $\text{depth}_{\bullet}$ 5, $\text{depth}_{\circ}$ 3

The features associated with lexical items at the leaves of the derivation tree determine what derivational steps can apply, as explained in [81,34,61,84], but the relevant thing here is the recursion in the derivation. As indicated in the captions above, let the *size* of a tree be the number of nodes; let the *depth* of a tree be the length of its longest path, and for each operation $f$ in the tree (or for each node label when the tree is not a derivation tree), let the $\textit{depth}_f$ be the greatest number of occurrences of $f$ on any path from root to leaf. Let's say that a tree is recursive if for some $f$, $\text{depth}_f$ is greater than 1. Recursion in a derivation tree then indicates that some operation $f$ is applying to its own output (perhaps mediated by other operations).

Linguists and psycholinguists usually give most attention to derived trees, like the one on the left, above, but for most processing problems, that is a mistake! For parsing, for example, it is the derivation that is more fundamental, since the existence of a derivation is what determines whether a structure is assigned at all, and the derivations immediately determine what the derived structures are (while the converse does not generally hold).[1] We will accordingly attend primarily to recursion in the operations, in the derivation tree, rather than to recursion in the associated derived structures.

### *1.2. Computational preliminaries*

In computer science, a problem is often approached by, first, giving a clear and perspicuous definition of what needs to be computed, and then carefully searching for efficient ways of computing it. These two objectives pull different directions! This happens because, roughly, (T) the simplest definitions

[1] A certain arbitrariness in the derived structures can be revealed in a precise way by showing that quite different derived structures can yield isomorphic sets of derivations, with identical leaves – a key idea behind the results established in Theorems 1 and 2, mentioned below.

may require many steps to derive a result, while (P) the most time-efficient computations are those that take rather few steps.

(T) To specify what is computed, we aim for simplicity.

(P) For efficiency, aim to keep calculations flat.

The first point is very important! Computer scientists have slogans emphasizing the importance of simplicity and that a loop or recursion should always be used whenever some step or sequence of steps needs to be repeated:[2]

> *'Inside every big problem is a little problem trying to get out.'*
> *'Two or more, use a* for*.'*

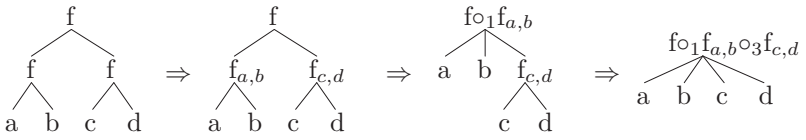
If a specification is too complex, it becomes infeasible to assess whether it is right! Linguists emphasize the analogous point too: we would like to find as simple a perspective as possible on as much as possible in the language. For example, Chomsky [14] notes that there are fundamental similarities between merge • and move ∘, and he accordingly frames his theory in terms of one merge operation •. Note that if all the operations in the derivation tree above were instances of the same rule •, then $\text{depth}_{\bullet}$ is increased.

However, the second point, (P), is also important! Obviously, computer scientists often want to compute solutions with minimum resources, and to model natural computations that are fast and easy, we would like a model which reveals how those computations can be fast and easy.

The conflict between (T) and (P) is tamed in computer science by the theory of how an algorithm can be implemented. When we can specify a function easily, in a 'high level language', we can sometimes convert that specification into an efficient implementation with a 'compiler'. Many of the steps in the conversions automated by optimizing compilers serve precisely to decrease recursive depth. Some steps of this kind include: recursion unfolding, loop unrolling, partial evaluation, inlining, macro expansion, deforestation, and many others [91,3,38,59,78,13,53,47,32].[3] There is a substantial body of theory here, but the basic idea is similar to the speed-up already mentioned above. For example, given a specification that requires a calculation of $\text{depth}_f\ 2$, as shown on the left below, a first conversion might specialize the preterminal steps of the calculation:

[2] The first of these is sometimes attributed to Tony Hoare, the latter to Edsger Dijkstra.

[3] In compiling programming languages, loops need to be treated differently from recursion. These two mechanisms are expressively equivalent. Indeed this kind of equivalence was a significant motivation for the Church-Turing thesis. For present purposes, we will not introduce loops, and the difference between loops and recursion on standard machine architectures (or neurally plausible ones) will not be explored. The relevant point here is simply that both loops and recursion can define calculations of various recursive depths, and these depths are decreased by optimizing compilers.




This first conversion of the problem decreases the $\text{depth}_f$ from 2 to 1. The overall size and depth of the tree remains the same, but in many cases $f_{a,b}, f_{c,d}$ can be more efficient than two applications of $f$ just because the specialized functions need not handle such a wide range of arguments. The second and third conversions shown above 'compose' the functions together, 'unfolding' the computation, as we did earlier when we composed mp with itself several times. These latter steps decrease the size and depth of the calculation overall. Typically,

- steps like these decrease recursive depth, but increase program size;
- finding a 'best' unfolding for a particular machine with size less than a fixed finite bound is typically intractable [77]; some recent 'profiling compilers' optimize those parts of the code that is most heavily used, a sensible 'practice effect' strategy which could have an analog in natural systems;
- the 'best' unfolding also depends, of course, on architecture of the machine running it, on what steps are basic, as is noted in linguistic contexts where we are interested in the possible neural implementations [80, 68, 7].

In sum, when considering how a computation might be realized, for example a computation that decides whether a sequence has a derivation from some grammar, it is important to remember that this kind of problem can often be solved (i.e. solved exactly and correctly) with definitions that have been 'flattened' by methods like this.[4] This kind of flattening affects processing complexity without any change at all to the mapping computed.

## 2. Linguistic structure

Now let's turn to the definition of linguistic structure. The previous discussion suggests that it may be useful to have this goal:

[4] What counts as an 'implementation' of an algorithm, exactly? Some theoretical proposals require a very fine grained correspondence between the steps of the algorithm and the steps of the implementation, while other require less [65, 10]. For some extremely simple neural mechanisms in sensory and motor systems, we can begin to see how relevant computations could be implemented, sometimes all the way down to the molecular level [46, 24, 25, 89]. But it is not yet clear what to expect from an ideally completed psychological/neurophysiological theory of human language abilities and their realization. As Berwick and Weinberg have pointed out in their excellent discussion of the situation, we of course aim to characterize the relation as simply and straightforwardly as possible [7, §2]. But the implementation of programming languages is a rather large and complex story, and one expects the full story about the implementation of linguistic abilities to be even more so.

(M0) Disentangle what is computed from how it is computed, its realization.

Recursion matters in different ways for the specification of what is computed and for the performance model, since the recursion in the specification of what is computed often represents the recognition of simplifying regularities, but the optimal or psychologically appropriate computation may be flattened in the ways mentioned above. In fact, as (M0) suggests, we will see: (H2) performance models do in fact need to flatten where linguists don't. First, though, it is important to notice that there is significant agreement about some fundamental properties of the recursive mechanisms that define linguistic structures.

Although some of the early grammars and processing models were extremely powerful (e.g. Chomsky's 'Aspects' theory [70], and ATN processing models [93]), there were challenges to the idea that such powerful models were necessary [73], and in the late 1980's a surprising consensus began to emerge. It turned out that many independently proposed grammar formalisms were 'equivalent' in the sense that they could define exactly the same languages, including some non-context free languages, but remaining efficiently parsable. The proofs of these equivalences were generally rather easy, revealing some fundamental similarities among the recursive mechanisms. It later turned out that formal grammars inspired by Chomskian syntax, mentioned above, were also weakly equivalent to other independently proposed and well known formalisms. This yielded a surprising convergence, established in a number of papers, which can be summarized like this, writing 'CFG' for 'languages defined by context free grammars', 'TAG' for 'languages defined by tree adjoining grammars', 'CCG' for 'languages defined by combinatory categorial grammars', 'MG' for the 'languages defined by minimalist grammars', 'MCTAG' for 'languages defined by set-local multi-component tree adjoining grammars', 'MCFG' for 'languages defined by multiple context free grammars', 'CS' for 'languages defined by context sensitive grammars', 'RE' for 'recursively enumerable languages', 'Aspects' for 'languages defined by the transformational grammar of [70]', and 'HPSG' for 'languages defined by head-driven phrase structure grammar' [90,60,62, 33,79,70,6,37,86,87,35]:

**Thm 1.** CFG⊂ $\boxed{\text{TAG=CCG⊂MG=MCTAG=MCFG}}$ ⊂CS⊂RE=Aspects=HPSG.

The boxed language classes are, to use Joshi's [39] term, 'mildly context sensitive,' and Joshi proposes that human languages are in one of these. Following the results of Theorem 1, it was also discovered that many variants of minimalist grammars are equivalent: adding head movement (MGH), adding directional selection with 'head parameters' (DMG), adding asymmetrical feature checking (CMG), adding phases (PMG), adding sidewards movement (SMMG) [62,63,49,81,82]:

**Thm 2.** MG=MGH=DMG=CMG=PMG=RMG=SMMG.

All these grammars define the same class of mildly context sensitive languages. This consensus is extended to some optimality systems by recent

work [48,43,42,28] based on earlier studies of OT phonology [23,36]. This is the consensus (H1) mentioned earlier:

(H1) There is a significant consensus about the kinds of recursive operations needed to define human languages: they are 'mildly context sensitive'. (Joshi [39])

This claim still faces some challenges, but the challenges are rather minor in the sense that, even if they are successful, they require only a slight weakening of the claim, staying in the polynomially parsable class, properly included in the context sensitive languages.[5]

One respect in which minimalist grammars (MGs) are stronger than some similar, earlier proposals is that they allow a phrase to move after something has been extracted from it. This kind of 'remnant movement' has been proposed now for quite a wide range of constructions [92,66,44, 52,54,4,1]:

$[_{\text{VP}}\ t_1$ Gelesen$]_2$ hat [das Buch$]_1$ keiner $t_2$
read has the book noone

$[_{\text{AP}}$How likely $[t_1$ to win$]]_2$ is John$_1$ $t_2$?

$[_{\text{VP}}$Criticized by his boss $t_1]_2$ John$_1$ has never been $t_2$.

John $[_{\text{VP}}$reads $t_1]_2$ [no novels$]_1$ $t_2$.

Most earlier 'slash-passing' and 'store a filler, wait for a gap' parsing models cannot handle remnant movement, but standard parsing methods (CKY, Earley,...) have been shown sound, complete, efficient for the whole range of MGs [34], handling remnant movement and everything else that can be defined with this grammar formalism. Without remnant movement, the move and merge operations of MGs are strictly weaker, defining only context free languages [51].

## 3. Performance and variation

With Theorems 1, 2 and ongoing work that is still enlarging the scope of those claims, the 'linguistic wars' may not be over, but we can now see that, behind the endless skirmishes, a significant consensus lurks in the background. In many cases, we have automated recipes for converting grammars of one type into equivalent grammars of other types, so significant proposals expressed in one framework can often be adapted fairly easily to others. With this mind, let's consider how minimalist grammars (MGs) could be incorporated into a performance model. It is sometimes held that this is impossible, that MGs are unsuitable for performance models.[6] But since MGs participate in the the broadening consensus, this becomes implausible: the

[5] The remaining controversy concerns, for example, the status of copying-like operations in grammar, which threatens the 'linear growth' property [64,9,50], and the appropriate treatment of dependencies in scrambling [74,5,69].

[6] For example, Ferreira says in 2005,

ways of accounting for influences on human online syntactic analysis in one of these formalisms can be adapted to the others.

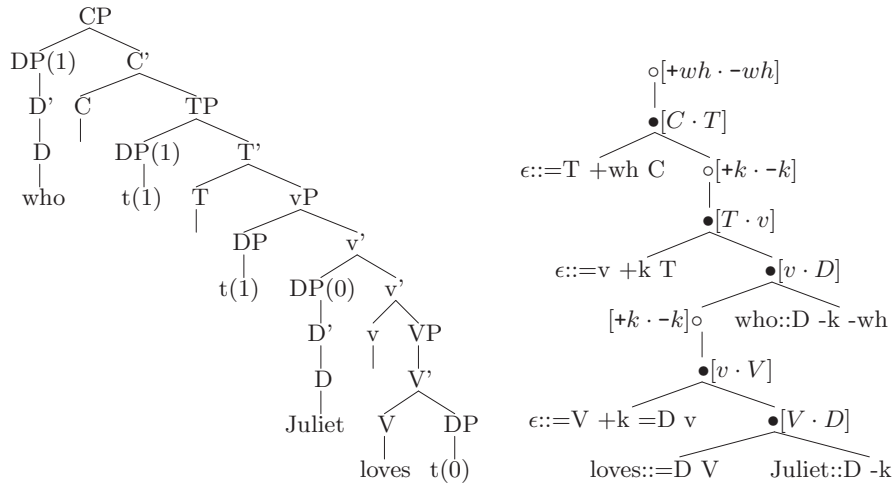
When an utterance is structurally ambiguous, human subjects frequently notice one structure before, or instead of, the others. Here we coarsely classify some of the proposals about 'first pass' parsing preferences, in order to point out briefly how all of these performance models must draw finer distinctions among derivational steps than the linguistic theory does. Consequently, even when the model computes exactly the structures defined by the grammar, the depth of depth of recursion for the computation is lower than the depth of recursion for the grammatical derivation.

*Categorial frequencies.* Let's say that a proposal about parsing preferences is *categorial* if it says one or another (partial) analysis is preferred because of the categories of the expressions involved. It is rather difficult to experimentally distinguish purely categorial influences from influences that are based on the particular lexical identities of elements of the same category, as discussed for example in [26, 41, §2.6], but many models of human recognition complexity and ambiguity resolution encode category-based preferences. This includes, for example, the models based on probabilistic context free grammars in [40, 30, 55, 56].

The merge steps of a minimalist derivation can be specialized for each category by marking each step with the features that trigger its application. This is achieved by mapping an MG with a particular lexicon into a strongly equivalent multiple context free grammar, as is done, in effect, by the conversion of an MG to a strongly equivalent MCFG [60].

> ... many psycholinguists are disenchanted with generative grammar. One reason is that the Minimalist Program is difficult to adapt to processing models. Another is that generative theories appear to rest on a weak empirical foundation... [I]t is now clear that no one interested in human performance can ignore the possible effects of things such as frequency and exposure on ease of processing. [19, pp.365, 370]

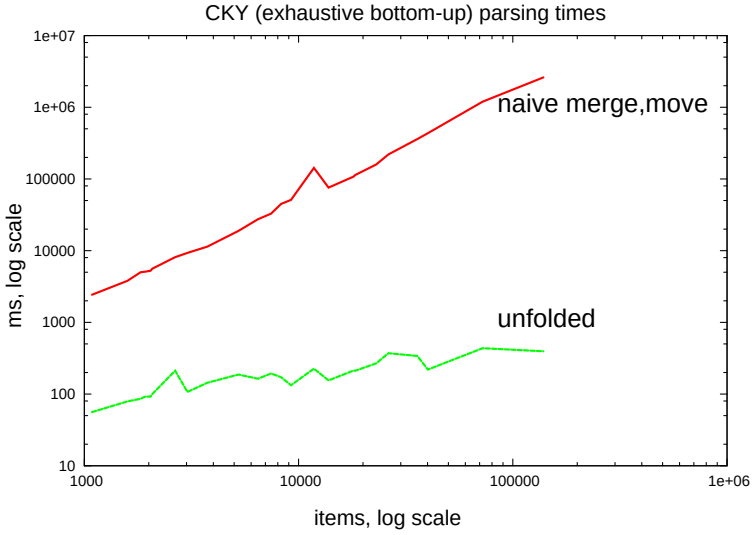
Old questions about the relation between grammar and performance [8, 21] are still alive today, prompted in particular by instances where simple grammatical sequences are not recognized as acceptable, and instances were ungrammatical sequences seem to be acceptable [20, 71, 72]. Here we aim only to defend the very general methodological and structural points already indicated: (M0,M1,H1,H2).




$\text{depth}_{\bullet[x,y]}\ 1,\ \text{depth}_{\circ[x,y]}\ 1$

The steps, now classified by the categories they operate on, can be assigned different probabilities in a manner exactly analogous to the assignment of probabilities to the rules of a context free grammar. This kind of step was taken by Hale [30, 29], for example, in a model of processing complexity.

This step, while it increases grammar size, can decrease processing time very significantly exactly because each step is more specialized. The following graph compares the processing times required to compute the same derived structures with a 'naive' implementation compared to one where each rule is specified for particular categories:




This graph plots processing time versus the number of 'items' computed by a CKY parser, just to illustrate how significant the effect can be on

a standard computing device. Notice that with the operations split into separate categories, the $\text{depth}_f$ for these new operations $f$ will tend to be significantly less in each analysis than the original $\text{depth}_\bullet$ and $\text{depth}_\circ$, but the output of the analysis is not changed in any way.

*Lexical frequencies.* More common in psychological models is a much finer tuning: not by category, but by particular lexical items [22,88,57]. It is familiar at least since Zipf [94] that some lexical items combine much more frequently than others in the same category. One simple computational model of this kind of preference is a 'bilexical' grammar which, in effect, treats each category as including the particular lexical item which is its 'head' in some sense of that term [58,67,17]. When this refinement is applied to MGs, in most discourses it becomes rare to see $\text{depth}_f$ greater than 1 for any (categorially and lexically specific) operation $f$.

*Left context and discourse effects.* Even more specific operations are obtained when we consider not just the pair of heads being combined, but the whole linguistic left context that has already been seen, together with the discourse situation. It is perhaps still controversial how much of this material really conditions first pass parsing preferences, but the evidence shows that it is applied quite quickly [15,2,12]. In parsing models where the full left context is available, it has been found useful in prediction [76,75, 83] – an unsurprising result!

*Modular perspectives.* The previous models were presented as involving refinements of the rules of grammar, so that more specialized rules can apply in more specialized contexts (to achieve exactly the same analysis). But these models need not be described (or implemented) that way. The reason for replacing the original operations $\bullet, \circ$ of the grammar with a large set much more specific $\bullet_C, \circ_C$ operations that apply only in particular contexts C is to allow for different probabilistic assessments of derivations in those different contexts. But obviously, this can be done without specializing the operations: we can calculate the probability of the derivations using the original operations in a model where those probabilities are conditioned by context. For example, as we take each step (with one of the original operations), we can obtain the probability of the extended partial derivation by multiplying the previous probability times the probability of this step, *in context*. This provides a ranking of the partial derivations at each step, which could be used for example in 'beam parsing' strategies that prune away the extremely improbable analyses incrementally [40,76, 75], or in one-parse-at-a-time strategies that pursue the most probable or most expedient option at each point [31]. In these models, the 'finest' classification of derivations comes not from the derivation steps, which are very general, but from the ranking steps, $\text{rank}_C$ which will assign each derivation a probability or weight in context.

Note that instead of having a large family of ranking functions $rank_C$ each specialized for some context $C$, we could treat the context $C$ as an argument to a very general ranking function. In fact, this is the usual perspective: the ranking function is given some representation of context $C$

and of the derivation $D$. So could depict a derivation tree with very general operations $\bullet, \circ, rank$ but then the leaves of rank would include not just linguistic material but also contexts, changing the nature of the comparison between this evaluation tree and the grammatical derivation trees. In any case, the effect of the context here is clearly to classify derivations very finely, along dimensions that matter to performance but not to the syntactic structural options. It is uncontroversial that this must be done, somehow, to model context effects on human sentence processing. The review of options here emphasizes the fact that such effects can be obtained with standard grammars, either by refining the operations of those grammars or by adding separate ranking steps. And of course all these steps could all be collapsed by 'optimizing' or 'practice' effects in other ways, analogous to methods used in compiling programming languages, without any change in what is computed.

## 4. Summary: How depth of recursion matters

The study of how linguistic structure is recognized takes place in the setting of significant agreement about what that structure is:

(H1) Human languages are mildly context sensitive.

When we consider any non-trivial computation, it is not hard to understand the importance of these basic methodological precepts:

(M0) Disentangle what is computed from how it is computed, its realization.

(M1) Even in the study of a single domain, there can be strong reasons to favor a highly recursive definition over a less recursive one, and vice versa.

Considering the study of human language recognition and production in particular,

(H2) Performance models need to flatten where linguists don't.

No reasonable linguist has argued that there are human languages with grammars that are non-recursive in the senses defined here. And there is very little controversy (mentioned in footnote 5) about whether the operations defining human languages are ones that define 'mildly context sensitive' languages. The controversy raised by Everett [18] and others concerns (interesting but) more superficial questions about derived structure: e.g. are there languages in which we never find clauses containing clauses?

(H2) is true simply because the recursion in the theory of language allows us to state regularities that are independent of many influences on the performance of linguistic tasks. It is hard to see how reasonable progress could be made without separating these issues, just as it is hard to see how to implement a complex high level program on a manufactured computer

without carefully separating and detailing the implementation. Adding non-structural factors preserves structure but hides it, reducing the depth of recursion and overwhelming us with detail unless it is approached systematically.

## References


1. Abels, K. Towards a restrictive theory of (remnant) movement: Improper movement, remnant movement, and a linear asymmetry. *Linguistic Variation Yearbook 2007 7* (2007), 53–120.
2. Altmann, G. T. M., van Nice, K. Y., Garnham, A., and Henstra, J.-A. Late closure in context. *Journal of Memory and Language 38*, 4 (1998), 459–484.
3. Appel, A. W. Unrolling recursions saves space. Technical report tr-363-92, Dept. of Computer Science, Princeton University, 1992.
4. Baltin, M. R. Movement to the higher V is remnant movement. *Linguistic Inquiry 33*, 4 (2002), 653–659.
5. Becker, T., Rambow, O., and Niv, M. The derivational generative power of formal systems, or, scrambling is beyond LCFRS. IRCS technical report 92-38, University of Pennsylvania, 1992.
6. Berwick, R. C. Computational complexity of lexical functional grammar. In *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics, ACL'81* (1981), pp. 7–12.
7. Berwick, R. C., and Weinberg, A. S. *The Grammatical Basis of Linguistic Performance: Language Use and Acquisition*. MIT Press, Cambridge, Massachusetts, 1984.
8. Bever, T. G. The cognitive basis for linguistic structures. In *Cognition and the Development of Language*, J. Hayes, Ed. Wiley, NY, 1970.
9. Bhatt, R., and Joshi, A. Semilinearity is a syntactic invariant: a reply to Michaelis and Kracht. *Linguistic Inquiry 35* (2004), 683–692.
10. Blass, A., Dershowitz, N., and Gurevich, Y. When are two algorithms the same? *Bulletin of Symbolic Logic 15*, 2 (2009), 145–168.
11. Buss, S. R. On gödel's theorems on lengths of proofs i: Number of lines and speedup for arithmetics. *Journal of Symbolic Logic 59*, 2 (1994), 737–756.
12. Chambers, C. G., Tanenhaus, M. K., Eberhard, K. M., Filip, H., and Carlson, G. N. Actions and affordances in syntactic ambiguity resolution. *Journal of Experimental Psychology: Learning, Memory and Cognition 30*, 3 (2004), 687–696.
13. Chitil, O. Type inference builds a short cut to deforestation. In *International Conference on Functional Programming* (1999), pp. 249–260.
14. Chomsky, N. *The Minimalist Program*. MIT Press, Cambridge, Massachusetts, 1995.
15. Crain, S., and Steedman, M. On not being led up the garden path. In *Natural Language Parsing*, D. Dowty, L. Karttunen, and A. Zwicky, Eds. Cambridge University Press, NY, 1985.
16. den Besten, H., and Webelhuth, G. Stranding. In *Scrambling and Barriers*, G. Grewendorf and W. Sternefeld, Eds. Academic Press, NY, 1990.
17. Eisner, J. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in Probabilistic and Other Parsing Technologies*, H. Bunt and A. Nijholt, Eds. Kluwer Academic Publishers, October 2000, pp. 29–62.
18. Everett, D. L. Cultural constraints on grammar and cognition in Piraha: Another look at the design features of human language. *Current Anthropology 46* (2005), 621–646.
19. Ferreira, F. Psycholinguistics, formal grammars, and cognitive science. *Linguistic Review 22* (2005), 365–380.

20. Ferreira, F., and Patson, N. D. The 'good enough' approach to language comprehension. *Language and Linguistics Compass 1* (2007), 71–83.
21. Fodor, J. A., Bever, T. G., and Garrett, M. F. *The Psychology of Language: An Introduction to Psycholinguistics and Generative Grammar*. McGraw-Hill, NY, 1976.
22. Ford, M., Bresnan, J., and Kaplan, R. M. A competence-based theory of syntactic closure. In *The Mental Representation of Grammatical Relations*, J. Bresnan, Ed. MIT Press, Cambridge, Massachusetts, 1982.
23. Frank, R., and Satta, G. Optimality theory and the generative complexity of constraint violability. *Computational Linguistics 24* (1998), 307–315.
24. Franosch, J.-M. P., Lingenheil, M., and van Hemmen, J. L. How a frog can learn what is where in the dark. *Physical Review Letters 95*, 7 (2005), 078106.
25. Friedel, P. *Sensory Information Processing: Detection, Feature Extraction, and Multimodal Integration*. PhD thesis, Technical University of Munich, 2008.
26. Gibson, E., and Pearlmutter, N. J. Constraints on sentence processing. *Trends in Cognitive Science 2* (1998), 262–268.
27. Gödel, K. Uber die Länge von Beweisen. *Ergebnisse eines Mathematischen Kolloquiums* (1936), 23–24. English translation in *Kurt Gödel: Collected Works, Volume 1*. Oxford University Press, NY, 1986, pp. 396-399.
28. Graf, T. Reference-set constraints as linear tree transductions via controlled optimality systems. In *Proceedings of the Conference on Formal Grammar, ESSLLI'10* (2010).
29. Hale, J. *Grammar, Uncertainty, and Sentence Processing*. PhD thesis, Johns Hopkins University, 2003.
30. Hale, J. Uncertainty about the rest of the sentence. *Cognitive Science 30*, 1 (2006), 609–642.
31. Hale, J. T. What a rational parser would do. *Cognitive Science 35*, 3 (2011), 399–443.
32. Hamilton, G. Higher order deforestation. *Fundamenta Informaticae 69*, 1-2 (2006), 39–61.
33. Harkema, H. A characterization of minimalist languages. In *Logical Aspects of Computational Linguistics* (NY, 2001), P. de Groote, G. Morrill, and C. Retoré, Eds., Lecture Notes in Artificial Intelligence, No. 2099, Springer, pp. 193–211.
34. Harkema, H. *Parsing Minimalist Languages*. PhD thesis, University of California, Los Angeles, 2001.
35. Jaeger, E., Francez, N., and Wintner, S. Guaranteeing parsing termination of unification grammars. *Journal of Logic, Language and Information 14*, 2 (2005), 199–234.
36. Jäger, G. Gradient constraints in finite state OT: The unidirectional and the bidirectional case. In *More than Words: A Festschrift for Dieter Wunderlich*, J. Greenberg, Ed. Akademie Verlag, Berlin, 2002, pp. 299–325.
37. Johnson, M. *Attribute Value Logic and The Theory of Grammar*. No. 16 in CSLI Lecture Notes Series. CSLI Publications, Chicago, 1988.
38. Jones, N. D., Gomard, C. K., and Sestoft, P. *Partial Evaluation and Automatic Program Generation*. Prentice-Hall, Englewood Cliffs, New Jersey, 1993.
39. Joshi, A. How much context-sensitivity is necessary for characterizing structural descriptions. In *Natural Language Processing: Theoretical, Computational and Psychological Perspectives*, D. Dowty, L. Karttunen, and A. Zwicky, Eds. Cambridge University Press, NY, 1985, pp. 206–250.
40. Jurafsky, D. A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science forthcoming* (1996).

41. Jurafsky, D. Probabilistic modeling in psycholinguistics: Comprehension and production. In *Probabilistic Linguistics*, R. Bod, J. Hay, and S. Jannedy, Eds. MIT Press, Cambridge, Massachusetts, 2003, pp. 39–96.
42. Kanazawa, M. A pumping lemma for well-nested multiple context free grammars. In *13th International Conference on Developments in Language Theory, DLT 2009* (2009).
43. Kanazawa, M., and Salvati, S. Generating control languages with abstract categorial grammars. In *Proceedings of the 12th conference on Formal Grammar (FG'07)* (Stanford, California, 2007), L. Kallmeyer, P. Monachesi, G. Penn, and G. Satta, Eds., CLSI Publications.
44. Kayne, R. S. Overt vs. covert movment. *Syntax 1*, 2 (1998), 128–191.
45. Keenan, E. L., and Stabler, E. P. *Bare Grammar: Lectures on Linguistic Invariants*. CSLI Publications, Stanford, California, 2003.
46. Kempter, R., Leibold, C., Wagner, H., and van Hemmen, J. L. Formation of temporal-feature maps by axonal propagation of synaptic learning. *Proceedings of the National Academy of Sciences of the United States of America 98*, 7 (2001), 4166–4171.
47. Kennedy, K., and Allen, R. *Optimizing Compilers for Modern Architectures: A Dependence-based Approach*. Morgan Kaufmann, San Mateo, California, 2001.
48. Kepser, S., and Mönnich, U. Properties of linear context free tree languages with an application to optimality theory. *Theoretical Computer Science 354* (2006), 82–97.
49. Kobele, G. M. Formalizing mirror theory. *Grammars 5* (2002), 177–221.
50. Kobele, G. M. *Generating Copies: An Investigation into Structural Identity in Language and Grammar*. PhD thesis, UCLA, 2006.
51. Kobele, G. M. Without remnant movement, mgs are context-free. In *Proceedings of the 11th Mathematics of Language (MOL)* (2009).
52. Koopman, H., and Szabolcsi, A. *Verbal Complexes*. MIT Press, Cambridge, Massachusetts, 2000.
53. Kühnemann, A. Comparison of deforestation techniques for functional programs and for tree transducers. In *Fuji International Symposium on Functional and Logic Programming* (1999), pp. 114–130.
54. Lee, F. VP remnant movement and VSO in Quiaviní Zapotec. In *The Syntax of Verb Initial Languages*, A. Carnie and E. Guilfoyle, Eds. Oxford University Press, Oxford, 2000.
55. Levy, R. Expectation-based syntactic comprehension. *Cognition 106* (2008), 1126–1177.
56. Levy, R., Reali, F., and Griffiths, T. Modeling the effects of memory on human online sentence processing with particle filters. Proceedings of the Twenty-second Annual Conference on Neural Information Processing Systems, 2009.
57. MacDonald, M., Pearlmutter, N. J., and Seidenberg, M. S. Syntactic ambiguity as lexical ambiguity resolution. In *Perspectives on Sentence Processing*, C. Clifton, L. Frazier, and K. Rayner, Eds. Lawrence Erlbaum, Hillsdale, New Jersey, 1994, pp. 155–180.
58. Magerman, D. M. *Natural Language Parsing as Statistical Pattern Recognition*. PhD thesis, Stanford University, 1994.
59. Marlow, S., and Wadler, P. Deforestation for higher-order functions. In *Functional Programming* (1992), pp. 154–165.
60. Michaelis, J. Derivational minimalism is mildly context-sensitive. In *Proceedings, Logical Aspects of Computational Linguistics, LACL'98* (NY, 1998), Springer, pp. 179–198.
61. Michaelis, J. *On Formal Properties of Minimalist Grammars*. PhD thesis, Universität Potsdam, 2001. *Linguistics in Potsdam 13*, Universitätsbibliothek, Potsdam, Germany.

62. Michaelis, J. Transforming linear context free rewriting systems into minimalist grammars. In *Logical Aspects of Computational Linguistics* (NY, 2001), P. de Groote, G. Morrill, and C. Retoré, Eds., Lecture Notes in Artificial Intelligence, No. 2099, Springer, pp. 228–244.
63. Michaelis, J. Notes on the complexity of complex heads in a minimalist grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks, TAG+6* (2002), pp. 57–65.
64. Michaelis, J., and Kracht, M. Semilinearity as a syntactic invariant. In *Logical Aspects of Computational Linguistics* (NY, 1997), C. Retoré, Ed., Springer-Verlag (Lecture Notes in Computer Science 1328), pp. 37–40.
65. Moschovakis, Y. N. What is an algorithm? In *Mathematics unlimited – 2001 and beyond*, B. Engquist and W. Schmid, Eds. Springer, NY, 2001, pp. 919–936.
66. Müller, G. *Incomplete Category Fronting*. Kluwer, Boston, 1998.
67. Nederhof, M.-J., and Satta, G. Left-to-right parsing and bilexical context-free grammars. In *Proceedings of ANLP-NAACL 2000* (2000).
68. Parberry, I. Circuit complexity and feedforward neural networks. In *Mathematical Perspectives on Neural Networks*, P. Smolensky, M. C. Mozer, and D. Rumelhart, Eds. Erlbaum, Mahwah, New Jersey, 1996.
69. Perekrestenko, A. Minimalist grammars with unbounded scrambling and nondiscriminating barriers are NP-hard. In *Proceedings of the 2nd International Conference on Language and Automata Theory and Applications, LATA2008* (Berlin, 2008), C. Martín-Vide, F. Otto, and H. Fernau, Eds., Lecture Notes in Computer Science 5196, Springer-Verlag, pp. 421–432.
70. Peters, P. S., and Ritchie, R. W. On the generative power of transformational grammar. *Information Sciences 6* (1973), 49–83.
71. Phillips, C., and Wagers, M. Relating time and structure in linguistics and psycholinguistics. In *Oxford Handbook of Psycholinguistics*, G. Gaskell, Ed. Oxford University Press, Oxford, 2007, pp. 739–756.
72. Phillips, C., Wagers, M., and Lau, E. Grammatical illusions and selective fallibility in real-time language comprehension. *Language and Linguistics Compass forthcoming* (2010).
73. Pullum, G. K., and Gazdar, G. Natural languages and context free languages. *Linguistics and Philosophy 4* (1982), 471–504.
74. Rambow, O. *Formal and Computational Aspects of Natural Language Syntax*. PhD thesis, University of Pennsylvania, 1994. Computer and Information Science Technical report MS-CIS-94-52 (LINC LAB 278).
75. Roark, B. Probabilistic top-down parsing and language modeling. *Computational Linguistics 27*, 2 (2001), 249–276.
76. Roark, B. Robust garden path parsing. *Natural Language Engineering 10*, 1 (2004), 1–24.
77. Scheifler, R. W. An analysis of inline substitution for a structured programming language. *Communications of the Association for Computing Machinery 20* (1977), 647–654.
78. Seidl, H., and Sørensen, M. H. Constraints to stop higher-order deforestation. In *ACM Symposium on Principles of Programming Languages* (1997), ACM Press, pp. 400–413.
79. Seki, H., Matsumura, T., Fujii, M., and Kasami, T. On multiple context-free grammars. *Theoretical Computer Science 88* (1991), 191–229.
80. Smolensky, P., and Legendre, G. *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar, Volume I: Cognitive Architecture*. MIT Press, Cambridge, Massachusetts, 2006.
81. Stabler, E. P. Derivational minimalism. In *Logical Aspects of Computational Linguistics*, C. Retoré, Ed. Springer-Verlag (Lecture Notes in Computer Science 1328), NY, 1997, pp. 68–95.

82. Stabler, E. P. Computational perspectives on minimalism. In *Oxford Handbook of Minimalism*, C. Boeckx, Ed. Oxford University Press, Oxford, 2010, pp. 617–641.
83. Stabler, E. P. Top-down recognizers for MCFGs and MGs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics (CMCL), 49th Annual Meeting of the Association for Computational Linguistics* (2011), F. Keller and D. Reitter, Eds.
84. Stabler, E. P., and Keenan, E. L. Structural similarity. *Theoretical Computer Science 293* (2003), 345–363.
85. Statman, R. *Structural Complexity of Proofs*. PhD thesis, Stanford University, 1974.
86. Torenvliet, L., and Trautwein, M. A note on the complexity of restricted attribute-value grammars. In *Proceedings of Computational Linguistics In the Netherlands, CLIN5* (Twente, The Netherlands, 1995), M. Moll and A. Nijholt, Eds., Department of Computer Science, University of Twente, pp. 145–164.
87. Trautwein, M. The complexity of structure-sharing in unification-based grammars. In *Proceedings of Computational Linguistics In the Netherlands, CLIN5* (1995), pp. 165–180.
88. Trueswell, J. The role of lexical frequency in syntactic ambiguity resolution. *Journal of Memory and Language 35* (1996), 566–585.
89. van Hemmen, J. L., and Schwartz, A. B. Population vector code: a geometric universal as actuator. *Biological Cybernetics 98*, 6 (2008), 509–518.
90. Vijay-Shanker, K., Weir, D., and Joshi, A. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics* (1987), pp. 104–111.
91. Wadler, P. Deforestation: Transforming programs to eliminate trees. *Theoretical Computer Science 73*, 2 (1990), 231–248.
92. Webelhuth, G., and den Besten, H. Adjunction and remnant topicalization in the germanic SOV-languages. Paper presented at the GLOW conference, Venice, 1987.
93. Woods, W. A. Transition network grammars for natural language analysis. *Communications of the Association for Computing Machinery 13*, 10 (1970), 591–606.
94. Zipf, G. K. *The Psychobiology of Language: An introduction to dynamic philology*. Houghton-Mifflin, Boston, 1935.