Universal supergrammar: *231 in neutral word order

David P. Medeiros

September 2021

Abstract

I present a model of neutral word order options in natural language. The core proposal is that languages share a universal base structure with inherent head-complement-specifier linear order, and a universal "supergrammar" maps this underlying order to the set of typologically possible information-neutral surface orders, consisting of its stack-sortable (231-avoiding) permutations. The mapping procedure can be formulated either as a parsing algorithm based on stack-sorting, or as a generative model involving postorder and preorder traversals of freely-generated n-ary branching trees. This single-principle universal grammar explains and unifies some well-known word order universals, while also generating phenomena that challenge traditional approaches. Applications include Cinque's version of Greenberg's Universal 20, the Final-Over-Final Condition, a modified Head Movement Constraint allowing attested long head movement, English Affix Hopping, Germanic cross-serial subject-verb dependencies, and Icelandic Stylistic Fronting.

1.0 Introduction

In this paper, I explore the consequences of the following proposal about neutral word order possibilities in natural language.

> (1) a. Languages share a common base syntactic structure, with a universal underlying linear order. Supposing that this base structure can be analyzed in terms of the X-bar theoretic (Chomsky 1970, Jackendoff 1977, Stowell 1981) notions of head, complement, and specifier, the universal order of the base is *head-complement-specifier*.
>
> b. Across languages, possible information-neutral surface word orders are the stack-sortable (*i.e.*, 231-avoiding) permutations of the universal base order.

It is important to point out that we are only aiming for an account of information-neutral word order possibilities, explicitly setting aside A-bar movement, wh-fronting, topic, focus, etc. In the present proposal, a mapping procedure relates the base not to a single surface output, but to a set of outputs. As strings, these surface outputs are exactly the stack-sortable permutations of the underlying base order, counted by the Catalan numbers (1, 2, 5, 14, 42, etc), growing much more slowly than the set of all possible permutations, counted by the factorial function $n!$ (1, 2, 6, 24, 120, etc). These orders can also be characterized as the 231-avoiding permutations of the underlying base order, considered as the identity permutation i = 123... (A 231-avoiding permutation is an order of elements in i not containing subsequence *...b...c...a..., for any subsequence ...a...b...c... in i).

This mapping can be formulated in two equivalent ways: as a universal parsing algorithm, based on stack-sorting (Medeiros 2018), or as a competence-level generative

procedure involving traversals of freely-generated, n-ary branching, CS-style trees (Medeiros 2021). The mapping determines a unique bracketed structure for each permissible surface order, which corresponds in detail to standard accounts of the relevant surface structure/derivation.

The set of surface outputs is generally larger than what is realized in any given language. But crucially, not all orders can be generated; the orders which cannot be generated this way account for a range of word order universals that were previously viewed as unrelated. This includes Cinque's (2005) version of Universal 20 (Greenberg 1963), the Final-Over-Final Condition (Holmberg 2000, Biberauer *et al* 2014, Sheehan *et al* 2017, *i.a.*), and a version of the Head Movement Constraint (HMC; Travis 1984) that allows attested Long Head Movement (LHM; Rivero 1991, Lema & Rivero 1991, *i.a.*). Beyond ruling out universally-forbidden word order patterns, the account also successfully generates syntactic phenomena that are problematic for many current syntactic theories, including Affix Hopping (Chomsky 1957), and cross-serial subject-verb dependencies in Dutch (Huybregts 1976, Bresnan *et al* 1982) and Swiss German (Shieber 1985). Finally, we discover an intriguing account of the possibilities and restrictions of Stylistic Fronting (SF; Maling 1980/1990; Jónsson 1991; Holmberg 2000, 2006; Ott 2018, *i.a.*) in Icelandic, including its Subject-Gap Restriction and Accessibility Hierarchy.

What is remarkable is that all of the above fall out as immediate consequences of (1), without invoking any further constraints or mechanisms, within or across languages. This is a considerable simplification of the mix of existing constraints and mechanisms covering the same empirical terrain within existing theories. Beyond unifying the technical apparatus, we also find improved empirical coverage. For example, we derive a novel generalization about where the HMC can be violated (i.e., where LHM is possible) that has not been noticed before, and appears to be exceptionless.

1.1 What this paper is not about

Before moving on, it worth saying something about some important matters that must be set aside. As mentioned above, the goal is a theory of information-neutral word order possibilities across languages. This means, first, that I will have nothing to say about many important syntactic phenomena like wh-movement, topicalization, focus movement, and the like. This can be taken as an implicit endorsement of Chomsky's ideas about tying the Duality of Semantics to some fundamental distinction in the workings of syntax.[1] That said, obviously a more complete theory would spell out the nature of this other side of syntax, a task I leave for future work.

The second aspect of this proposal worth pointing out is that the theory is mute on the relationship between the possibilities afforded by the universal "supergrammar" and the more restricted word order possibilities within individual languages. One strong reading of the

[1] Chomsky has proposed that the distinction between information-neutral structure and discourse-information effects is tied to his distinction between External Merge (EM) and Internal Merge (IM): "The two types of Merge correlate well with the duality of semantics that has been studied from various points of view over the years. EM yields generalized argument structure, and IM all other semantic properties: discourse-related and scopal properties. The correlation is close, and might turn out to be perfect if enough were understood." (Chomsky 2007: 10)

proposal is that the system described here is an adequate description of the basic cognitive system underlying the processing of neutral order for any language, and that differences between languages must be attributed to something else. One plausible source of such differences is predictive learning by language-acquiring children of the subset of universally-available patterns their language happens to exploit. On such a view, learning the grammar of individual languages would amount to "learning by forgetting": humans are equipped at birth with an invariant cognitive system that encompasses the grammatical possibilities of all languages at once, and learning one language or another reduces to reinforcing only a subset of the richer universal possibilities. While that may be a reasonable direction to pursue, this too is a topic for another work, and I keep here to a focus on proposal (1) and its narrow consequences for word order.

Finally, one may wonder about the status of Merge in this proposal. Obviously, the relationship between underlying structure and surface structure is rather different here than in familiar theories. Nevertheless, as in any generative proposal, one needs a basic combinatory mechanism to build the underlying constituent structure, and this corresponds more or less to the standard concept of External Merge. However, a crucial claim here is that the base structure is universally linearly ordered, in head-complement-specifier order, *contra* Kayne (1994). One could, I suppose, continue to insist that Merge is inherently unordered set formation, in which case we would have to introduce an internal, base-level linearization operation, separate from the mapping procedure which relates the base to various possible surface word orders.

It seems more reasonable to me to suppose that the relevant implementation of Merge is inherently asymmetric. This is especially so for the parsing-based view I describe in which, as we will see, a pre-processing step of stack-sorting rearranges surface word orders into an invariant output, which is then fed to a universal Shift-Reduce semantic parser. In this way of thinking about things, the initial stack-sorting step does the work of Internal Merge; that is, it is the transformational component. The invariant Shift-Reduce parser handles External Merge, corresponding to the base component. This, of course, revives the old observation that a phrase structure grammar matches the actions of a Shift-Reduce stack machine. The Reduce step corresponds closely to the notion of External Merge, building a higher-order non-terminal node from a contiguous sequence of categories on top of the stack that match a phrase structure rule. The operands of Merge then have a definite linear order, corresponding to their order in the stack, which provides a computational basis for an asymmetry in Merge. Yet again, though, a fuller discussion would take us too far afield. And identifying syntactic operations with a parsing algorithm likewise raises some difficult questions; for this reason, I provide an alternative implementation of the theory as a competence-level generative account.

1.2 Structure of the paper

The remainder of this paper is structured as follows. In section 2, I sketch some conceptual preliminaries, defining stack-sorting, permutations, subsequences, and permutations avoiding a forbidden subsequence, and showing how the dynamics of stack-sorting induce implicit bracketed surface structure. In section 3, I discuss Cinque's (2005) version of Greenberg's Universal 20, showing that we derive exactly the same predictions in the present account. This includes not just the same possible and impossible orders, but nearly-identical

bracketed structures. I also discuss the apparent counter-examples to Cinque's typology in Shupamem raised by Nchare (2012), showing that all orders in that language that fall outside Cinque's typology involve focus, and are thus irrelevant to the generalization about neutral orders pursued here. Section 4 takes up the Final-Over-Final Condition (FOFC), showing that the present account derives every major case of FOFC as a consequence of our *231 theorem.

In section 5, I discuss Travis' (1984) Head Movement Constraint, showing that the core data taken to support the HMC is predicted by this account. At the same time, known exceptions to the HMC, in the form of attested long head movement patterns, are also shown to be generated by this system, obeying a novel generalization concerning the linear position of the subject. Section 6 presents our treatment of cross-serial relations in surface order, including cross-serial subject-verb dependencies in Dutch and Swiss German, showing that these patterns too are readily explained in the present framework, without recourse to additional operations or constraints. I also provide an analysis of English Affix Hopping, an apparently problematic case of head lowering, where we reconstruct something remarkably like Chomsky's (1957) classic transformational analysis. Section 7 provides an account of Icelandic Stylistic Fronting, an apparently "exotic" movement process affecting both heads and phrases, and governed by an intricate Accessibility Hierarchy. I show that the present account extends without modification to this phenomenon as well, readily explaining many of its details. Section 8 discusses some further refinements and extensions of the basic framework, confronting the issue of cycles. Section 9 presents an alternative conceptualization of the framework as a generative system involving tree traversals; the final section summarizes and sketches some broad conclusions.

2.0 Some formal preliminaries

This section presents some formal background and definitions that underpin what follows. Important notions include permutations, forbidden permutations, and stack-sorting.

2.1 Some definitions

(2) *Permutation* Given a set of symbols, a permutation is a sequential arrangement of those symbols. Given a reference sequence taken as the identity permutation, we can describe other permutations of the same set of elements perspicuously in terms of a numerical sequence, where the identity permutation is the sequence 1 2 3 4 5... . In what follows, the identity permutation is a language-invariant representation of the underlying hierarchy of an expression, and we will be considering surface orders as permutations of this basic sequence.

(3) *Index*. The number associated with a position in the identity permutation, which we will use to refer to the relative positions of lexical elements within the invariant base order.

(4) *Subsequence* Given a sequence of symbols (a permutation), a subsequence is any linear arrangement of a subset of symbols from that sequence that preserves their relative linear order. For example, the permutation 1 2 5 4 3 contains 2 4 as a subsequence.

(5) *Forbidden permutation* This work characterizes unavailable word orders in terms of a subsequence contour within their surface order (namely, *231). This condition does not refer to any three specific lexical items or hierarchical positions. Rather, given a linearly-ordered representation of the hierarchy as the identity permutation, we rule out any surface order permutation that contains a subsequence bca, where a < b < c in the identity order. To be clear, this condition does not require adjacency of the elements in either the surface order or in the identity permutation/underlying hierarchical order. So, for example, given the identity permutation 1 2 3 4 5 6 ..., the order 4 1 6 2 3 5 contains the forbidden *231 contour (its subsequences 4 6 2 and 4 6 3 have mid-high-low index pattern).

(6) *Base order* A base order is the identity permutation of the lexical elements in an expression. The idea is that expressions in different languages with commensurable lexical content share the same underlying hierarchical representation. In the present framework, this representation is a particular permutation of the elements involved. For example, the base order for a transitive clause (motivated in section 4) is C < Pol < T < V < O < Adv < S. Thus, C Pol T V O Adv S is the identity permutation 1 2 3 4 5 6 7; C has index 1, Pol has index 2, etc. Clause orders in different languages are different permutations of this underlying base order. The English clause *...that they didn't often eat cake* arranges elements in the order C S T Pol Adv V O (taking the Neg head *-n't* to instantiate the Pol position; see below). In terms of numerical indices, the English clause is the permutation 1 7 3 2 6 4 5. This permutation is 231-free; none of its subsequences form the forbidden *231 contour.

(7) *Legal bracketing* A legal bracketing is a string consisting of left and right brackets meeting the following two conditions: (i) at each position in the string, the number of preceding right brackets may not exceed the number of preceding left brackets, and (ii) at the end of the string, the number of left brackets and right brackets are equal. So, for example, '( ) ( )' and '((( )))' are legal bracketings, while ') (' and '(( )' are not (failing condition (i) and (ii), respectively). The number of these strings of fixed length is a number from the Catalan sequence (1, 1, 2, 5, 14, 42, 132, ...). With no bracket pairs, we have one legal bracketing (the empty string); there is one choice for one pair of brackets, two choices for two pairs, 5 for three pairs, 14 for four pairs, etc.

2.2 Stack-sorting

I present a slightly edited version of the stack-sorting algorithm discussed in Medeiros (2018), itself a minor variant of Knuth's classic algorithm.[2] The algorithm (8) processes an input

[2] The important change here reflects a choice to represent the desired output -- the identity permutation, in present terms -- as an increasing sequence 123... . In Medeiros (2018), the relevant nominal hierarchy [[[[N] AdjP] NumP] DemP] was instead numbered 4321. One consequence of the convention in that paper is that the forbidden permutation is then characterized as *213, rather than *231. The increasing index sequence adopted here is more in line with conventions in other works on permutations and related computer science notions.

sequence into an output sequence, by means of two operations: Push and Pop, which take elements one at a time from input to stack, and thence from stack to output, respectively.

(8) Stack-Sorting Algorithm

While input is non-empty,
    If S < I, Pop.
    Else Push.
While Stack is non-empty,
    Pop.

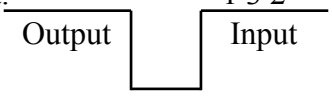
(9) Definitions for (8)

I: next item in input.
S: item on top of stack.
x < y: *x* precedes *y* in the base order.
Push: move I from input onto stack.
Pop: move S from stack to output.

To illustrate how stack-sorting works, let's walk through how the algorithm applies to some of the permutations of the identity 123. Our goal is to sort any surface order into 123 output; as we will see; this succeeds for five of the six order permutations, but fails for one, 231.

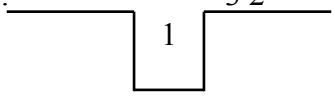
Stack-sorting is often represented as in the following diagram. The input sequence (here, 132) begins on the right; its elements are "pushed" onto the stack one at a time, interleaved with operations "popping" elements one at a time to form the output.

(10) a.



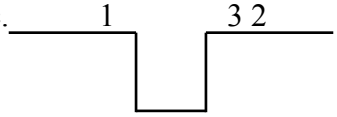
Start: input 132 at right
    Input is non-empty...
        There is no S, so (S < I) is False...
            So: Push (1)

b.



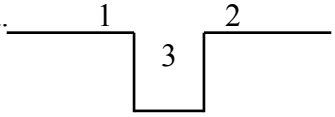
Push 1 onto the stack
    Input non-empty
        S < I: True (1 < 3)
            So: Pop (1)

c.



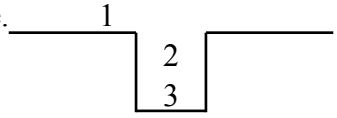
Pop 1 from the stack
    Input non-empty
        No S element; S < I False
            So: Push (3)

d.

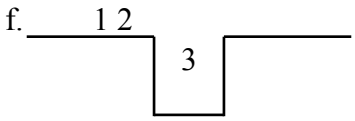


Push 3 onto the stack
    Input non-empty
        S < I: False (3 > 2)
            So: Push (2)

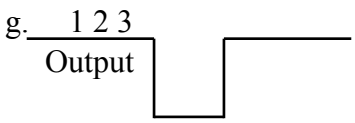
e.



Push 2 onto the stack
    Input empty; stack non-empty
        So: Pop(2)



Pop 2 from the stack
Input empty; stack non-empty
So: Pop(3)

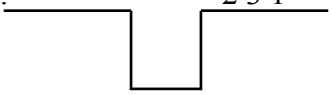


Pop 3 from the stack
Input and Stack empty

In this way, we stack-sort the input permutation 132 into the desired output, the identity permutation 123. Stack-sorting can successfully rearrange many different permutations into the underlying identity. For example, if the input is 123, identical to the desired output, each element is pushed and immediately popped. If the input is the mirror image of the output, 321, the entire sequence is first pushed onto the stack, and then popped, reversing its order.

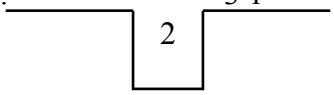
Importantly, however, not all sequences can be stack-sorted. Among the six possible permutations of 123, five can be stack-sorted (123, 132, 213, 321, 312) , but one cannot: 231. The following example illustrates why stack-sorting fails for this input sequence.

(11) a.



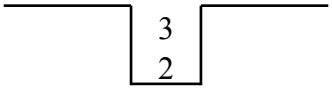
Start: input 231 at right

b.



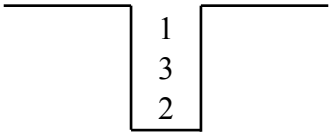
Push 2 onto the stack

c.



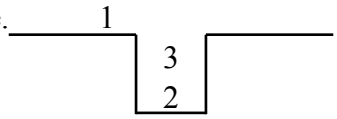
Push 3 onto the stack

d.



Push 1 onto the stack

e.



Pop 1 from the stack

At this point, the problem with this order becomes clear. We wish to assemble the output in the order 123. However, after we have popped the 1, the element on top of the stack is 3. We would have to somehow reach deeper into the stack to retrieve the 2 before popping the 3; this is not allowed. Instead, with the input emptied, the algorithm dictates that the top-of-stack element

3 must be Popped to the output, followed by Pop of the remaining element 2 in the stack. This produces the illicit output order 132.

2.3 Putting things together

With these initial definitions in place, we can now elaborate somewhat on the content of what follows. As stated at the outset, the proposal here is that typologically possible neutral word orders are the stack-sortable permutations of the base order, taken as the identity permutation. Put another way, possible neutral word orders are the 231-avoiding permutations of the base head-complement-specifier order.

As pointed out in Medeiros (2018), it is rather natural to read the Push and Pop operations of stack-sorting as left and right brackets, respectively, labeled by the lexical element they affect (*e.g.*, Push(N) ~ $[_N$; Pop(N) ~ $]_N$). Reading Pushes and Pops as left and right brackets in this way, something remarkable emerges:

(12) The set of stack-sortable (231-avoiding) permutations of a given base order corresponds to the set of all legal bracketings, for a fixed number of bracket pairs.

That is, the stack-sortable permutations exhaust the set of legal Push and Pop sequences, which correspond to all legal ways of pairing a given number of left and right brackets. This invites a way of thinking in which the legal bracketings are generated directly as a first step, with labels from a base order then written onto right brackets, and word order then read from corresponding left brackets. In effect, this is the tree traversal view we will develop near the end of the paper.

For now, though, the important takeaway is that we can represent Push and Pop sequences, as dictated by the stack-sorting algorithm, as a sequence of labeled brackets. As we will see when we turn to Universal 20 in the following section, these bracketings are not arbitrary, but correspond in detail to the bracketings formed by standard move-and-merge derivations of the relevant orders. Nevertheless, they are also somewhat simpler, losing some superfluous structural layers, and allowing *n*-ary branching. This is an intriguing result, since the stack-sorting procedure is not merely a notational variant of the standard conception of derivations. Of particular note, from the parsing perspective, is that rather than performing movements to disrupt an initial uniform base structure, we are in a sense performing "anti-movement" to reconstruct the base from a given surface order. Significantly, we do not need order-particular instructions to accomplish this; the language-invariant stack-sorting algorithm (8) does the job for any attested orders, but only for those, explaining at the same time why certain orders are typologically impossible.

With this in hand, we can now spell out the kind of architecture we have in mind. In broad strokes, this will be cast as a two-stage parsing device. In the initial pre-processing stage, the language-invariant stack-sorting algorithm (8) converts all and only the stack-sortable (231-avoiding) word order permutations into a unique output order, representing the base structure. This base order is then input to a Shift-Reduce semantic parser, which implements a universal

phrase structure grammar, building the familiar kind of constituent structure from the base order. Figure 1 below summarizes the intended action of the architecture.

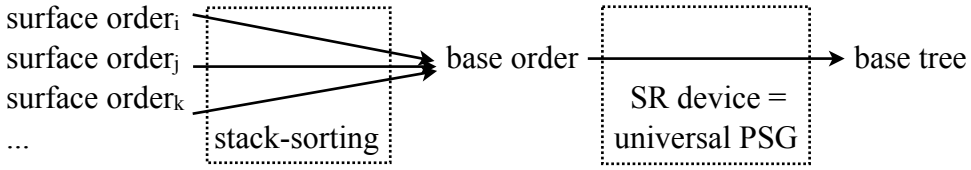


Figure 1. The architecture of the proposal as a parsing device. The invariant stack-sorting algorithm sorts any typologically possible word orders into a unique, universal base order. The base order is then fed to a Shift-Reduce semantic parser, implementing a language-invariant phrase structure grammar, which builds the familiar constituent structure.

Intriguingly, we can maintain that the structure at the base level is entirely context-free; it is the addition of the stack-sorting pre-processing step that allows this system to handle attested orders exhibiting cross-serial dependencies and discontinuous constituency. As a simple example, consider an input containing the common clause order VSO. This order famously presents a problem to constituent-based analyses of surface order; we want the V and the O to form a VP constituent, just as in other languages, yet the S intervenes between them in the surface order. In the usual way of thinking about things, verb movement is taken to raise the verb out of the VP past the S to create the surface order. Of course, that familiar way of thinking then faces the problem of how to motivate the relevant movement(s), often linked to morphological requirements.

In the present architecture, we do not face the same problem of finding an order-specific motivation to move the V out of the VP. Instead, the order is input to the universal stack-sorting algorithm, which performs the steps Push (V), Pop(V), Push(S), Push(O), Pop(O), Pop(S). This produces the base order, in invariant head-complement-specifier order (in this case, VOS; see section 4.4 for some remarks on this perhaps-surprising rendering of the base order). This base order is what is read by the SR device, building the familiar hierarchical constituent structure, [[$_{VP}$ V O] S], containing a VP excluding the S. The very same base order is produced by stack-sorting any of the other stack-sortable permutations of the base; these other permissible surface orders are SVO, SOV, VOS, OVS. On the other hand, *OSV cannot be processed in this way; it is the non-stack-sortable *231 permutation of the relevant base order.[3]

2.4 Rationalizing the architecture

[3] This would seem to predict that OSV is impossible as a surface word order. A caveat is in order: the prediction is that OSV is impossible as an *information-neutral* word order. This does not mean that OSV cannot occur; it means that such an order requires appeal to discourse-information effects, such as topic or focus interpretation of the O. As for the very small number of languages reported to have OSV as their basic clause order, we might appeal to Mahajan's (2000) point that few if any full sentences are in fact information-neutral. For example, surface subjects are often topics. See section 4.4. for more.

The present proposal can be seen as an extension of the ideas about parsing proposed in Medeiros' (2018) ULTRA (Universal Linear Transduction Reactive Automaton) model. The key innovation in that model is the mechanism of stack-sorting. Stack-sorting is a linear transduction algorithm, utilizing a last-in, first-out stack memory structure to reorganize an input sequence into a desired output sequence. Unlike the various more powerful sorting algorithms that are commonly used, stack-sorting is a partial sorting algorithm: a given output sequence can only be achieved for a subset of possible inputs (these are the stack-sortable strings).

In the context of linguistic parsing, Medeiros (2018) suggested that stack-sorting can be thought of as storage and retrieval of the linguistic input in a memory system dominated by recency effects, which in effect gives a stack-like structure (the most recent input is the most accessible, as in a LIFO stack). In that work, the desired invariant output was identified as the "order of composition", focusing on the case of Universal 20. For that application, the 14 attested nominal orders identified by Cinque (2005) are exactly the strings that can be stack-sorted into the order *N Adj Num Dem*, Cinque's nominal hierarchy read "bottom-up".

The main difference between the current framework and the ULTRA model concerns the nature and ordering of the output of stack-sorting. Rather than saying that stack-sorting produces an "order of composition", we are building the input to a language-invariant Shift-Reduce (SR) semantic parser. This base order is in head-complement-specifier order. Given that the order is to be read by an SR device, this is one of four possible orderings for a basic X-bar structure; two logically possible orderings, namely head-specifier-complement and complement-specifier-head, are automatically ruled out by the inherent context-free nature of a pushdown automaton like an SR parser. In fact, it is one of two orders (head-comp-spec, spec-comp-head) that can be specified by a single statement about the order of heads with respect to everything else, in this case "heads first". But that is still a stipulation.

In this regard, we might appeal to another argument made in Medeiros (2018), that this ordering of the semantic structure resembles Reverse Polish Notation (RPN). Thus, this parsing architecture closely mirrors a classic architecture for arithmetic calculators, which stack-sort user input into RPN for computation by an internal SR device. The motivation for this design for calculators is particularly intriguing, from the Minimalist point of view: it is considered an optimal design for minimizing the burden on internal memory resources. Insofar as similar considerations carry over to the present case, then head-complement-specifier order can be motivated non-stipulatively, on Minimalist grounds.

Two further considerations might best be thought about from this parsing perspective. The first concerns discourse-information structure and A-bar movement, which we have set aside as outside the scope of this work. Drawing on Henson's (1998) extremely successful Start-End Model of short-term memory, in which recency and primacy are encoded separately, Medeiros (2018: 11-12) speculates that Henson's recency code underlies information-neutral syntax, while the primacy code is crucially involved in discourse-information effects and A-bar relations. This makes immediate sense of two striking properties of this distinct syntactic system: its association with the left periphery, and its potentially long-distance character. Ultimately, the Duality of Semantics might be grounded in this dual coding of short-term memory.

A second consideration which might be resolved in the parsing perspective concerns the effects described by Abels (2016). Seeking to extend Cinque's theory of nominal ordering to a

more general theory of neutral word order, Abels considers the more general case of "satellites" of a lexical head. He argues that a successful extension must be relativized to classes of satellites, and that it is impossible to find a single hierarchical ordering encompassing all classes of satellites of a single head (notably, a verb) at once. On the one hand, this argument may or may not go through in the present framework, given our different assumptions and architecture. But granting the point, this effect too can probably be rationalized within the memory-based parsing framework. Specifically, there is a body of work emphasizing the role of content-addressable memory in parsing (McElree 2000), and highly successful theories of parsing incorporating cue-based retrieval (Lewis & Vasishth 2005; Lewis, Vasishth, & Van Dyke 2006). One way of reconciling Abels' observations with the present mechanism is to say that the present framework is an abstraction of recency effects in a storage-and-retrieval parser, while Abels' satellite classes implicate a role for content-addressability in this process. The different classes of satellites, then, would reflect the "grain size" of content-addressability (apparently implicating a rather small set of categorial distinctions in this domain), the idea being that items from one category can be retrieved with relatively little interference from elements belonging to a distinct category, while recency effects win out within a given category.

These are intriguing speculations, perhaps, but I have nothing to add beyond pointing out that this unusual architecture might be rationalized in terms of properties of the memory systems involved in language processing. Nevertheless, another goal of this paper is to provide an alternative formulation of the same system as a competence-level generative system, which need not be interpreted as a theory about processing *per se*, avoiding the potential pitfalls of identifying the grammar with the parser. As such, the relationship between abstract grammar and real-time processing is ultimately orthogonal to our concerns, and the architecture can be evaluated simply as a theory of the syntax of neutral word order variation. I postpone a specification of the alternate generative formulation until later in this work. Instead, we have enough of the technical apparatus in hand to appreciate its consequences for word order universals; we begin in the following section with Universal 20.

3. Generating Universal 20

As our first empirical application, consider possible and impossible neutral orders in the noun phrase, as described in Greenberg's Universal 20.

> "When any or all of the items (demonstrative, numeral, and descriptive adjective) precede the noun, they are always found in that order. If they follow, the order is either the same or its exact opposite." (Greenberg 1963: 87)

Considerable work since Greenberg's seminal study has sought to refine the empirical picture in this domain. I focus on the proposal of Cinque (2005) below.

3.1 Cinque's typology of attested orders

According to Cinque (2005), 14 of 24 possible orders of these four elements are attested.

(13) O*rders of demonstrative, numeral, adjective, noun, after Cinque (2005)*

a. **Dem Num Adj N**
b. **Dem Num N Adj**
c. **Dem N Num Adj**
d. **N Dem Num Adj**
e. *Num Dem Adj N
f. *Num Dem N Adj
g. *Num N Dem Adj
h. *N Num Dem Adj
i. *Adj Dem Num N
j. *Adj Dem N Num
k. **Adj N Dem Num**
l. **N Adj Dem Num**
m. *Dem Adj Num N
n. **Dem Adj N Num**
o. **Dem N Adj Num**
p. **N Dem Adj Num**
q. *Num Adj Dem N
r. **Num Adj N Dem**
s. **Num N Adj Dem**
t. **N Num Adj Dem**
u. *Adj Num Dem N
v. *Adj Num N Dem
w. **Adj N Num Dem**
x. **N Adj Num Dem**

Cinque shows that this pattern can be succinctly described by assuming a universal underlying base, built by a uniform sequence of External Merge operations, affected by phrasal movement, but not head movement or remnant movement (*i.e*. Internal Merge in the noun phrase must affect the noun, possibly pied-piping dominating structure).[4] His hierarchy is given in (14).

(14) [DemP ... [NumP ... [AdjP ... [N]]]]

The hierarchy in (14) is shorthand for a more articulated structure. Specifically, Cinque assumes the nominal modifiers are specifiers of associated functional phrases; he also posits interspersed agreement phrases, to host potential movements. Cinque (2005) does not provide

[4] Cinque adopts Kayne's (1994) Linear Correspondence Axiom (LCA), which requires extra structure to provide landing sites for movement. Abels & Neeleman (2012) argue that the LCA is unneeded; the relevant constraint is simply that movement is leftward.

bracketed representations for each order, instead describing each derivation in text. Steddy & Samek-Lodovici (2011), deriving the same result in OT, do:

(15) *Bracketed representations from Steddy & Samek-Lodovici (2011*)

a. [$_{AgrWP}$ [$_{WP}$ DemP $_{W}$ [$_{AgrXP}$ [$_{XP}$ NumP $_{X}$ [$_{AgrYP}$ [$_{YP}$ AP $_{Y}$ NP]]]]]]
b. [$_{AgrWP}$ [$_{WP}$ DemP $_{W}$ [$_{AgrXP}$ [$_{XP}$ NumP $_{X}$ [$_{AgrYP}$ NP [$_{YP}$ AP $_{Y}$ t_{NP}]]]]]]
c. [$_{AgrWP}$ [$_{WP}$ DemP $_{W}$ [$_{AgrXP}$ NP [$_{XP}$ NumP $_{X}$ [$_{AgrYP}$ [$_{YP}$ AP $_{Y}$ t_{NP}]]]]]]
d. [$_{AgrWP}$ NP [$_{WP}$ DemP $_{W}$ [$_{AgrXP}$ [$_{XP}$ NumP $_{X}$ [$_{AgrYP}$ [$_{YP}$ AP $_{Y}$ t_{NP}]]]]]]
k. [$_{AgrWP}$ [$_{YP}$ AP $_{Y}$ NP] [$_{WP}$ DemP $_{W}$ [$_{AgrXP}$ [$_{XP}$ NumP $_{X}$ [$_{AgrYP}$ t_{YP}]]]]]
l. [$_{AgrWP}$ [$_{AgrYP}$ NP [$_{YP}$ AP $_{Y}$ t_{NP}] [$_{WP}$ DemP $_{W}$ [$_{AgrXP}$ [$_{XP}$ NumP $_{X}$ t_{AgrYP}]]]]]
n. [$_{AgrWP}$ [$_{WP}$ DemP $_{W}$ [$_{AgrXP}$ [$_{YP}$ AP $_{Y}$ NP] [$_{XP}$ NumP $_{X}$ [$_{AgrYP}$ t_{YP}]]]]]
o. [$_{AgrWP}$ [$_{WP}$ DemP $_{W}$ [$_{AgrXP}$ [$_{AgrYP}$ NP [$_{YP}$ AP $_{Y}$ t_{NP}]] [$_{XP}$ NumP $_{X}$ t_{AgrYP}]]]]
p. [$_{AgrWP}$ NP [$_{WP}$ DemP $_{W}$ [$_{AgrXP}$ [$_{YP}$ AP $_{Y}$ t_{NP}] [$_{XP}$ NumP $_{X}$ [$_{AgrYP}$ t_{YP}]]]]]
r. [$_{AgrWP}$ [$_{XP}$ NumP $_{X}$ [$_{AgrYP}$ [$_{YP}$ AP $_{Y}$ NP]]] [$_{WP}$ DemP $_{W}$ [$_{AgrXP}$ t_{XP}]]]
s. [$_{AgrWP}$ [$_{XP}$ NumP $_{X}$ [$_{AgrYP}$ NP [$_{YP}$ AP $_{Y}$ t_{NP}]]] [$_{WP}$ DemP $_{W}$ [$_{AgrXP}$ t_{XP}]]
t. [$_{AgrWP}$ [$_{AgrXP}$ NP [$_{XP}$ NumP $_{X}$ [$_{AgrYP}$ [$_{YP}$ AP $_{Y}$ t_{NP}]]] [$_{WP}$ DemP $_{W}$ t_{AgrXP}]]]
w. [$_{AgrWP}$ [$_{AgrXP}$ [$_{YP}$ AP $_{Y}$ NP] [$_{XP}$ NumP $_{X}$ [$_{AgrYP}$ t_{YP}] [$_{WP}$ DemP $_{W}$ t_{XP}]]]]
x. [$_{AgrWP}$ [$_{AgrXP}$ [$_{AgrYP}$ NP [$_{YP}$ AP $_{Y}$ t_{NP}]] [$_{XP}$ NumP $_{X}$ t_{AgrYP}] [$_{WP}$ DemP $_{W}$ t_{AgrXP}]]]

3.2 On the empirical status of Cinque's revision of Universal 20: the case of Shupamem

There is considerable debate in the literature about the empirical status of Cinque's typology (see, *e.g.*, Dryer 2009, 2018; Nchare 2012; Cinque 2014; Abels 2016; Steedman 2020, *i.a.*). A full discussion goes beyond the scope of this paper. But a crucial caveat is worth discussing and defending: that the relevant typology concerns information-neutral, unmarked orders.

A particularly sharp challenge is presented by the extensive discussion of nominal ordering in Shupamem (Grassfields Bantu) by Nchare (2012). Nchare documents 19 permitted orders of demonstrative, numeral, adjective, and noun in Shupamem. This set of orders excludes some of the orders permitted by Cinque, while including a number of others that fall outside Cinque's typology. I reproduce the list in (16) (after Nchare 2012: 134, ex. 10), alongside Cinque's for comparison. The examples illustrated in Shupamem permute the orders of demonstrative *ʃì* 'this' [sic], numeral *kpà* 'four', adjective *mìŋkɛ't* 'dirty', and noun *pɔ`n* 'children' (note that postnominal modifiers are prefixed with noun class agreement *pí-*, which in the case of the demonstrative produces the form *pì'*).

(16) Orders in Cinque (2005) (left), vs. orders in Shupamem (Nchare 2012, right)

a. Dem Num Adj N	ʃì kpà mìŋkɛ't pɔ`n
b. Dem Num N Adj	ʃì kpà pɔ`n pí-mìŋkɛ't
c. Dem N Num Adj	*ʃì pɔ`n pí-kpà pí-mìŋkɛ't
d. N Dem Num Adj	*pɔ`n pì' kpà pí-mìŋkɛ't
e. *Num Dem Adj N	kpà ʃì mìŋkɛ't pɔ`n
f. *Num Dem N Adj	kpà ʃì pɔ`n pí-mìŋkɛ't
g. *Num N Dem Adj	*kpà pɔ`n pì' pí-mìŋkɛ't

h. *N Num Dem Adj	*pɔ`n pí-kpà pì' pí-mìŋkɛ't
i. *Adj Dem Num N	mìŋkɛ't ʃì kpà pɔ`n
j. *Adj Dem N Num	*mìŋkɛ't ʃì pɔ`n pí-kpà
k. Adj N Dem Num	mìŋkɛ't pɔ`n pì' pí-kpà
l. N Adj Dem Num	pɔ`n pí-mìŋkɛ't pì' pí-kpà
m. *Dem Adj Num N	ʃì mìŋkɛ't kpà pɔ`n
n. Dem Adj N Num	ʃì mìŋkɛ't pɔ`n pí-kpà
o. Dem N Adj Num	ʃì pɔ`n pí-mìŋkɛ't pí-kpà
p. N Dem Adj Num	pɔ`n pì' pí-mìŋkɛ't pí-kpà
q. *Num Adj Dem N	kpà mìŋkɛ't ʃì pɔ`n
r. Num Adj N Dem	kpà mìŋkɛ't pɔ`n pì'
s. Num N Adj Dem	kpà pɔ`n pí-mìŋkɛ't pì'
t. N Num Adj Dem	pɔ`n pí-kpà pí-mìŋkɛ't pì'
u. *Adj Num Dem N	mìŋkɛ't kpà ʃì pɔ`n
v. *Adj Num N Dem	mìŋkɛ't kpà pɔ`n pì'
w. Adj N Num Dem	mìŋkɛ't pɔ`n pí-kpà ʃì
x. N Adj Num Dem	pɔ`n pí-mìŋkɛ't pí-kpà ʃì

Summarizing, Nchare reports that Shupamem allows nineteen orders of these elements. In the present framework, we may readily dismiss the orders universally permitted by Cinque that do not happen to be available in Shupamem, namely orders (16c) *Dem N Num Adj* and (16d) *N Dem Num Adj*. However, the orders that are in fact available in Shupamem, but that lie outside Cinque's typology, would at first appear to be more of a problem. Specifically, there are seven orders reported in Shupamem that are outside Cinque's typology: (16e) *Num Dem Adj N*, (16f) *Num Dem N Adj*, (16i) *Adj Dem Num N*, (16m) *Dem Adj Num N*, (16q) *Num Adj Dem N*, (16u) *Adj Num Dem N*, and (16v) *Adj Num N Dem*.

A crucial point is that Nchare specifically counts non-neutral orders in Shupamem: "The revisited typology repeated in (10) will include focus as well as non focus orders" (Nchare 2012: 135). Thus, if all seven of the orders (16e, 16f, 16i, 16m, 16q, 16u, and 16v) in Shupamem that are not counted in Cinque's typology in fact involve focus, then they are irrelevant for present purposes. Let me repeat that the empirical target of the present study is information-neutral ordering, explicitly setting aside discourse-information effects including focus.

Indeed, it turns out that Nchare specifically mentions focus in the derivation of all of these seven problematic orders. The derivation of (16f) is assumed to be a further movement applied to (16e), which itself "[...] can be derived if we assume that there is a phrasal movement of the numeral to the specifier of DP where it checks the focus feature under D" (*ibid*., 215). Order (16i) "[...] is derived by fronting the AP *mìŋkɛ't* 'dirty' to the specifier position of DP to check its focus feature under D" (*ibid*., 218). Likewise in order (16m), "[...] the adjective undergoes a phrasal movement to a focus position" (*ibid*., 224). As for order (16q), *Num Adj Dem N*, Nchare's diagram (64q) shows the Num moving to spec of a D marked with a [+Foc] feature. (*ibid*., 227) The same is true for orders (16u) and (16v) (*ibid*., 231); for the former, Nchare is explicit that there is "AP movement to spec-DP to check its focus feature" (*ibid*., 231).

Thus, the results of Nchare's (2012) investigation of ordering in Shupamem are entirely consistent with the present account. While discourse-information effects make other orders possible, this is expected from the present perspective, and does not falsify the predictions made here, which keep solely to information-neutral ordering.

Before moving on, it is worth highlighting order (16p), *N Dem Adj Num*. While both Cinque (2005) and Nchare (2012) count this order as attested, in earlier work Nchare (2011) argued that is was not a grammatical order in Shupamem. This order presents theoretical challenges to the different accounts provided by Cinque and Nchare. While the details need not concern us, the large-bore problem surrounds the pattern of cross-serial dependencies found in this order. Consider selectional relations among (functional phrases hosting) nominal modifiers and each hierarchically lower element (what Steedman (2020) calls the *order of command*). The local relationship between N and Adj is disrupted in the surface form by the intervening Dem; likewise, the hierarchically local relationship between Dem and Num is interrupted by the intervening Adj.

This pattern turns out to be incompatible with the theoretical apparatus Nchare uses to account for other orders in Shupamem, which crucially invokes the Freezing Principle. The basic idea is that in the Merge-based derivation of order (16p), a syntactic object containing both the noun and the adjective moves above the base position of the numeral. From there, another movement applies, stranding the adjective in place, while moving the noun to precede the demonstrative. In Nchare's words, "[...] the extraction of NP from spec-AgrP to front it into the higher spec-AgrP seems to violates the freezing principle. I have no handy explanation for this violation in (64p). I will leave this issue for further investigation." (Nchare 2012: 226) Cinque, meanwhile, noting only three languages with this order, considers it "possibly spurious" (Cinque 2005: 320), suggesting that perhaps "[...] such subextractions should be ruled out entirely" (*ibid.*, 323, fn. 27).

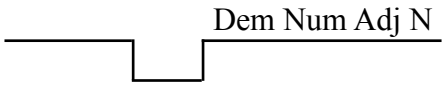
This is our first glimpse of an important feature of our treatment of hierarchy-order relations: it readily generates cross-serial dependencies, of exactly the sort attested in natural language syntax. In later sections, we will return to this point in the context of better-known cross-serial dependencies, showing that the theoretical problems they have raised for earlier theories dissolve under this account.

3.3 Reproducing Cinque's typology and bracketing

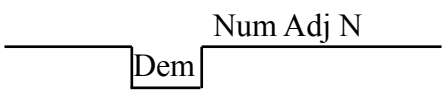
Returning to our theoretical account of Cinque's typology, we reconstruct exactly Cinque's (2005 *et seq*) version of Universal 20. Following his analysis in which demonstrative, numeral, and adjective are specifiers of functional phrases above the noun, the base order is (17).

(17) [[[[N] AdjP] NumP] DemP] *Base order for U-20 (with underlying brackets)*

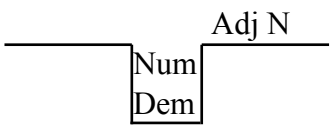
To illustrate how we will be stack-sorting nominal orders, consider first the English order (a), Dem-Num-Adj-N. This is the mirror image of the desired output sequence (17): the head N, followed by its modifiers as specifiers of stacked functional phrases. In (18), we see the stack-sorting process for this order, which ends up producing the desired (17) as output.

(18) a. 

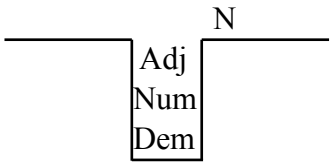
Start: input Dem-Num-Adj-N at right

b. 

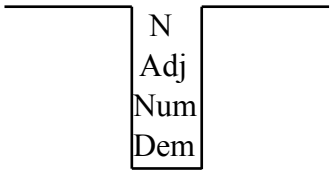
Push Dem onto the stack

c. 

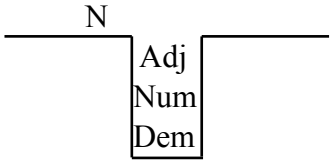
Push Num onto the stack

d. 

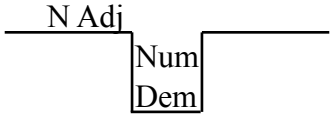
Push Adj onto the stack

e. 

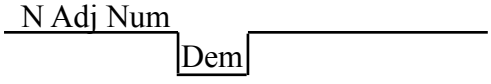
Push N onto the stack

f. 

Pop N from the stack

g. 

Pop Adj from the stack

h. 

Pop Num from the stack

i. 

Pop Dem from the stack:
output N Adj Num Dem at left

At the end of the process illustrated here, we have constructed the desired base sequence (17), in head-complement-specifier order. The sequence of operations is this:

(19) Push(Dem), Push(Num), Push(Adj), Push(N), Pop(N), Pop(Adj), Pop(Num), Pop(Dem)

As discussed above, we can consider Push operations to be left brackets, while Pop operations correspond to right brackets, both labeled by the element they affect. Translating (19) into a bracketed representation in this way, we get (20). As we will see in a moment, this corresponds systematically to the bracketing in Cinque's derivation, as shown above in (15).

(20) $_{\text{Dem}}$ ($_{\text{Num}}$ ($_{\text{Adj}}$ ($_{\text{N}}$ N) $_{\text{Adj}}$) $_{\text{Num}}$) $_{\text{Dem}}$)

The set of nominal orders which can be stack-sorted into the base order (17) are exactly the same 14 attested orders Cinque (2005) describes; the remaining 10 unattested orders are non-stack-sortable. Collecting the Push and Pop sequences for each attested order and representing them as bracketed representations (see Medeiros (2018) for full details), as in the conversion of (19) to (20), we get Table 1.

Nominal order	Bracketed representation (from Push-Pop stack-sorting sequence)
a. Dem-Num-Adj-N	($_{\text{Dem}}$($_{\text{Num}}$($_{\text{Adj}}$($_{\text{N}}$ N)$_{\text{Adj}}$)$_{\text{Num}}$)$_{\text{Dem}}$)
b. Dem-Num-N-Adj	($_{\text{Dem}}$($_{\text{Num}}$($_{\text{N}}$ N)($_{\text{Adj}}$ Adj)$_{\text{Num}}$)$_{\text{Dem}}$)
c. Dem-N-Num-Adj	($_{\text{Dem}}$($_{\text{N}}$ N)($_{\text{Num}}$($_{\text{Adj}}$ Adj)$_{\text{Num}}$)$_{\text{Dem}}$)
d. N-Dem-Num-Adj	($_{\text{N}}$ N)($_{\text{Dem}}$($_{\text{Num}}$($_{\text{Adj}}$ Adj)$_{\text{Num}}$)$_{\text{Dem}}$)
k. Adj-N-Dem-Num	($_{\text{Adj}}$($_{\text{N}}$ N)$_{\text{Adj}}$)($_{\text{Dem}}$($_{\text{Num}}$ Num)$_{\text{Dem}}$)
l. N-Adj-Dem-Num	($_{\text{N}}$ N)($_{\text{Adj}}$ Adj)($_{\text{Dem}}$($_{\text{Num}}$ Num)$_{\text{Dem}}$)
n. Dem-Adj-N-Num	($_{\text{Dem}}$($_{\text{Adj}}$($_{\text{N}}$ N)$_{\text{Adj}}$)($_{\text{Num}}$ Num)$_{\text{Dem}}$)
o. Dem-N-Adj-Num	($_{\text{Dem}}$($_{\text{N}}$ N)($_{\text{Adj}}$ Adj)($_{\text{Num}}$ Num)$_{\text{Dem}}$)
p. N-Dem-Adj-Num	($_{\text{N}}$ N)($_{\text{Dem}}$($_{\text{Adj}}$ Adj)($_{\text{Num}}$ Num)$_{\text{Dem}}$)
r. Num-Adj-N-Dem	($_{\text{Num}}$($_{\text{Adj}}$($_{\text{N}}$ N)$_{\text{Adj}}$)$_{\text{Num}}$)($_{\text{Dem}}$ Dem)
s. Num-N-Adj-Dem	($_{\text{Num}}$($_{\text{N}}$ N)($_{\text{Adj}}$ Adj)$_{\text{Num}}$)($_{\text{Dem}}$ Dem)
t. N-Num-Adj-Dem	($_{\text{N}}$ N)($_{\text{Num}}$($_{\text{Adj}}$ Adj)$_{\text{Num}}$)($_{\text{Dem}}$ Dem)
w. Adj-N-Num-Dem	($_{\text{Adj}}$($_{\text{N}}$ N)$_{\text{Adj}}$)($_{\text{Num}}$ Num)($_{\text{Dem}}$ Dem)
x. N-Adj-Num-Dem	($_{\text{N}}$ N)($_{\text{Adj}}$ Adj)($_{\text{Num}}$ Num)($_{\text{Dem}}$ Dem)

Table 1: Nominal orders and bracketings from stack-sorting operations

As already mentioned, there is another way of thinking about this pattern. Note that the bracketings here are simply all possible legal pairings of four left and right brackets (corresponding to the four elements Dem, Num, Adj, N). We can begin by generating all legal pairings of four sets of brackets. Writing the base order in (11) onto the right brackets as labels, and then reading word order from the labels of the corresponding left brackets, we derive the same word orders and labeled bracketed representations as we do from the stack-sorting procedure just illustrated (Table 2).

Brackets	Label right brackets	Label left brackets	Nominal order
(((())))	((((N)$_{\text{Adj}}$)$_{\text{Num}}$)$_{\text{Dem}}$)	($_{\text{Dem}}$($_{\text{Num}}$($_{\text{Adj}}$($_{\text{N}}$ N)$_{\text{Adj}}$)$_{\text{Num}}$)$_{\text{Dem}}$)	a. Dem-Num-Adj-N
(((())))	(((N)(Adj)$_{\text{Num}}$)$_{\text{Dem}}$)	($_{\text{Dem}}$($_{\text{Num}}$($_{\text{N}}$ N)($_{\text{Adj}}$ Adj)$_{\text{Num}}$)$_{\text{Dem}}$)	b. Dem-Num-N-Adj
(()(()))	((N)((Adj)$_{\text{Num}}$)$_{\text{Dem}}$)	($_{\text{Dem}}$($_{\text{N}}$ N)($_{\text{Num}}$($_{\text{Adj}}$ Adj)$_{\text{Num}}$)$_{\text{Dem}}$)	c. Dem-N-Num-Adj
()((()))	(N)(((Adj)$_{\text{Num}}$)$_{\text{Dem}}$)	($_{\text{N}}$ N)($_{\text{Dem}}$($_{\text{Num}}$($_{\text{Adj}}$ Adj)$_{\text{Num}}$)$_{\text{Dem}}$)	d. N-Dem-Num-Adj

(())(())	((N)Adj)((Num)Dem)	(Adj(N N)Adj)(Dem(Num Num)Dem)	k. Adj-N-Dem-Num
()()(())	(N)(Adj)((Num)Dem)	(N N)(Adj Adj)(Dem(Num Num)Dem)	l. N-Adj-Dem-Num
((()))()	(((N)Adj)(Num)Dem)	(Dem(Adj(N N)Adj)(Num Num)Dem)	n. Dem-Adj-N-Num
(()()())	((N)(Adj)(Num)Dem)	(Dem(N N)(Adj Adj)(Num Num)Dem)	o. Dem-N-Adj-Num
()(()())	(N)((Adj)(Num)Dem)	(N N)(Dem(Adj Adj)(Num Num)Dem)	p. N-Dem-Adj-Num
((()))()	(((N)Adj)Num)(Dem)	(Num(Adj(N N)Adj)Num)(Dem Dem)	r. Num-Adj-N-Dem
(()())()	((N)(Adj)Num)(Dem)	(Num(N N)(Adj Adj)Num)(Dem Dem)	s. Num-N-Adj-Dem
()(())()	(N)((Adj)Num)(Dem)	(N N)(Num(Adj Adj)Num)(Dem Dem)	t. N-Num-Adj-Dem
(())()()	((N)Adj)(Num)(Dem)	(Adj(N N)Adj)(Num Num)(Dem Dem)	w. Adj-N-Num-Dem
()()()()	(N)(Adj)(Num)(Dem)	(N N)(Adj Adj)(Num Num)(Dem Dem)	x. N-Adj-Num-Dem

Table 2: Nominal orders from legal bracketings

Even more significant than merely deriving the same set of orders, we also generate a simplified version of exactly the same bracketing that results from Cinque's Merge and Move account. From the bracketed representations in (15), keep only left brackets immediately before overt items, and the matching right brackets (we assume NP contains ...[N...]...). The result is identical to the bracketing in Tables 1 and 2. For example, consider order (13n) Dem-Adj-N-Num, with traditional representation in (15n), repeated in (21), now with [...N...] in NP.

(21) [$_{AgrWP}$ [$_{WP}$ DemP $_{W}$ [$_{AgrXP}$ [$_{YP}$ AP $_{Y}$ [$_{NP}$ N]] [$_{XP}$ NumP $_{X}$ [$_{AgrYP}$ t$_{YP}$]]]]]

Keeping only the left brackets immediately preceding overt elements, and their matching right brackets, yields (22) (I suppress the labels).

(22) [DemP [AP [N]] [NumP]]

The bracketing derived in the present account for order (13n) is repeated in (23).

(23) (Dem(Adj(N N)Adj)(Num Num)Dem)

In this novel kind of representation, left brackets are positions of pronunciation. Suppressing right bracket labels, and writing overt elements following their corresponding left bracket positions, gives (24).

(24) (Dem (Adj (N) (Num))

This is identical to the simplified standard bracketing for this order shown in (22); the interested reader may verify that this correspondence holds for every nominal order in (13/15) and Tables 1 and 2.

As a final note, we may associate these elements with indices representing their relative order in the base (17): N = 1, Adj = 2, Num = 3, Dem = 4. If we translate the set of logically possible nominal orders into their sequence of base indices, we will see that the attested, stack-

sortable orders are all 231-avoiding. Continuing with the example of order (15n) Dem-Adj-N-Num, the corresponding index sequence is 4213. This is 231-free.

Contrast this with all of the unattested orders; to pick one example, order (13e), *Num Dem Adj N. In terms of indices representing the base order of these elements, this is a 3412 order. This order contains the forbidden *231 contour (indeed, in two different ways: the subsequences Num...Dem...Adj, 342, and Num...Dem...N, 341, are both forbidden subsequences).

As we continue, it will be convenient to determine whether orders are stack-sortable or not by simply writing their corresponding base order (generally uncontroversial and fairly trivial, given our convention that the base order is in head-complement-specifier order), indexing the elements in their base order, and then examining the surface sequence for instances of the forbidden *231 contour. If no such contour is present, the order is stack-sortable, and predicted to be typologically possible. Conversely, finding any 231-like subsequence leads us to predict that the surface order in question is not possible as a neutral word order, in any language. Needless to say, this is a remarkably simple, even boring way of "doing syntax": we will not be worrying about steps of movement and what drives them, for example. Nevertheless, this approach yields empirically correct predictions across an interesting range of cases, subsuming the effects of a number of different word order universals and principles that have been thought of as having some other source (*e.g.*, minimality effects in movement). As we will see, our single *231 principle does the work of all of these principles, and arguably does a better job.

3.4 Interim summary

This section has reviewed Cinque's (2005) version of Greenberg's (1963) Universal 20. We considered the empirical challenge posed by Nchare's (2012) description of Shupamem, concluding that all of the putative counter-examples to Cinque's typology presented by that language are in fact non-neutral focus orders, and as such irrelevant to the present study.

We have seen an application of the surprisingly simple architecture of our universal stack-sorting supergrammar. Cinque's set of 14 attested neutral orders prove to be exactly the stack-sortable permutations of the base order, while his 10 unattested orders are its non-stack-sortable permutations. Equivalently, we get the same result by simply generating all legal pairings of brackets, labeling all right brackets in left-to-right order with the invariant base order, and reading word order in from left brackets, Furthermore, the Surface Trees we generate turn out to correspond directly to the syntactic structures derived in Cinque's account, though they are systematically simpler. This is remarkable, as Cinque's account represents a traditional External and Internal Merge derivation of the relevant orders and structures, obeying a parochial condition (movement in this domain can only be phrasal, excluding head movement and remnant movement). Here, by contrast, the same orders and structures are derived in quite a different way, without invoking movement; the unattested orders are not ruled out by a constraint on movement, but simply cannot be generated.

In the next section, we turn to what appears to be an unrelated set of facts, the Final-Over-Final Condition, and show that they are predicted by the same architecture.

4 Generating the Final-Over-Final Condition

In this section, I show that the present account provides a ready explanation for another intensively studied word order universal, the Final-Over-Final Condition (FOFC; Holmberg 2000, Biberauer *et al* 2014, Sheehan et al 2017 *i.a.*). This is a surprising unification, as Universal 20 and FOFC appear to conflict; see for example Roberts (2017) on modifying the hierarchy for the noun phrase (17) to be compatible with FOFC.

4.1 Background: The Final Over Final Condition

FOFC prohibits configuration (25):

(25) *[$_{\alpha P}$ [$_{\beta P}$ β γP] α] *The Final Over Final Condition*

That is, a head-final phrase cannot dominate a head-initial phrase. The example below, from Finnish, illustrates the phenomenon.

(26) a. yli [rajan maitten välillä] [P_1 [N_1 [[N_2] P_2]]]
across border countries between
'across the border between countries'

b.*[rajan maitten välillä] yli *[[N_1 [[N_2] P_2]] P_1]
border countries between across (Biberauer *et al* 2014: 187, *ex*. 29)

In the ungrammatical (26b), the outermost P_1 has its NP complement on the left, while the embedded nominal has its PP complement on the right. This is the banned *head-final over head-initial configuration. Biberauer *et al* (2014) list the following FOFC effects; these configurations are robustly ungrammatical across languages.

(27) a. *V-O-Aux *[$_{AuxP}$ [$_{VP}$ V DP] Aux]
b. *V-O-C *[$_{CP}$ [$_{TP}$ T VP] C] or *[$_{CP}$ [$_{TP}$ [$_{VP}$ V O] T] C]
c. *C-TP-V *[$_{VP}$ [$_{CP}$ C TP] V]
d. *N-O-P *[$_{PP}$ [$_{DP/NP}$ D/N PP] P]
e. *Num-NP-D(em) *[$_{D(em)P}$ [$_{NumP}$ Num NP] D(em)][5]
f. *Pol-TP-C *[$_{CP}$ [$_{PolP}$ Pol TP] C]
(Biberauer *et al* 2014: 196, *ex*. 46)

These canonical FOFC effects obtain when the elements in question are in a head-complement relation. This well-known characterization of the domain of this condition is the key

[5] See Roberts (2017) for motivation of this claim. D(em) here reflects an analysis where Dem originates low in the hierarchy, and in some languages moves to higher head D.

to the unification of this class of word order constraints with the Universal 20 pattern, as shown in the next subsection.

4.2 Head-complement order in Deep Strings

In the current framework, we adopt a rather traditional X-bar format for the base structure.[6] Unlike now-standard approaches involving unordered semantic representations, we assume that the base has a uniform head-complement-specifier order (which follows from a simpler stipulation: heads first).

(19) Head Complement *Head-complement base order*

This convention, together with the *231 theorem of TTG, predicts the basic phenomenology of the Final-Over-Final Condition (FOFC; Sheehan *et al* 2017) in structures characterized by head-complement relations.

To see this, consider a configuration with nested complementation: head α takes a complement headed by β, which in turn has complement γP. The base order is then (29) α β γP, and the forbidden *231 permutation is (30) *β γP α.

(29) α [β [γP]] *Nested complementation base structure*

(30) * β γP α *Forbidden word order*

Order (21) is traditionally described as a head-final phrase (αP) dominating a head-initial phrase (βP); this is exactly the configuration ruled out by FOFC (25), repeated as (31).

(31) *[$_{\alpha P}$ [$_{\beta P}$ β γP] α]

For example, if head *Aux* has complement headed by *V*, with complement *Obj*, the base order is *Aux V Obj* (32). We correctly exclude unattested *231 order **V Obj Aux* (33).

(32) Aux [V [Obj]]

(33) *V Obj Aux

---

[6] Additional stipulations may be required to model conjunction, set aside here. But note that if we treat coordination asymmetrically with the X-bar mechanisms here, akin to [N PP] complementation (e.g., coordination of N heads would form [N_1 [& N_2]] in traditional notation), we would predict an apparent typological gap in monosyndetic coordination (Haspelmath 2017) for order *& N_2 N_1 (this application is an observation of Ryan Walter Smith). See also Zwart (2009), and Roberts (2019: 602-3).

Since the reasoning is about heads and complements (not just verbs and auxiliaries), we expect this to generalize to any head-complement chain, reconstructing the core of FOFC.

4.3 Further extensions of FOFC

What about structures with both adjuncts and complements? Sheehan (2017) argues that FOFC extends to certain adjunct relations. Concretely, parallel to the FOFC effect **V Obj Aux*, **V Adv Aux* is unattested. A full discussion is put aside, but note that this effect is correctly predicted here. Following much recent cartographic work, we treat adverbials as specifiers of functional phrases, which will thus occur in the later portion of the base order. The base structure for the case Sheehan discusses is then [Aux [[V ...] Adv]] (34); unattested **V Adv Aux* (35) is the forbidden *231 permutation.

(34) [Aux [[V...] Adv]] *Auxiliary, verb, adverb base structure*

(35) *V Adv Aux *Forbidden word order*

Next, let us consider a fuller structure for the clause. Arranging standard assumptions about clause structure into our assumed head-complement-specifier base order, major categories of a transitive clause are underlyingly as in (36). Here, it is important to note that S and O really signify thematic subject and object (rather than the superficial grammatical functions); in other words, external argument and internal argument.

(36) [C [Pol [T [Asp [*v* [V O] S]]]]] *base structure for transitive clause*

It is helpful to consider elements belonging to a single hierarchy three at a time; we should find, for each such triple, five attested orders and one forbidden order. Drawing on order (36), understanding that the *O* position may be realized as clausal complement *CP*, we make the following predictions (among many others) about impossible neutral orders.

(37) a. *O S V
b. *CP S V
c. *O S T
d. *V O T
e. *V O C
f. *V CP T
g. *[C TP] V
h. *Pol TP C
i. *V S T

An adpositional phrase object *O* will be hierarchically ordered after a noun head *N* it complements (38); I take adposition *P* to be a head with noun phrase complement *NP* (39).

(38) N O_N *base order for noun and complement object*

(39) P NP *base order for P and nominal complement*

Taken together, this yields base order (40), with forbidden Surface String (41).

(40) P [N O_N] *Deep String for PP within PP*

(41) *N O_N P *Forbidden order*

This explains the typological gap illustrated in Finnish (26b) above, previously described with FOFC. In fact, setting aside (27e) (we adopt Cinque's hierarchy for Universal 20 effects), we have reconstructed the list of canonical FOFC effects in Biberauer *et al* (2014: 196), repeated below in (42).

(42) *FOFC effects predicted here*

a. *V O Aux	see (24)
b. *V O C	(31e)
c. *C TP V	(31g)
d. *N O P	(35)
e. *Pol TP C	(31h)

Beyond reconstructing this core of FOFC effects, (37) contains other interesting predictions. If one basic clause order is to be ruled out, **O S V* appears to be the right choice (31a), as it is the rarest cross-linguistic order. Among 1376 languages recorded in WALS (Dryer 2013) as having a single dominant clause order, only four are reported to have this order (Warao, Venezuela; Nadëb, Brazil; Wik Ngathana, Australia; Tobati, Indonesia). That said, some mechanism going beyond the simple base-generation system here must be invoked for the handful of languages with OSV orders. Another interesting prediction is (37i), taken up again in section 5 below as a reformulation of Travis' (1984) Head Movement Constraint.

4.4 An aside: VOS as the base order?

Before moving on, a comment is probably in order about the assumed underlying order of basic clausal components V, O, and S, and its relation to more or less frequent surface word orders. The issue is that we claim VOS is the underlying base order for the clause; this may seem surprising, as VOS is a rare surface order, exhibited by something like 2% of languages. In fact, there is an even more basic question looming in the background: given some underlying base order, why should there be any pre-processing step (like the stack-sorting mechanism explored here) at all? In other words, why don't we expect the base order to be the surface order for all languages?

Thinking more carefully about things, however, this kind of internal-external isomorphism is probably not what we expect. It is a familiar idea that language presents different

problems for adults and children. In seeking to understand typological trends in language structure, it makes sense to pay particular attention to the problems various configurations present to children. This is because of a kind of filtering effect: only those languages that are acquirable by children will survive as possible adult languages.

A robust finding of studies of language comprehension is that, for native adult speakers, comprehension is incremental, proceeding word-by-word. Indeed, it is clear that comprehension is even faster than that, running ahead of the actual input and predicting what is to come. It has also become clear that word-by-word incrementality relies on this kind of forward predictive ability. Colin Phillips' group at the University of Maryland has shown, for example, that in OV languages like Japanese, immediate integration of the O requires prediction of a V, before the V is actually encountered (Phillips & Lewis 2013, Momma et al 2015; note that this aligns with the head-complement base order we have proposed). Development of this predictive skill is clearly a major component of acquiring a language; in the present account, differences between languages may be exclusively due to this effect.

At the same time, it is also clear that being able to predict in a language is not to be identified with being able to comprehend the language at all (contrary to some recent proposals identifying prediction and cognition). See especially Huettig & Mani (2016) on this point. It seems, instead, that language comprehension involves dual mechanisms of control, both proactive and reactive (Braver, Gray, & Burgess 2007).

With this in mind, consider the situation of a child encountering a language for which they have not yet developed adult-level predictive skills. Thus, the (apparent) word-by-word incrementality of adult comprehension is not yet in place, and there is some degree of lag between receiving auditory input and constructing an internal representation of the signal. The internal processing may be relatively slow or fast compared to the input stream. When processing is slow, the received input must be held in a memory buffer until it can be integrated into the internal structural representation. It is known that, relative to other kinds of input, comprehensible natural language in the auditory modality exhibits an outsized role for recency effects (Surprenant, Pitt, Crowder 1993). In computational terms, recency corresponds to a stack-like memory structure (i.e., last-in, first-out).

Suppose that our proposal is correct, in that there is a fixed, common linear order for comprehension across languages, independent of surface order. For example, in the case of verb and object, this is V-O (consistent with the experimental findings mentioned above indicating that in OV languages, immediate integration of the O requires prediction of V). More generally, we have supposed that the underlying order is head-complement-specifier.[7] Given the storage and retrieval routine just described, and assuming that to a first approximation the available memory store behaves like a stack, it follows that the orders that the child can process are the stack-sortable orders. That is, this kind of device is capable of comprehending just those orders that can be passed, one item at a time, into a stack memory, interleaved with operations

[7] This assumes that the X-bar format is adequate to model natural language semantic structure, which may well be an oversimplification. In particular, effects like coordination and unstructured iteration seem to require something more (or, especially in the latter case, less). See Krivochen (2016, 2020) for relevant discussion.

retrieving one item at a time to form the desired underlying base order in the correct (head-complement-specifier) order.

Thinking of things this way may shed light on our apparent problem in supposing that VOS is the basic underlying order. This is only a problem on the assumption that the underlying order should be reflected directly in surface order. Given the remarks just above, this requires the language-learning child to parse such languages "fast", item-by-item as the input is received, in order to achieve the desired output order. Particularly in the earliest stages of acquisition, this seems overly demanding. The least demanding scenario, from the point of view of the child, is if a clause of the language can be decoded "at leisure" after it has been heard. Given the stack-like memory correlating with the dominance of recency effects in language comprehension, underlying VOS order would be easiest to achieve for the mirror-image order, SOV. That sounds promising, since SOV is indeed the most common order across the languages of the world.

Continuing in this vein, there is plausibly an asymmetry between V, on the one hand, and O and S: the former is a head, and the latter are (at least potentially) full phrases. In terms of speed of processing, it is reasonable to suppose that a single head might be processed "fast", while a full NP would take rather more time to process. In that case, we expect the positioning of V to be rather flexible; SVO or VSO order would present relatively little processing difficulty, if V can be handled relatively quickly. But OVS and VOS surface orders require processing of the O to be complete before the S is encountered, which, by this reasoning, is significantly more demanding, possibly explaining the relative rarity of these orders.

4.5 Interim summary

In this section, we have demonstrated that our basic theorem, *231, extends without additional machinery to cover the core empirical terrain described by the Final-Over-Final Condition. Specifically, the head-final over head-initial configuration banned by FOFC instantiates a *231 permutation of the underlying base orders, because the base is organized in head-complement-specifier order.

This is interesting for a number of reasons. First, FOFC has been held up as an instance of a purely syntax-internal constraint that is not explained by other factors. Something of the sort is still true in this architecture, but FOFC, as an instance of *231, is a necessary consequence of our system, rather than an additional constraint on movement that could have turned out otherwise. This result is also significant in that the account unifies Universal 20 and FOFC as instances of the same phenomenon. This is starkly at odds with the traditional treatment, where the two effects seem to have little to do with each other. Indeed, they in fact seem to conflict; see the discussion in Roberts (2017) for relevant considerations.

In the following section, we turn to another application of our *231 theorem: it derives a version of the Head Movement Constraint, while allowing known exceptions.

5 The Head Movement Constraint and its exceptions

The present account also explains Travis' (1984) Head Movement Constraint (HMC), while correctly predicting some well-known exceptions. Travis argues that a head cannot move to a higher head position over an intervening governing head, formalizing this claim with her Head Movement Constraint (HMC).

(43) Head Movement Constraint (HMC) (Travis, 1984, 131)
An Xº may only move into the Yº which properly governs it.

The HMC (43) rules out configurations like (38), where head Zº has "skipped" intervening head Yº and left-adjoined to Xº.

(44) *[... Zº-Xº ... [... Yº ... [... t_{Z^o}]]]]

Of course, merely requiring head movement to be short does not suffice to rule out (44); it is also required that Zº cannot move to Yº first and then excorporate, moving without Yº to Xº. That requirement is a stipulation that does not appear to follow from independent principles, and is quite different from phrasal movement, which obeys no such restriction (phrases may move successive-cyclically without picking up additional structure along the way). Indeed, head movement remains controversial, posing a number of challenges to standard accounts of syntactic movement, and has inspired a variety of analyses; see Dékany (2018) for a recent overview.

That the present account extends to these effects is surprising at first glance, as movement violating the HMC does not produce an impossible *231 order of the heads themselves. Instead, HMC-violating movement "skipping" an intervening head, as in (44), produces 312 order among the heads, readily generated by this system.

Consider a simplified version of our base clause structure (36). For ease of exposition, we focus on the core C-T-V categories; adding more elements does not affect the conclusions. First, we examine each possible permutation of *C, T, V* order (45). Next to each order permutation, we write the corresponding index sequence (123, etc), and, for clarity, the traditional description of the derivation of the order (e.g., V-to-T, indicating head movement of V to T).

(45) Permutations of *C,T,V* head order

a. C T V	1 2 3	base order
b. V T C	3 2 1	V-to-T-to-C
c. C V T	1 3 2	V-to-T
d. V C T	3 1 2	HMC violation, V-to-C skipping T: ok
e. *T V C	*2 3 1	FOFC violation
f. T C V	2 1 3	T-to-C

As indicated, our *231 principle only rules out the FOFC-violating order *T-V-C. The order violating the head movement constraint, V-C-T ("long" head movement of V to C, skipping T) is generated without problems.

However, something interesting emerges when we consider the order of these elements with the addition of another element from later in the clause base structure, an external argument S. We repeat the permutations in (45), now trying all possible surface positions of S with respect to the orders in (45); a * marks an impossible position (one that will produce the forbidden *231 contour in the surface order).

(46) Permutations of *C,T,V* head order with interposed *S*

a. (S) C (S) T (S) V (S)	(4) 1 (4) 2 (4) 3	base order
b. (S) V (*S) T (*S) C (S)	(4) 3 (*4) 2 (*4) 1 (4)	V-to-T-to-C
c. (S) C (S) V (*S) T (S)	(4) 1 (4) 3 (*4) 2 (4)	V-to-T
d. (S) V (*S) C (*S) T (S)	(4) 3 (*4) 1 (*4) 2 (4)	HMC violation: *231
e. *T V C (already ruled out)	*2 3 1	FOFC violation
f. (S) T (*S) C (S) V (S)	(4) 2 (*4) 1 (4) 3 (4)	T-to-C

In (46), we generate: (46a) *C T V* (no head movement); (46b) *V T C* (full roll-up of heads obeying HMC, "V-to-T-to-C"); (46c) *C V T* (partial HMC-obeying movement, "V-to-T"); and (46f) *T C V* (partial HMC-obeying movement, "T-to-C"). The independently FOFC-violating order (46e) is ruled out already, and adding the subject anywhere has no effect (the forbidden subsequence persists regardless of additional material). Meanwhile, (46d) *V C T* ("V-to-C, skipping T"), the HMC-violating order, is ruled out only if a higher-index element (*e.g.*, the external argument *S*) intervenes between *V* and *T* (or another higher head).

We have generated all the core cases of local head movement in the C-T-V system. Note, too, that we derive another important effect: obligatory surface adjacency of the head cluster. That is, a later element from the base order, such as the external argument, may never occur between an inverted sequence of heads (because the heads then form a 21 sequence; the later 3 between them produces the forbidden *231 permutation). In other words, all of the instances of head movement forming a "complex word", such as V-to-T, or V-to-T-to-C, necessarily occur adjacent, without the possibility of other material intervening. Most interesting of all, we arrive at a new claim: there is nothing wrong *per se* with the "improper head movement" case (45d) that violates the HMC. Instead, what we expect to be ruled out is only a subsequence in which a later element from the base order, in particular the external argument S, occurs between the long head-moved V and a higher head like C or T in the surface order. Abstracting now from the simple C-V-T system to a larger set of clausal head positions, we rule out (47):

(47) *V Subj *v*/voice/Aux/T

That is, the verb cannot precede an external argument which precedes some head above *V* (in fact, we have seen this prediction already, in (37i) in section 4). As far as I know, that gives the right facts for V-to-T and T-to-C movement captured by the HMC. Importantly, understanding the HMC as actually reflecting the condition in (47) also allows us to account for "long" head movement in Breton, a much-discussed violation of the HMC.

(48) *Breton*

Lennet en deus Anna al levr
read.pprt has Anna the book
'Anna has read the book' (Roberts 2010: 194)

I take the base order for this example to be (49); (50) shows the indexing of the Surface String, which is indeed 231-free, and thus generable in this system (the treatment of the auxiliary Aux and associated affix -Fx will be discussed in much more detail in the following section, where I discuss English Affix Hopping).

(49) T Aux -Fx V O S
1 2 3 4 5 6

(50) Lennet en deus Anna al levr
V -Fx Aux T[8] S O
4 3 2 1 6 5

Certain Slavic languages also allow fronting of a bare participle, as in Bulgarian.

(51) a. Bjah **pročel** knigata.
had read the.book
'I had read the book.'
b. **Pročel** bjah knigata.
read had the.book
'I had read the book.' (Harizanov & Gribanova 2019:482)

(52) a. Bihte bili **arestuvani** ot policijata.
would been arrest.PTCP by the.police
'You would be arrested by the police.'
b. **Arestuvani** bihte bili ot policijata.
arrest.PTCP would been by the.police
'You would be arrested by the police.' (Embick & Izvorski 1997:231)

Interestingly, in all these cases information-neutral long head movement obeys the **V S T* condition in (47).[9] That is, either the subject is placed after the entire verbal complex, as in Breton (48), or the subject is null, as in the Bulgarian examples in (49); (50) shows a prepositionally-marked passivized thematic subject, which again does not intervene between participle and higher heads right of the participle.

---

[8] The relative order of T and Aux within *en deus* is unknown to me, but ordering T before the Aux produces a 231-free order (431265) as well.

[9] Ian Roberts (p.c.) and Maria-Luisa Rivero (p.c.) observe that this appears to be true quite generally.

An important aspect of this analysis is that it treats head movement in the same terms as cases of apparent lowering, most famously in the English verbal system (in Affix Hopping), but proposed for phenomena in other languages as well, such as C-to-Neg lowering in Irish as proposed by McCloskey (2017). This is a step forward, as such cases closely resemble upward head movement, with the single exception of the linear position of the cluster of heads (the morphologically complex word). Despite this phenomenological similarity, lowering and Affix Hopping have generally been viewed as effects distinct from head movement.

The analysis of head movement has long presented theoretical problems. In standard analyses, it violates the Extension Condition on Merge, and produces a structure in which the moved element does not c-command its trace. Here, this kind of pattern is united with other information-neutral ordering phenomena. No special counter-cyclic mechanisms, morphological readjustment, construction-specific constraints such as a ban on excorporation, or other such special machinery is invoked. As we will see, the analysis also treats lowering operations like Affix Hopping in the same terms, a welcome result.

Summarizing, our *231 principle of neutral word order derives a version of Travis' (1984) Head Movement Constraint (HMC) that covers core cases of (V-to-)T(-to-C) movement, including obligatory surface adjacency of the "head cluster", while also allowing attested LHM as in Breton and Bulgarian. We discover a novel and apparently exceptionless generalization about when neutral LHM is possible: only when it obeys **V S T/Aux*. No special principles or mechanisms are invoked; head movement, often seen as unlike other kinds of syntactic movement, falls together with Universal 20 and FOFC as an immediate consequence of our single *231 principle.

6 Generating some well-known crossing dependencies

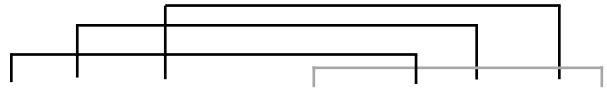
Thus far, we have mostly been concerned with ruling out typologically unattested orders. In this section, I turn to showing that the analysis of allowed orders extends to somewhat exotic constructions that have figured prominently in arguments that natural language grammars are mildly context-sensitive (Joshi 1985). Specifically, the architecture provides simple analyses of attested cross-serial dependencies, including unbounded crossing subject-verb dependencies in certain Germanic languages, as well as the more limited crossing pattern seen in English Affix Hopping.

We have already glimpsed this additional generative power. For example, we briefly discussed the discontinuous constituency seen in the common word order VSO (section 2.3), and we noted that some of the orders allowed by Cinque's version of Universal 20 contain cross-serial dependencies (for example, his order (p) N-Dem-Adj-Num). While phenomena like VSO order have been taken as strong motivation for transformations (in this case, head movement of the verb to some higher position), the effects that we will examine in this section have proven much more challenging to describe within the usual conception of transformations. That is, even allowing upward, leftward movement does not suffice to generate them. In the case of English Affix-Hopping, the problem is that the relevant movement seems rather to be rightward lowering movement (of affixes associated with higher auxiliary verbs onto lower verbal stems).

Meanwhile, the long-distance cross-serial dependencies we will see immediately below would seem to require something like Richards' "Tucking in" movement, failing to obey the Extension Condition, or a radically different notion of movement, as in Tree-Adjoining Grammar. As we will see, our framework suffices to describe these patterns and how they are sorted into standard head-complement-specifier base structures, without any additional devices.

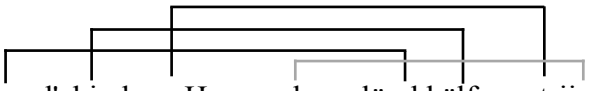
6.1 *Cross-serial subject verb dependencies*

Bresnan *et al* (1982) discuss unbounded crossing subject-verb dependencies in Dutch (Huybregts 1976). Example (53), taken from Steedman (2000: 25), illustrates the phenomenon.



(53) ...omdat ik Cecilia Henk de nijlpaarden zag helpen voeren
...because I Cecilia Henk the hippos saw help feed
'...because I saw Cecilia help Henk feed the hippos'

Shieber (1985) discusses similar word orders in Swiss German, which also show long-distance cross-serial case dependencies, as in (54).



(54) ... das mer d'chind em Hans es huus lönd hälfe aastriiche
... that we the children Hans the house let help paint
'...that we let the children help Hans paint the house'

Interestingly, the system already established can base-generate these orders.[10] I take the Dutch example (53) above to contain the categories in (55), abstracting away from internal structure of the object *de nijlpaarden* (see section 8) and segmenting a Tense suffix from inflected and non-finite verbs, even if realized as zero.

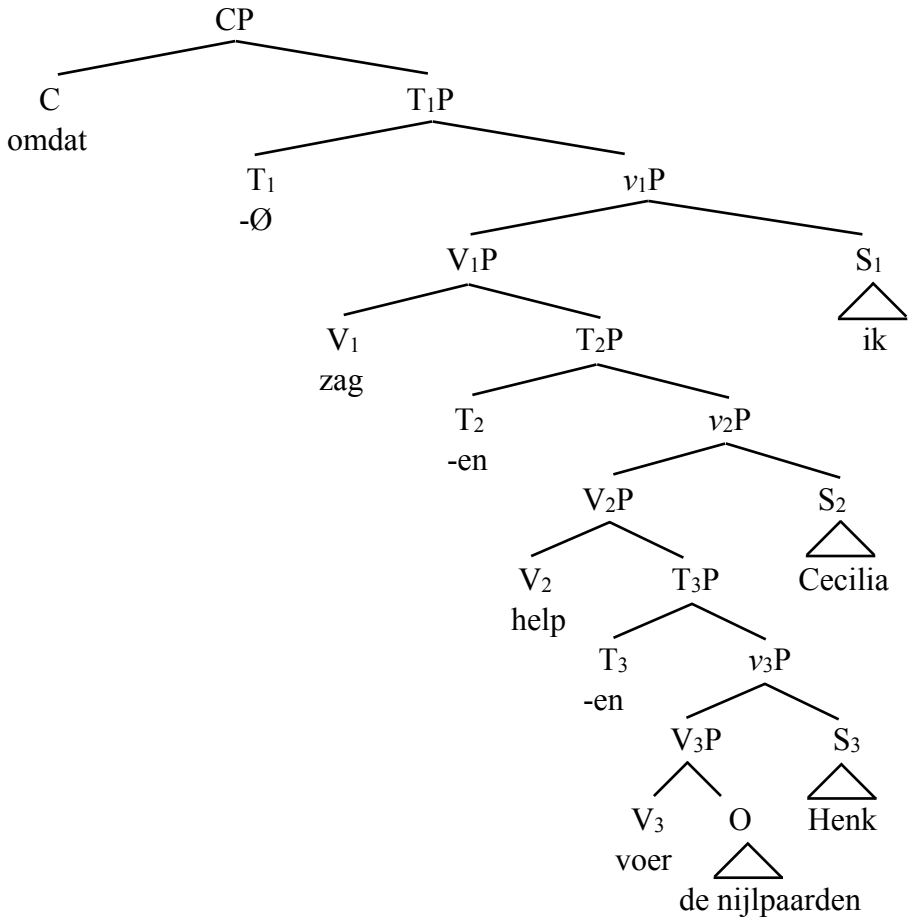
(55) ...omdat ik Cecilia Henk de nijlpaarden zag-Ø help-en voer-en
C S_1 S_2 S_3 O_3 V_1 T_1 V_2 T_2 V_3 T_3

The categories in (55) will be rendered as a single base order, which we assemble incrementally for clarity. Following the general clause ordering (36), with the standard assumption that complement clauses occupy the canonical direct object position, allows us to

[10] Stabler (2004) discusses four different classes of cross-serial dependency constructions, with distinct formal properties. I restrict attention to the two classes in this section.

assemble the base order for iterated clausal complementation.[11] For single clausal embedding, [$_{CP1}$ ...[$_{CP2}$]], we have the base order C-T_1-V_1-$\underline{T_2\text{-}V_2\text{-}O_2\text{-}S_2}$-$S_1$. Replacing O_2 with another embedded clause, we derive (56), the base structure for sentence (53) above.

(56) *Base structure for (53)*



Given this base structure, we can write the bracketed surface structure representation for the Dutch word order (53),[12] shown in (57). Here, I only show the categories, for reasons of space.

---

[11] At least for these structures, we are implicitly developing a simple account of recursion by substitution. For this example, the account requires that the entire structure be available at once; the clauses in this example cannot be cyclic domains, sorted one at a time (see discussion in section 8). Other clauses may be; I leave fuller consideration of recursion in this architecture to future work, beyond the brief comments below.

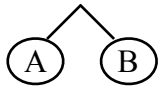
[12] An important question is whether these trees provide a basis for a successful theory of prosody. See also section 6.3 below, where it is shown that these trees are closer to standard representations of clausal architecture than initial appearances suggest.

(57) (C C) (S1 (S2 (S3 (O (V1 (T1 T1) V1) (V2 (T2 T2) V2) (V3 (T3 T3) V3) O) S1) S2) S1)

At this point, it is worth introducing some further representational technology, as the bracketed representation in (57) is difficult to read. There is a natural correspondence between legal bracketings and *n*-ary branching trees. We will go into more detail on this correspondence later in the paper; for now, let us simply illustrate how to convert labeled bracketings into corresponding *n*-ary branching trees. We do this in the obvious way: each bracket pair corresponds to a single node in the tree. Where a bracket pair encloses another bracket pair, the relevant nodes will stand in a dominance (containment) relation in the tree. And the matched labels of left and right brackets will simply be taken as the label of the corresponding tree node. Finally, to ensure that disconnected bracketings, like ( )( ), make a single rooted tree, we add an unlabeled root node dominating everything else.

Let us illustrate this tree notation with some simple examples first before applying it to the Dutch structure. Suppose we had a bracketed surface structure (A A)(B B). This corresponds naturally to the following tree structure.

(58) Tree corresponding to bracketed representation (A A)(B B)



It should be clear that this tree representation encodes exactly the same information as the bracketed representation (A A)(B B); namely, we have two disjoint objects, neither containing the other. It is rather subtle, but we understand the expression (A A)(B B) to itself be a whole object, hence the "spurious", unlabeled node dominating them both.

Now consider the other way of organizing a pair of labeled brackets: (B (A A) B). This will be drawn as a tree structure as in (59).

(59) Tree corresponding to bracketed representation (B (A A) B)

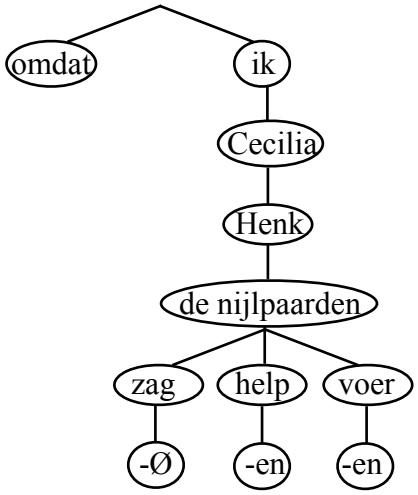
Again, it should be obvious that this is just another way of representing the same information: node/bracket pair A contains node/bracket pair B. We will have more to say about this in section 8, but since we want these brackets to represent stack-sorting operations, these two options exhaust the possibilities. That is, we will not allow *(B B)(A A) or *(A (B B) A), assuming the base order is AB (rather than BA). The brackets have meaning as Push and Pop operations, and only the forms in (58-59) order these operations correctly to output the base order.

The point of introducing this tree notation is that we can now represent the surface structure for the Dutch order (or, indeed, any stack-sortable surface order of any base structure) perspicuously. We will find an interesting use for this kind of representation in the generative

formulation presented later on. For now, though, (60) is the surface tree corresponding to the rather opaque bracketed form in (57), for Dutch sentence (53).

(50) ...omdat ik Cecilia Henk de nijlpaarden zag-0 help-en voer-en

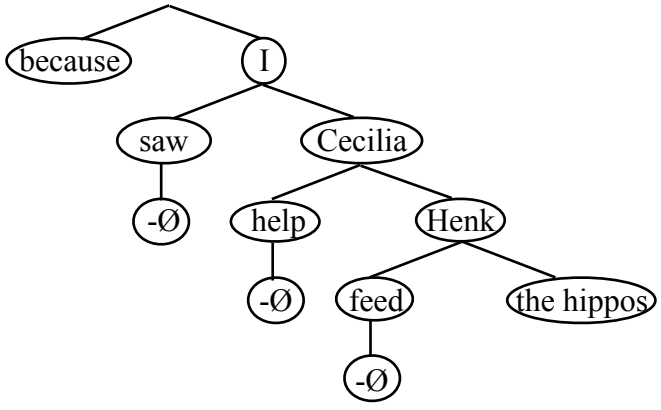
Category	C	S_1	S_2	S_3	O_3	V_1	T_1	V_2	T_2	V_3	T_3
Index	1	11	10	9	8	3	2	5	4	7	6



Note that with the relevant universal base structure resolved as in (56), we can readily represent other surface word orders of the same elements, as in English (61).

(61) ...because I saw -0 Cecilia help -Ø Henk feed -Ø the hippos

C	S_1	V_1	T_1	S_2	V_2	T_2	S_3	V_3	T_3	O_3	*Category*
1	11	3	2	10	5	4	9	7	6	8	*base order index*



We will see later on that two natural ways of moving from node to node in this tree (namely, preorder and postorder traversal) yield the word order and base order, respectively. This corresponds to the fact that in the less-readable equivalent bracketed surface structure representation, left brackets are positions in word order, and right brackets are positions in the base order.

6.2 Affix Hopping

The same architecture that provides a successful analysis of cross-serial subject-verb dependencies also readily allows another crossing configuration that has figured prominently in generative work. This is the pattern known as Affix Hopping. Chomsky's (1957) analysis of Affix Hopping provided a strong argument for the necessity of transformational rules, going beyond the generative capacity of phrase structure systems. Interestingly, however, this pattern has not been easy to analyze with the tools available in later iterations of generative theories. For example, early Minimalist work proposed that the relevant pattern did not involve overt syntactic movement at all, but rather resulted from checking features on fully inflected lexical item inserted from the lexicon. As we will see, the present account allows a return to something very close to the original transformational analysis, without introducing any new machinery.

Sentence (62), *They hadn't been eating cake*, illustrates the phenomenon. As Chomsky (1957) noted, affixes group with preceding auxiliaries in distribution and meaning, despite being separated by the intervening verb in surface order. This involves a more limited and local form of cross-serial relations than those illustrated for Dutch in the previous section.

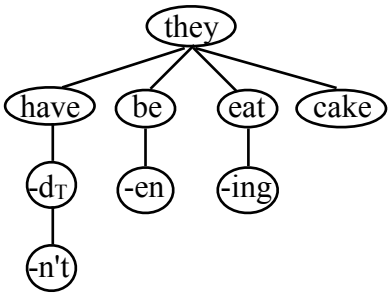
(62) They ha~~ve~~ -d -n't be -en eat -ing cake — Surface String
S Aux$_1$ T Pol Aux$_2$ -Fx$_1$ V -Fx$_2$ O — *categories*

In (63), I show the bracketed surface structure derived from stack-sorting (62).

(63) (They (have (-d (-n't -n't) -d) have) (be (-en -en) be) (eat (-ing -ing) eat) (cake cake) they)

This is, no doubt, difficult to read. Utilizing the correspondence between bracketed representations and *n*-ary branching trees described above, we can represent this instead as (64).

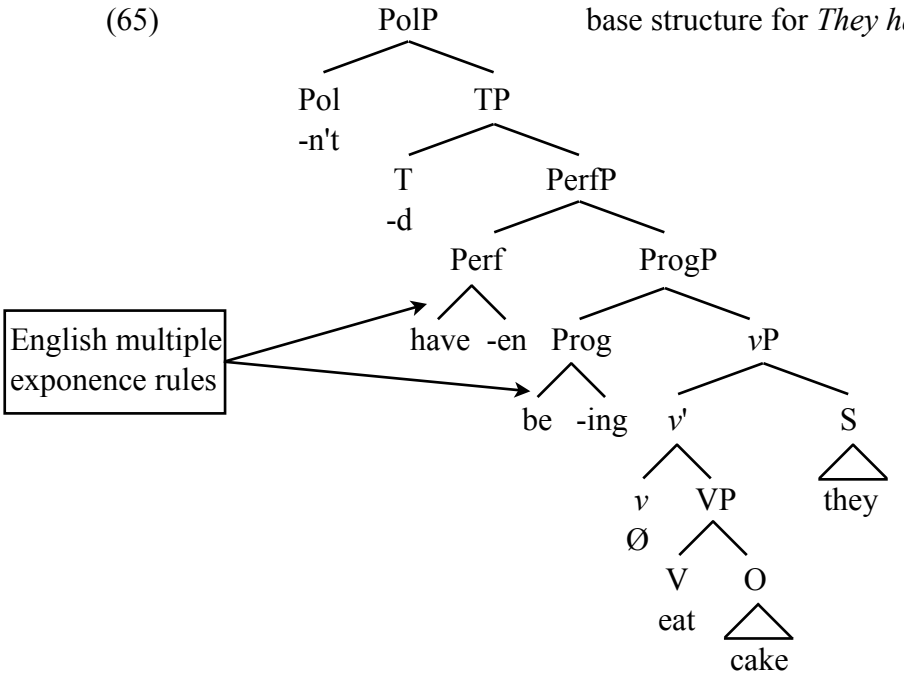
(64)



The representations in (63-64) above rely on the base structure given in (65) below, which I take to be a fairly uncontroversial proposal about the underlying hierarchy of the relevant categories. The only unusual property here is the rightward positioning of the specifiers, according to our head-complement-specifier convention for the base order. Note that auxiliaries and their associated affixes are adjacent in the base, plausibly constituting pieces of a single category. In other words, the auxiliary and associated affix reflect a multiple exponence rule specific to English.[13]

[13] We could also treat auxiliaries and affixes as hierarchically adjacent heads, as in Harwood (2013).

(65) base structure for *They hadn't been eating cake*

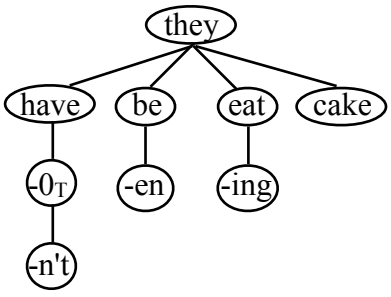


In effect, we have recovered Chomsky's (1957) classic analysis of Affix Hopping, wherein auxiliaries and associated affixes are introduced as a single lexical item, and a transformation "hops" the affix onto the following verbal element. Here, the relevant transformation is part of the standard deformation of the surface order into the base order by stack-sorting, the basic mechanism of this framework. No special principles or extra machinery is required; this is run-of-the-mill neutral syntactic displacement as implemented here. This seems an encouraging result.

6.3 Motivating the surface structure trees for clauses

At this point, the reader may well be curious about the surface trees here. At first glance, they bear little relation to well-established descriptions of branching structure. Consider again (64).

(66)



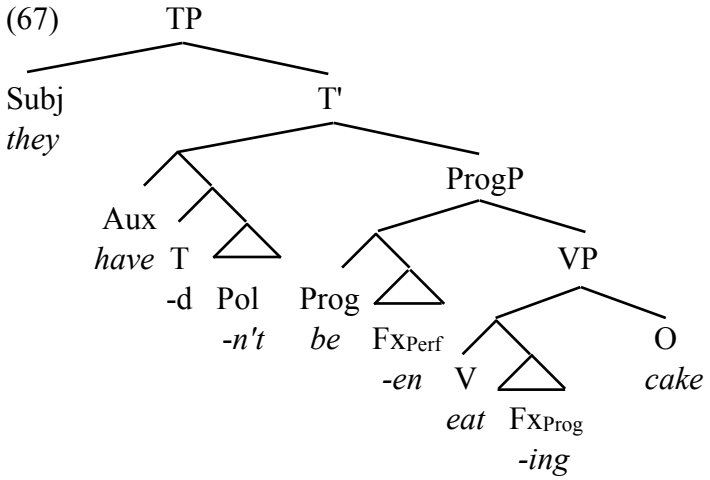
Obviously, this differs from standard representations of the structure of clauses in English. Rather than strict binary branching, we have an instance of quaternary branching. And

"terminal" elements are represented as directly dominating other terminals, with the subject on top of the tree. This seems quite alien.

But recall from the discussion of Universal 20 in section 3 that the bracketing in this theory is systematically "flatter" than standard representations. In particular, in going from standard representations and derivations to our representations, we lose left brackets that don't immediately precede a lexical item (and their associated right brackets).

To recover a more familiar version of the surface structure tree, we might try inserting some extra layers of structure that we may imagine have been lost in this mapping. Inspired by how Cinque's tree structures correspond to the simpler *n*-ary branching surface trees we find for nominal structures, we can reverse the process, arriving at something like (67).[14] The labels of the "recovered" phrasal non-terminals are included for familiarity, but should not be taken too seriously.

(67)



This way of "decompactifying" the surface structures we derive from stack-sorting leads us to conclude that head-adjunction structures are right-branching rather than left-branching; the

---

[14] For those interested in how this tree is found, we begin by formulating the surface structure representation as a string of labeled brackets, which can be understood to give the proper order of Push and Pop operations sorting the surface order into the base order:

(i) [They [have [-d [-n't -n't] -d] have] [be [-en -en] be] [eat [-ing -ing] eat] [cake cake] they].

To recover a "normal" representation with overt material on terminals, insert the terminals before the associated left brackets:

(ii) [They They [have have [-d -d [-n't -n't -n't] -d] have] [be be [-en -en -en] be] [eat eat [-ing -ing -ing] eat] [cake cake cake] they].

Finally, we insert extra brackets (or relabel existing brackets) to make the structure binary-branching in the familiar way:

(iii) [TP They [T' [Perf have[-d[-n't]]] [ProgP [Prog be[-en]] [VP[V eat[-ing]] [NP cake]]]]]].

The tree in (67) matches (iii). One might quibble about the proper labels at various places in (iii), but the point concerns the branching structure, which, *modulo* the remarks above about the direction of branching within head clusters, follows standard assumptions.

classical head-adjunction treatment of head-movement leading to that conclusion is, as we have seen, problematic anyway. One positive consequence of this kind of tree structure is that hierarchical and linear-order relations within head clusters mimic the Kaynean correspondence of hierarchy and order in larger structures (*i.e.*, rightward in the string is downward in the tree, even within head-adjunction structures). Otherwise, this looks similar to standard representations of the surface structure, with the subject in a high specifier (of TP, or AgrSP perhaps).

To be clear, the above should not be read as an endorsement of this more complicated kind of analysis (*i.e.*, I don't think we're collapsing the "real" tree). The cleaner relationship between word order and base order provided by the stack-sorting account is to be preferred, I would argue, unless some argument can be mustered in favor of the more complicated traditional analysis. The point is simply that the structure we find in this theory is a good deal closer to standard conclusions about the surface form than appearances may at first indicate.

6.4 Summary

In this section, we have seen that the present account readily allows cross-serial dependencies that have proven challenging to capture in other frameworks. In particular, both the cross-serial subject-verb dependencies of some Germanic languages, as well as the familiar pattern of Affix Hopping in English, turn out to require nothing beyond the tools we have already developed. Both patterns obey our fundamental *231 condition, given a relatively uncontroversial understanding of their base structure; as such, they are expected to be typologically possible.

There is, perhaps, something unsatisfying about all of this; it is so trivial as to be almost uninteresting. And while it is no doubt a good thing to do more work with less theoretical machinery, eliminating devices like morphological readjustment rules that have been invoked to get the proper orders, something else is lost too. For example, consider Bobaljik's (1994) Merger-Under-Adjacency (MUA) account of Affix Hopping. While adding complexity to the theory of syntax and its interaction with morphology, the MUA account is also able to nicely capture facts about *do*-insertion in English. This is the kind of effect that we do not expect to be able to explain within the present framework. That is, given that other languages allow main verbs to appear in the position obligatorily occupied by *do* in English under certain circumstances, we cannot explain why English must insert *do* where it does. This grammatical residue must be attributed to something specific to the grammar of English, and has no possible source in a universal theory of word order. At best, all we can say from the current vantage point is which orders are typologically possible, and which are universally forbidden; the logic of individual languages' more intricate rule systems must be explained in other ways.

We have also introduced a tree notation equivalent to our surface structure bracketing, which itself can be understood as representing the Push and pop operations of stack-sorting. This notation will play a large role in the alternative generative formulation introduced in section 9. In the following section, we turn to another "exotic" movement phenomenon that challenges standard conceptions of syntactic movement: Icelandic Stylistic Fronting.

7 Stylistic Fronting

This section sketches an account of the syntactic phenomenon known as Stylistic Fronting (SF; Maling 1980/1990, Jónsson 1991, Holmberg 2000, 2006, Ott 2018, a.o.) in terms of our *231 stack-sorting architecture. This phenomenon, although relatively "exotic" (it is found in Icelandic, Faroese, and Old Scandinavian, but not in other modern Scandinavian languages), presents a number of features that are puzzling from the point of view of standard theories of phrase structure and movement. Three properties in particular stand out in this regard: (i) SF is optional and information-neutral; (ii) SF can apply to a broad array of syntactic objects, both head-like and phrasal, including verbal participles, adverbs, negation, and argument NP or PP, dependent on the presence or absence of other such categories; (iii) SF is contingent on not having an overt subject in the typical subject position. I show that all of these facts are consistent with the predictions of TTG. The exposition below relies heavily on the work of Ott (2018), from which I draw the majority of the examples I consider.

Since its first description by Maling (1980/1990), SF has received various analyses. Platzack (1987) proposes that SF is movement to Spec, TP. Rögnvaldsson & Thráinsson (1990) pursue a similar analysis, analyzing SF as topicalization movement to Spec, TP. According to the treatment of Holmberg (2000), in SF T agrees with the subject, but moves a different category to fill Spec, TP at PF. Jónsson (1991) proposes to treat SF as head movement, adjoining to T. Bošković (2004), in turn, argues that SF moves to a null affixal F head above T. Finally, Ott (2018) argues that SF involves remnant movement, which gives a way to treat the phenomenon as strictly phrasal movement, even when only an overt head appears to move.

7.1 The subject-gap restriction

In descriptive terms, SF gives the appearance of movement of other categories to the typical subject position, which must not contain the external argument. The subject-gap restriction closely parallels the **V S T* prediction in the TTG account of long head movement. Bošković (2004) ascribes this to an affixation requirement on a null F head. Other analyses try to derive the subject-gap restriction by moving the stylistically-fronted element into Spec, TP, with considerable problems (notably, that that position is presumably filled by a trace when the subject is A-bar moved).

The basic phenomenon of the subject-gap restriction can be seen in (68): the negation element *ekki*, which is otherwise capable of being stylistically fronted, cannot be fronted if the subject position is filled. Note that, unlike English *-n't*, analyzed as an instance of the relatively high head Pol, I treat Icelandic *ekki* as a negative adverb (in a rightward specifier position between the internal argument *O* and the external argument *S*).[15]

Subject-gap violation (Bošković 2004: 40, ex. 5b)
(68) *Ég held að **ekki Halldór** hafi séð þessa mynd. ...*1 **7 8 3** 2 5 4 6

[15] See Roberts (2019: chapter 7) for extensive discussion of different cartographic sites for negation. Although the arguments must be reexamined in light of the different assumptions in this account, it is clear that Icelandic *ekki* cannot be analyzed as an instance of Pol here.

I think that not Halldor has seen this film *2 3 1

(69) að -i haf -ð sé [þessa mynd] ekki Halldór *base order for (68)*
C T Aux -Fx V O NegAdv S
1 2 3 4 5 6 7 8

Note, though, that the subject gap can be the result of A-bar extraction, which should leave a trace, blocking movement to Spec, TP.

SF of V participle with A-bar extracted subject. (Ott 2018: 3 ex. 7a)

(70) **Hver** heldur þú [$_{CP}$ að **stolið**$_i$ hafi t_i hjólinu] ...C V -Fx Aux T O
who think you that stolen has the bike 1 5 4 3 2 6

Other ways to satisfy subject-gap restriction are to have a postposed (necessarily indefinite) subject, or an impersonal.

Late subjects: (Ott 2018: 4, ex 8b)

(71) **Keypt**$_i$ hafa t_i þessa bók **margir stúdentar** V -Fx Aux T O S
bought have this book many students 4 3 2 1 5 6

Impersonal: (Ott 2018: 4 ex. 9a)

(72) **Keypt**$_i$ hefur verið t_i tölva fyrir starfsfólkið V -Fx$_2$ Aux$_1$ T Aux$_2$ -Fx$_1$ O PP
bought has been a computer for the staff 6 5 2 1 4 3 7 8

In the present framework, the Subject-gap restriction parallels the **V S T* prediction for long head movement discussed in section 5. That is, SF places a medial element in the base order left of the higher *T* head; having a subject (later in the base order than either) between them would necessarily create the forbidden *231 permutation pattern.

7.2 Promiscuity of SF & Accessibility Hierarchy

One of the more curious properties of SF is that it seems to affect both heads, such as participles, and full phrases, such as argument NPs or PPs. This "promiscuity" is especially problematic from the point of view of standard approaches that draw a sharp distinction between head and phrasal movement. In our framework, however, this is actually an expected result. In (73-75), we see that with a verb taking a NP or PP argument, SF can affect the V, or the O/PP, but not both.

(73) Þeir sem **búið**$_i$ hafa t_i [$_{PP}$ í Ósló] 1 5 4 3 2 6
those that lived have in Oslo (Ott 2018: 17, ex. 43b)

(74) Þeir sem **[$_{PP}$ í Óslo]**$_i$ hafa búið t_i 1 6 3 2 5 4
those that in Oslo have lived (Ott 2018: 9 ex. 18d)

(75) *þeir sem **[$_{vP}$ búið í Óslo]** hafa t_i ... *...1 5 **4 6 3** 2
those that lived in Oslo have (Ott 2018: 10, ex. 22a)

The index sequences to the right of the examples above come from the following base order (76). Thus, our *231 principle makes the right predictions: movement of just the verbal participle (73), or just the argument phrase (74), produces a 231-free surface word order. Moving both together, however, produces a 231-like permutation, correctly ruling out (75).

(76) sem -a haf -ið bú [í Óslo] *base order for (68-70) above*
C T Aux -Fx V PP
1 2 3 4 5 6

If an adverb or negation is present, it can undergo SF, but will "block SF of *v*P-internal material" (Ott 2018: 12)

(77) a. þegar **búið**$_i$ var t_i að borða ... 4 3 2 1 5 6
when finished was to eat
b. þegar **ekki**$_i$ var t_i búið að borða ... 7 2 1 4 3 5 6
when not was finished to eat
c. *þegar **búið**$_i$ var ekki t_i að borða ... 4 3 2 1 7 5 6 ok
when finished was not to eat (Ott 2018: 13, ex. 29a,b)

(78) -r va -ið bú að borða ekki *base order for (77a-c)*
T Aux -Fx V T V NegAdv
1 2 3 4 5 6 7

Here, we correctly allow SF of the participle when no adverb or negation is present (77a), and SF of the negation (77b). Example (77c), ungrammatical in Icelandic, is actually 231-free, and so theoretically generable in our system. In general, this sort of thing is as expected. That is, we do not expect any language to allow the full range of possible surface order permutations. However, we expect that the ungrammaticality of (77c) arises from a different source (perhaps some surface-oriented predictive template specific to Icelandic) than (75), which is ruled out for all languages by the *231 principle.[16]

A PP can undergo SF, but only if negation is not present (79). If it is present, only negation can undergo SF (80b), blocking SF of the PP (80c).

(79) Þeir sem **í Danmörku**$_i$ hafa verið t_i ... 1 6 3 2 5 4
those that in Denmark have been

(80) a. Þeir sem hafa ekki verið í Danmörku ... 1 3 2 7 5 4 6

[16] Whether these examples have a different status for speakers of Icelandic is an interesting question for future research.

those that have not been in Denmark

b. Þeir sem **ekki**$_i$ hafa t_i verið í Danmörku ... 1 7 3 2 5 4 6
those that not have been in Denmark

c. *Þeir sem **í Danmörku**$_i$ hafa ekki verið t_i *...1 **6** 3 2 **7** **5** 4
those that in Denmark have not been (Ott 2018:13, ex. 30,31)

(81) sem -a haf -ið ver [í Danmörku] ekki *base order for (79-80)*
C T Aux -Fx V PP NegAdv
1 2 3 4 5 6 7

Given this base order for the above examples, we correctly generate the attested SF options and rule out SF of PP in the presence of negation (80c).

In verb-particle constructions, either the verb or the particle can undergo SF. Negation, if present, blocks both, being the only candidate for SF then. We correctly exclude particle movement in the presence of negation, but not movement of the participle. (Ott 2018: 24, ex. 57,58)

(82) a. fundurinn sem **fram**$_i$ hefur farið t_i ...1 6 3 2 5 4
the meeting that forth has gone

b. fundurinn sem **farið**$_i$ hefur t_i fram ...1 5 4 3 2 6

(83) a. *fundurinn sem **fram**$_i$ hefur ekki farið t_i ... 1 **6** 3 2 **7** **5** 4 *231
the meeting that forth has not gone

b. *fundurinn sem **farið**$_i$ hefur ekki t_i fram ... 1 5 4 3 2 7 6 ok

(84) sem -ur hef -ið far fram (ekki) *base order for (82-83)*
C T Aux$_1$ -Fx$_1$ V Ptcl NegAdv
1 2 3 4 5 6 7

In verb-particle constructions with an object, Extraposition of the object (*i.e.*, postposing of O to the end of the Surface String) is a necessary condition for SF to apply. This is correctly captured by the present proposal: the order Ptcl ... O V in the ungrammatical (85a) is a *231 permutation; Ptcl ... V O order in the grammatical (85b) is not.

(85) a. *þð var þa sem **ut**$_i$ voru [$_{NP}$ einhverjir kettir] reknir t_i ...*1 **6** 3 2 **7** **5** 4
it was then that out were some cats driven

b. þð var þa sem **ut**$_i$ voru reknir t_i [$_{NP}$ einhverjir kettir] ... 1 6 3 2 5 4 7
it was then that out were driven some cats (Ott 2018: 26, ex. 64)

Extraposition of the object is optional in verb-particle constructions without SF.

(86) a. þð var þa sem það voru [$_{NP}$ einhverjir kettir] reknir ut ... 1 ? 3 2 7 5 4 6
it was then that EXPL were some cats driven out

b. þð var þa sem það voru reknir ut [NP einhverjir kettir] ... 1 ? 3 2 5 4 6 7
it was then that EXPL were driven out some cats
(Ott 2018: 27 fn. 37, citing Thráinsson)

(87) sem -u vor -ir rekn ut [NP einhverjir kettir] *base order for (85-86)*
C T Aux_1 $-Fx_1$ V Ptcl O
1 2 3 4 5 6 7

Again, the simple *231 principle gets the facts right, given the base order in (87).[17]

7.3 Summary on Stylistic Fronting

Stylistic Fronting is an example of neutral word order variation, the intended explanatory target of this account. In general, we see that our framework correctly allows all attested SF configurations. At the same time, we rule out many (but not all) of the things SF can't do. This is as expected: the current approach aims to capture word order variation across languages, and as a result, overgenerates with respect to particular languages. Thus, not all restrictions on SF fall out here, the typical result. The crucial claim is that we do allow every neutral expression, in any individual language. This is indeed the case, for all instances of SF examined here.

In particular, some but not all aspects of the Accessibility Hierarchy are explained. We correctly capture the subject-gap restriction, which parallels the *VST prediction about long head movement derived in section 5. Most importantly, the account sheds new light on the "promiscuous" nature of SF, affecting both heads and phrases of a variety of categories. This is a central consequence of our approach: all neutral movement reflects *231 over a unified linear representation of the underlying hierarchy, including heads and phrases. The hierarchy-order mapping we have developed collapses all varieties of neutral head and phrasal movement to a single mechanism. Different movement possibilities for different syntactic categories follow from the invariant order of the base and the *231 theorem.

In the next section, I take up some various loose ends that have been left aside.

8 Some loose ends

This section takes us some matters that have been left hanging in the discussion so far, which deserve some comment from the present perspective.

8.1 Cycles/Phases

[17] I do not assign a base position, or corresponding index, to the expletive element *það*. This is because the base order is a representation of thematic structure, and I assume that the expletive is inserted to satisfy some (presumably language-specific) surface-oriented predictive pattern. Many interesting questions arise here, which must be put aside for future work.

We have kept throughout to a relatively simplistic view of phrase structure. In particular, we have avoided the topic of recursive embedding, outside of the treatment of cross-serial subject-verb dependencies in section 6. The astute reader may have noted that we ignored internal structure of argument NPs and PPs in the previous section, assigning them a single index and deep string position with respect to the clause.

A full treatment of the topic is beyond the scope of this paper. But it is immediately clear that some notion of cycle is required for this account to get the facts right. To see this, consider a classic problem for the Final-over-Final Condition: head-final VPs may embed head-initial DPs.

(88) *German*
Johann hat [$_{VP}$ [$_{DP}$ einen Mann] gesehen].
Johann has a man seen
'Johann has seen a man.' (Biberauer *et al* 2014)

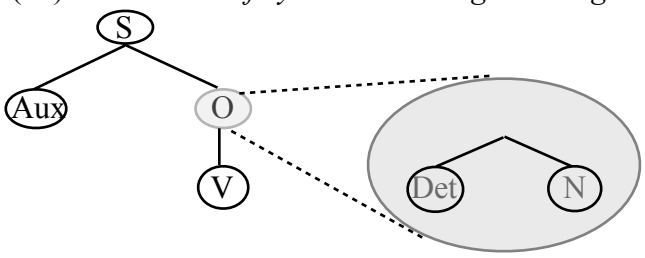
Simply treating all of the elements as part of a unified base order, [*Det N*] *V* should be a *231 order. Why, then, is it possible as a neutral order?

In the FOFC literature, the standard approach is to assume that the nominal is a separate cycle: a distinct hierarchy, or a different kind of extended projection. I adopt this solution, supposing that nominal and verbal cycles are disjoint for the purposes of the *231 condition.

One intriguing aspect of the present proposal is that a notion of *phase* is baked into the architecture. That is, the tree traversal algorithms, which take the place of Transfer in a standard Minimalist model, cannot apply at each step of incremental construction of the bare trees here. Instead, they must apply to whole trees, or subtrees, mapping a hierarchy onto them and reading off linear order. If this process is recursive (trees may embed references to already-transferred subtrees), further ordering predictions follow.

We can sketch how this would work for (88). Suppose a single node within the verbal cycle can contain a pointer to a separately-computed nominal cycle. The nominal subtree is generated, hierarchized, and linearized by itself, internally obeying the permutation-avoidance condition. But the internal structure of the nominal is unavailable, and irrelevant, within the embedding verbal cycle; its already frozen word order is "plugged in" at the corresponding node. Det-N order itself is 231-free, as is S-Aux-O-V.

(89) *Illustration of cyclic embedding allowing S Aux [Det N] V order*



Thus, we have two phases. The nominal *einen Mann* forms its own 231-avoiding cycle. Meanwhile, the embedding clausal cycle manipulates an atomic pointer to the nominal (O). Within the clausal cycle, the visible S-Aux-O-V order is 231-avoiding.

Many important questions arise at this point, especially about where such cycles are to be posited, under exactly what conditions. Here, the extensive discussion of this question in the FOFC literature is directly relevant. Further questions specific to this framework concern the effect of this additional machinery on possible global surface permutations. I set aside these topics for future work.

8.2 What about case and agreement?

Much recent work in generative theory has emphasized the roles of Case and agreement in driving syntactic operations. The reader may be left wondering about their role here, since they have not been mentioned at all. It is certainly notable that such effects play no explanatory role here in motivating syntactic movement, which has been radically reconceptualized.

In my (surely controversial) view, case and agreement are functionally-motivated E-language devices to support the identification of nominals with thematic argument positions in the base order, crudely and imperfectly. In the base order, and for the purposes of constructing the base tree via the Shift-Reduce semantic parser, different nominals are distinguished only by their structural context, in the familiar way. That is, the internal argument O is the complement of V; external argument S is the specifier of *v*P, and so on. Put another way, S and O are inherently NPs, whose status as S or O is defined only relative to the base structure they appear in, rather than by inherent properties of the items themselves.

Case and agreement are widely agreed to be uninterpretable at LF. As such, we expect that they do not appear in the base order, and have no role in the universal hierarchy-order mapping we are describing. Instead, we may hypothesize that they reflect surface-based templates (rooted in production-prediction feedback loops), which provide rough guides that mismatch the real structure in "exotic" constructions. We further expect that, where they occur, deviations from "proper" marking (which may not even be well-defined within the logic of the E-language) are in the direction of more local and more typical marking patterns in the language. I leave the matter here, though surely it is worthy of further pursuit.

In the next section, I present an alternative way of understanding the framework we have developed. Setting aside the idea of a stack-sorting universal parser, we see that an entirely equivalent account can be given as a generative model.

9 Tree Traversal Grammar

To this point, we have been describing our basic analytical tool, the *231 condition, as a consequence of a processing architecture. As sketched in section 2, this is conceived of as a two-stage parsing device, with an initial stack-sorting step rearranging surface word order into a common base order, which is then read by a Shift-Reduce semantic parser to build up a context-

free phrase structure representation. Notably, both the initial pre-processing step and the second-stage parser are language-invariant.

In this section, I describe an equivalent formalization. Here, we set aside issues of parsing, and instead construct a competence-level, generative account. To distinguish this from the stack-sorting perspective, we will call this formalization Tree Traversal Grammar (TTG).

Tree Traversal Grammar can be stated very simply. TTG employs two distinct generative devices: the Deep Grammar and the Surface Grammar. Related, we will be concerned with two different string forms for each sentence, the Deep String and Surface String. The Deep String, and its associated Deep Tree, is generated by the Deep Grammar, while the Surface Grammar generates ordered, *n*-ary branching Surface Trees. The Deep String is written onto the nodes of the freely generated Surface Tree in a sequence corresponding to postorder traversal of the tree. The Surface String is then read from the labeled Surface Tree by preorder traversal.[18]

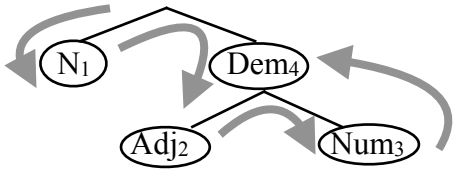
The notions of Deep and Surface Strings and Grammars correspond rather directly to the concepts we have been using within the parsing perspective. Specifically, the Surface String is the surface word order, while the Deep String is the base order. Likewise, the action of the stack-sorting algorithm is handled by the Surface Grammar, while the SR semantic parser which reads its output corresponds to the Deep Grammar.

9.1 Tree traversals

Tree traversals can be defined in terms of the priority of direction of travel from each node, with respect to three directions: Root (up to the dominating node), L (down to the leftmost daughter), R (down to the rightmost daughter). By providing an ordering of these three directions, we recursively define a method for traversing a tree, starting at the root. We make use of two standard tree traversal methods. The first, postorder traversal, is defined by the priority list (L, R, Root). Descriptively, postorder traversal visits nodes in a tree left-to-right and bottom-up. The other traversal method we will use is preorder traversal, which is defined by the priority list (Root, L, R). Descriptively, this visits nodes in the tree top-down and left-to-right.

As a first pass at how TTG works, I demonstrate its action for nominal order *N Dem Adj Num,* Cinque's (2005) order (p). The direction of postorder traversal is indicated by large grey arrows; subscript indices record the order in which the nodes are visited.

(90) *Postorder traversal*

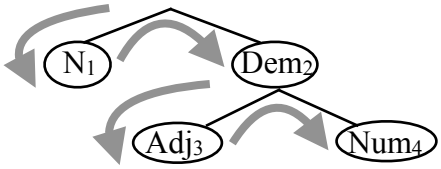


[18] Kural (2005) discusses tree traversal algorithms in the context of word order variation. Kural has traditional tree structures, including movement operations, and shows that some common word order patterns arise from traversing these trees in different ways, with different traversal algorithms corresponding to different word orders. The present proposal is different: the choice of traversal patterns does not vary between languages; rather, the two kinds of traversal are fixed but different for PF and LF, and what varies is the freely-generated Surface Tree to be traversed.

As shown, postorder visits the nodes in the order of the Deep String for this hierarchy; in this case, *N Adj Num Dem*. (See section 3 for much more on Universal 20.)

Once the tree has been labeled this way, linear order is read off by preorder traversal, which goes top down, left-to-right. The path of preorder traversal is shown with grey arrows in (91); this path visits the nodes in an order corresponding to the Surface String, *N Dem Adj Num*.

(91) *Preorder traversal*



9.2 Deep Grammar and Surface Grammar

The Deep String is a universally-linearized underlying form in head-complement-specifier order, generated by the Deep Grammar, a bog-standard phrase structure grammar. At this level, we reconstruct traditional notions of constituency and c-command in the base, all familiar and well-motivated.

The second, less familiar piece of the generative architecture is the Surface Grammar. This takes the head-complement-specifier Deep String as input, and freely generates any Dyck tree (see below) with a matching number of nodes. The Deep String is written onto this freely-generated Surface Tree in postorder, and the Surface String (the word order) is then read off the same tree in preorder, as illustrated in (90-91) above. Each tree has a distinct preorder (Surface String), but the same postorder (Deep String); this corresponds to the mapping of one meaning to many word orders in TTG.

Some surface orders cannot be generated; these are universally forbidden. Generated orders are universally available, but may or may not follow the ordering conventions of a particular language, predictively learned from experience. The claim is that the set of orders allowed in a given language is a subset of the set of universally allowed orders; that is, a language cannot exceptionally allow a universally-forbidden order as a surface form for an information-neutral expression. Figure 2 summarizes the architecture.

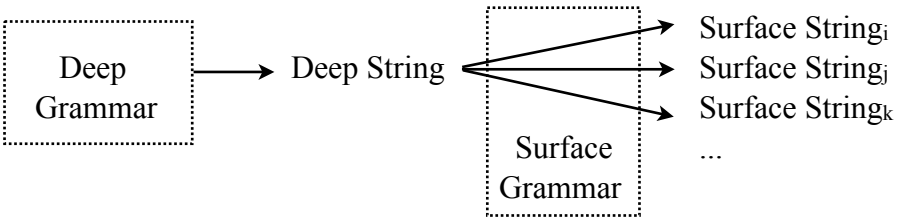


Figure 2. The generative architecture of Tree Traversal Grammar. The Deep Grammar builds a uniform Deep String, which is "refracted" through the different Surface Trees built by the Surface Grammar to produce an array of Surface Strings.

Notice that Figure 2 is very similar to the stack-sorting + SR architecture shown in Figure 1, except that the flow is reversed. That is, in the parsing perspective we start with a given surface word order, convert it into the base order, and build a base tree from it by SR parsing.

Here, we begin with a PSR generating the Deep String (base order), which is then transformed into the set of possible Surface Strings (word orders).

By hypothesis, any rooted, non-tangling, *n*-ary branching, ordered tree (Dyck tree) is a possible Surface Tree. These correspond 1-to-1 with legal bracketings: strings of left and right brackets in which, summing left-to-right, there are never more right brackets than left brackets, with equal numbers of each at the end. Figure 3 illustrates the correspondence, together with the corresponding order, explained below in (92):

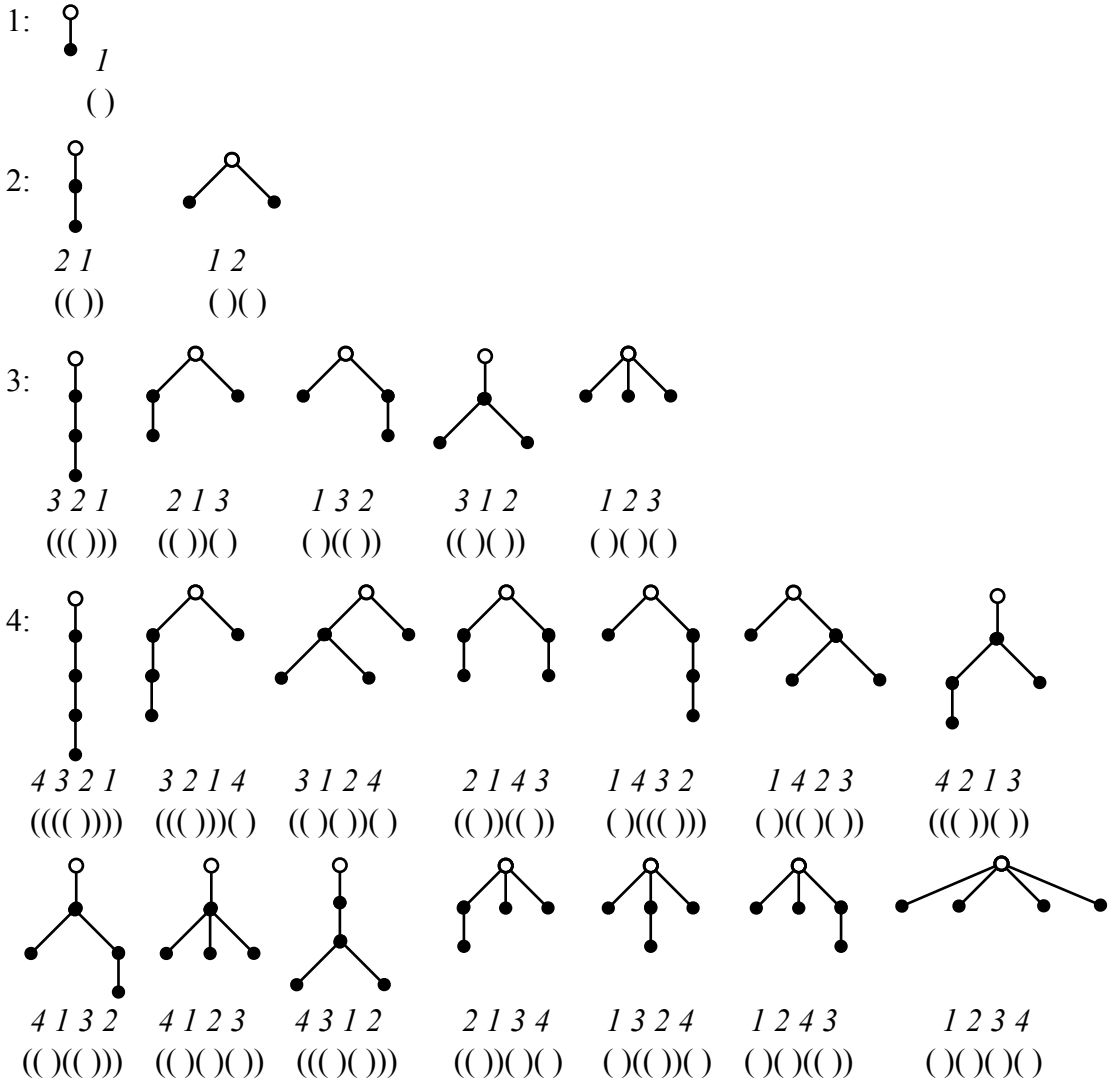


Figure 3: Legal bracketings (Dyck words) and possible Surface Trees (Dyck trees), with corresponding orders, for one, two, three, and four elements.

In terms of bracketed representations (recall that these are a notational variant of our *n*-ary branching trees), we can formulate the generative procedure as the following algorithm: (a) generate all legal bracketings, (b) label right brackets left-to-right with the Deep String, (c) copy

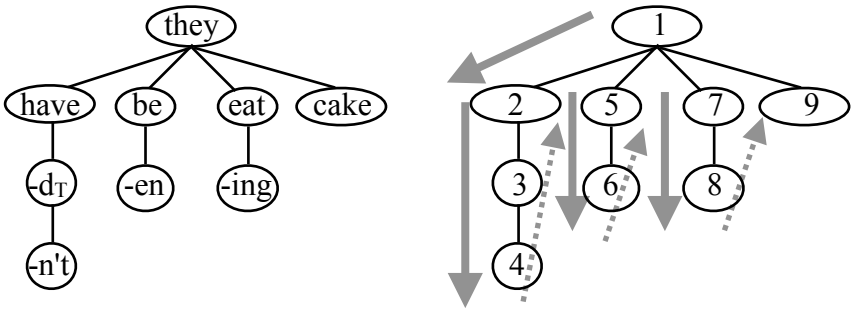
labels to matching left brackets, (d) read Surface String from left bracket labels. (92) illustrates; note that one logically possible permutation, *231, cannot be generated.

(92) *Generate orders from bracketings, for abstract Deep String 123*

a.	( )( )( )	(( ))( )	( )(( ))	(( )( ))	((( )))
b.	($_1$)($_2$)($_3$)	(($_1$)$_2$)($_3$)	($_1$)(($_2$)$_3$)	(($_1$)($_2$)$_3$)	((($_1$)$_2$)$_3$)
c.	($_1$ $_1$)($_2$ $_2$)($_3$ $_3$)	($_2$($_1$ $_1$)$_2$)($_3$ $_3$)	($_1$ $_1$)($_3$($_2$ $_2$)$_3$)	($_3$($_1$ $_1$)($_2$ $_2$)$_3$)	($_3$($_2$($_1$ $_1$)$_2$)$_3$)
d.	1 2 3	2 1 3	1 3 2	3 1 2	3 2 1 *2 3 1

Returning to the tree notation, I provide further illustration of the property that the Deep String (base order) is written by postorder traversal, and the Surface String (word order) is read by preorder traversal. We repeat our example of English Affix Hopping, *They hadn't been eating cake*. I show again the tree next to a copy of the same tree with individual nodes numbered in preorder. Grey arrows show the direction of the path.

(93)

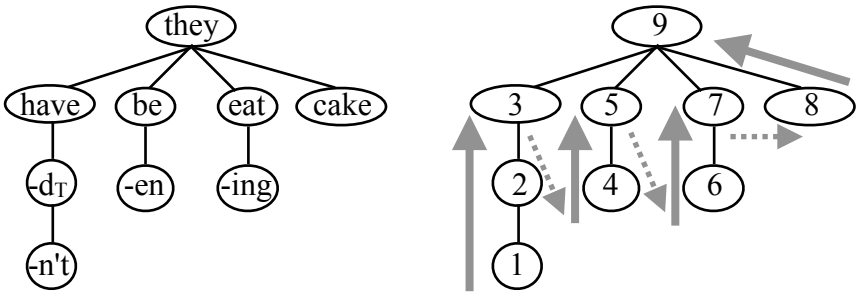


Assembling the words and morphemes from the nodes in the order visited by this path, we get string (94). As claimed, this is identical to the surface word order.

(94) They have -d -n't be -en eat -ing cake

And as for the second property, (95) shows the postorder traversal path.

(95)



Following this traversal path, we find the following sequence:

(96) -n't -d have -en be -ing eat cake they
Neg T Aux$_1$ -Fx$_1$ Aux$_2$ -Fx$_2$ V O S

There may seem to be something crucial missing from the account: we have said nothing about how the Surface Tree representations are derived. In a standard approach, this would involve specifying the base structure, and then positing movements to transform it, deriving the surface configuration step by step (in modern theories, applying Internal Merge operations interleaving with the External Merge operations constructing the base structure). The subject moves to specifier of TP, the auxiliary raises to T, some kind of morphological adjustment or syntactic lowering rule combines verbal stems and affixes, and so on.

In TTG, such devices are unnecessary. We do not need to distinguish and motivate head movement, and A-movement,[19] and the various phrasal movements deriving neutral word order variants. As a consequence, we also don't need features or other devices to drive such movements. Rather, given a realization of the underlying meaning as a Deep String, *231 is the only relevant (cross-linguistic) constraint on possible Surface Strings. Relatedly, it does not make much sense to think of elements moving to, or even being in, fixed structural positions such as Spec, TP, which are defined only in the Deep Tree.

TTG's Surface Trees do not represent constituents in the usual way; that is, units of meaning and continuous strings do not correspond to whole subtrees. Instead, constituency holds within the postorder traversal sequence (the Deep String), where a fairly trivial and language-invariant context-free phrase structure grammar (the Deep Grammar) suffices to build a standard LF structure Deep Tree, with inherent underlying head-complement-specifier order. Important questions remain about this conception of LF, especially about discourse-information related movement and its representation, not pursued here.

The architecture presents only a slight variation on the familiar assumption that LF and PF derive from the same pure-syntactic representation, but are interpreted by different processes. The standard implementation involves significant complications, especially with respect to chains created by movement: they must be pronounced in one position, while interpretation must read their scopal properties high, and their thematic properties low. Given a set structure built by traditional Merge, it is not trivial to identify chains (distinguishing them somehow from two independent instances of the same lexical form).

At least for the class of displacement considered here (i.e., information-neutral variations in word and morpheme order) the issues related to chains simply do not arise. Any *n*-ary branching Surface Tree is built, and each lexical element has a single, fixed position in that tree. However, hierarchization and linearization traverse the tree in a different order, and the linear order of lexical items within semantic and phonological string representations differs as a result.

9*.3 Where do the Surface Trees come from?*

This may still seem unsatisfactory. To the question of what is a possible Surface tree, TTG's answer resembles a zen *koan*: at the cross-linguistic level, by hypothesis, any *n*-ary branching tree is a possible Surface Tree. Even so, we are still left with the practical question of how to

---

[19] At least, insofar as A-movement is information-neutral. This would seem to be the case for something like subject-to-subject raising, but is much less clear for something like passivization. Many subtle and important questions arise at this point, which must be left for future research.

actually find the right tree for a given order, given an analysis of its underlying structure. It's easy to verify by inspection whether or not a Surface String is a 231-free permutation of the Deep String; if so, TTG generates it. But how do we draw the Surface Tree for a given order?

The procedure turns out to be fairly trivial, and to have a nice computational property of monotonicity. We will illustrate with the same English sentence we have been examining, *They hadn't been eating cake*. The first, crucial step is to segment and rearrange elements from the Surface String into the corresponding Deep String. We then index surface items according to their position in the underlying Deep String, as shown in (97).

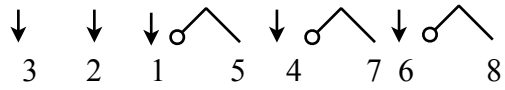
(97)

-n't	-d	have	-en	be	-ing	eat	cake	they
Pol	T	Aux_1	$-Fx_1$	Aux_2	$-Fx_2$	V	O	S
1	2	3	4	5	6	7	8	9

We then copy these indices onto the order in the Surface String, which lets us draw the tree. Compare indices of each pair of adjacent items in the Surface String *a*, *b*, writing $a < b$ or $a > b$ as appropriate. Each '<' corresponds to a horizontal relation between the respective nodes. If $a < b$, *b* attaches as sister to the highest-index node dominating *a* with lesser index than *b*. Meanwhile, $a > b$ indicates node *b* is immediately dominated by node *a*. Each adjacent comparison thus gives a "treelet", which can be assembled without ambiguity. (98) illustrates.

(98)

They	have	-d	-n't	be	-en	eat	-ing	cake
S	Aux_1	T	Pol	Aux_2	$-Fx_1$	V	$-Fx_2$	O
9	3	2	1	5	4	7	6	8
	>	>	>	<	>	<	>	<

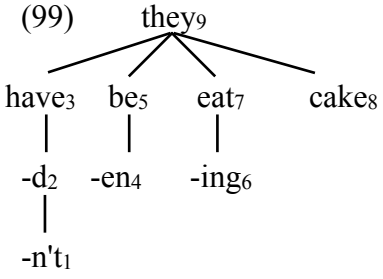


9 3 2 1 5 4 7 6 8

index sequence
< horizontal; > vertical
treelets

With these pairwise index comparisons, we produce a set of treelets, indicating how each node attaches to the partial Surface Tree formed by preceding elements in the Surface String. For the treelets in (98), we build the tree in (99). These are assembled incrementally in Surface String order. First, Aux *have* (index 3) attaches directly below external argument S *They* (index 9). Next, T *-d* is attached below the Aux, and in turn the polar negation *-n't* is below the T. The next category is progressive Aux *be*; it has higher index (5) than the Pol *-n't* (index 1), and also higher index than T *-d* (index 2) and Aux *have* (index 1), but lower index than S *they*. Thus, *be* attaches as a sister to *have* and a daughter of *they*. And so on; we end up with the tree in (99).

(99)



This, of course, is exactly the Surface Tree for this order; compare (93) and (95) above. Recall that the structure of a tree can be thought of as simply a list of pointers from parent nodes to child nodes. In these terms, the Surface Tree assembly procedure is strictly monotonic: it only adds new parent-child pointers to the partial list already assembled, never disrupting existing parent-child relations. In terms of tree geometry, we add new branches only at the bottom or right edge of the existing partial tree.

9.4 Possible Surface Strings grow much more slowly than logically possible orders

One concern about the minimal constraints on possible surface realizations of underlying structures in TTG is inherent in the many-to-one mapping of orders to meanings. Simply put, the number of possible Surface String realizations grows as the length of the Deep String increases. We can quantify this exactly: the number of possible Surface Strings for a Deep String of length *n* is the *n*th Catalan number, drawn from the sequence (1, 2, 5, 14, 42, 132, ...). However, this quantity grows much more slowly than the number of logically possible orders, which are counted by the factorial function $n! = n(n-1)(n-2)...(2)(1)$. (100) compares these quantities; the last column shows the *n*th Catalan number divided by *n*! as a percentage.

(100) Number of 231-avoiding orders (Cat) compared to possible orders of *n* items (*n*!)

n	Cat	*n*!	(Cat/*n*!) %
1	1	1	100%
2	2	2	100%
3	5	6	83.3%
4	14	24	58.3%
5	42	120	35.0%
6	132	720	18.3%
7	429	5040	8.51%
8	1430	40320	3.55%
9	4862	362880	1.34%
10	16796	3628800	.462%
11	58786	39916800	.147%
12	208012	479001600	.043%
13	742900	6227020800	.012%
14	2674440	87178291200	.0031%
15	9694845	1307674368000	.0007%

Thus, 231-avoidance is a rather weak condition for shorter sequences: it has no effect for Deep Strings of length 2 (*i.e.*, both logically possible orders are also 231-avoiding), and only rules out one of six logically possible orders of length 3. For four items, more than half of possible orders are generated by the TTG system (14 out of 24, as for Universal 20). However, as the length of the relevant strings increases, 231-avoidance becomes highly characteristic and

unlikely to arise by chance. For the simple English sentence (52) *They hadn't been eating cake*, we identified nine categories; at this size, only about 1% of possible strings are 231-avoiding.

This is likely significant in the context of acquisition, and the problem of identifying the categories of items segmented from the input stream. Observing many input strings containing the same elements in construction with various other elements, it should be relatively easy to narrow possible assignments of categories to such elements, as relatively few of them will satisfy the *231 condition. This is especially so when the same set of elements are observed in multiple possible orders. In other words, relative freedom of word order in a language may actually be of benefit to this aspect of language acquisition, a somewhat surprising conclusion.

9.5 Interim summary of TTG

This section has sketched another way of formulating the theory developed in this paper. Previously, we described the architecture as a universal parsing device, converting various word orders into a universal order and building a base tree from that. This implicitly identifies the grammar with the parser (Phillips 1996, 2001), a not uncontroversial proposal. In this section, we have shown that there is a formally equivalent notational variant, conceived of as a generative system. In this alternative, we begin by generating the base structure, then convert it into a set of possible surface structures and word orders by a write-read dual-traversal protocol over freely generated *n*-ary branching trees.

As we have seen, we end up with an entirely equivalent account of possible word orders and associated surface branching structures (which can themselves either be seen as strings of labeled bracket pairs, or as trees). That said, there may be other factors which favor one interpretation over the other. For example, the parsing model may better be able to accommodate Abels' 2016 data suggesting distinct "satellite classes" of verbs, as an interaction of cue-based retrieval interacting with recency effects, as discussed in section 2. On the other hand, the generative account abstracts away from real-time performance, and may be more useful for studying formal properties of the proposal.

10 Conclusion

This paper has introduced a novel framework for understanding neutral word order variation. The basic ingredients are twofold. First, we have claimed that there is a base structure shared by all languages, and that this structure is underlying ordered "heads first" (i.e., in head-complement-specifier order, to use familiar X-bar terms). Second, we have claimed that typologically possible word orders are the 231-avoiding permutations of the base order. We have focused on a realization of the architecture as a stack-sorting transducer feeding an SR machine, though we also introduced an alternative formulation fo the theory as a generative model involving tree traversals.

Some immediate questions arise, when presented with any new formalism for language structure. Does it overgenerate, creating many configurations not found in human language? Or do its inherent limits correspond in an interesting way to observed word order universals?

Next we must ask, does it undergenerate, failing to account for well-attested structures? A particularly thorny case is presented by constructions with cross-serial dependencies, which are a challenge for classic phrase structure theories.

Finally, and most importantly, we must ask about strong generation. Beyond merely generating attested orders and failing to generate unattested ones, does the formalism assign them the proper structure?

As we have seen, if we take the target of explanation to be the set of typologically possible orders (rather than the orders permitted in just a single language), the present account performs well on these metrics. We capture a range of word order universals, including Universal 20, the Final-Over-Final Condition, and a version of the Head Movement Constraint, in that orders violating these generalizations cannot be parsed (or generated) by our architecture. At the same time, we find that the system correctly allows a range of constructions that have been challenging to accommodate within other theories, including discontinuous constituency, cross-serial dependencies, head movement including apparent lowering, and so on. In certain cases, such as our treatment of long head movement, we appear to have improved on existing accounts in our empirical coverage. Importantly, we have done more than simply capture the proper range of string orders; we also automatically assign a surface structure representation to each allowed surface word order that corresponds very closely to existing descriptions.

Appealingly, all of these effects have been shown to follow from a single principle regulating the hierarchy-word order mapping: *231. This principle is itself a theorem, a necessary consequence of the framework. One could not keep the same basic mechanisms here while deriving any other principles of permitted and forbidden orders and structures.

This contrasts with contemporary accounts in which constraints on movement are extrinsic to the basic structure-building mechanisms that implement movement, especially where multiple devices are available to achieve a given surface output. Needless to say, a proliferation of devices for movement leads to a problem in acquisition: how is the child to distinguish between equivalent "movement" effects achieved by distinct modules? For example, observing a variation in word order, is this variation due to genuine syntactic movement, free linearization of a symmetric (*e.g.*, head-complement) structure without movement, PF movement, or morphological raising or lowering operations? The more mechanisms a theory admits to implement movement, the more severe this acquisition problem becomes. What we would like to find is that the problem dissolves, and given the basic properties of the constructions, there is little or no choice about how the surface form and underlying representation are related. Moreover, we hope to find the identity of the mechanism(s) involved to be determined by universal rather than language-particular principles.

The cost of covering all these phenomena in a single stroke is giving up on the dominant conception of Merge as unordered set formation, and Internal Merge as the engine of (neutral, at least) displacement. Certainly, we need recursive structure-building in syntax. And just as clearly, surface word order is not directly relevant to semantic interpretation. But the architecture described above shows that assuming an inherently ordered underlying representation simplifies and improves our understanding of the relationship between hierarchy and word order. At the same time, the present theory represents something of a retreat from understanding the details of individual languages, in particular with respect to how they select a subset of universally-

possible orders, placing a greater explanatory burden for this aspect of language variation on predictive learning during language acquisition. The payoff for this move is a conception of language acquisition that removes any need for variation in the underlying cognitive system, ascribing all variation in effect to learning from experience. Put simply, we are born knowing how to compute and comprehend language; what we learn is how to predict and produce a particular language. Much work remains to cash out this view, but it seems to me a promising direction to pursue.

References

Abels, K., & Neeleman, A. (2012). Linear asymmetries and the LCA. *Syntax* 15(1), 25-74.

Abels, K. (2016). The fundamental left–right asymmetry in the Germanic verb cluster. *The Journal of Comparative Germanic Linguistics* 19(3), 179-220.

Biberauer, T., Holmberg. A., and Roberts, I. (2014). A syntactic universal and its consequences. *Linguistic Inquiry* 45, 169-225.

Bošković, Z. (2004). PF merger in stylistic fronting and object shift. In *Minimality effects in syntax*, eds. A. Stepanov, G. Fanselow, and R. Vogel, 37–71. New York: Mouton de Gruyter.

Bresnan, J., Kaplan, R.M., Peters, S., & Zaenen, A. (1982). Cross-serial dependencies in Dutch. *Linguistic Inquiry* 13(4), 613-635.

Chomsky, N. (1957). *Syntactic Structures*. The Hague: Mouton.

Chomsky, N. (1981). *Lectures on Government and Binding*. Dordrecht: Foris.

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

Chomsky, Noam. (2007). Approaching UG from below. In Uli Sauerland & Hans- Martin Gärtner (eds), Interface + Recursion = Language? Chomsky's Minimal- ism and the View from Syntax and Semantics, 1–29. Berlin: de Gruyter.

Cinque, G. (1999). *Adverbs and Functional Heads: A Cross-linguistic Perspective*. New York: Oxford University Press.

Cinque, G. (2005). Deriving Greenberg's universal 20 and its exceptions. *Linguistic Inquiry* 36(3), 315-332.

Culbertson, J., Smolensky, P., and Wilson, C. (2013). Cognitive biases, linguistic universals, and constraint-based grammar learning. Topics in Cognitive Science 5 392–424.

Dékány, É. (2018). Approaches to head movement: A critical assessment. *Glossa: A Journal of General Linguistics* 3(1), 65.

Dryer, M. (2013). Order of subject, object and verb. In Dryer, M., & Haspelmath, M., (eds.), The World Atlas of Language Structures Online. Leipzig: Max Planck Institute for Evolutionary Anthropology. [http://wals.info/chapter/81, accessed 4 January 2021.]

Dryer, M. (2018). On the order of demonstrative, numeral, adjective, and noun. *Language* 94(4), 798-833.

Embick, D. & Izvorski, R. (1997). Participle-auxiliary word orders in Slavic. In Browne, W., Dornisch, E., Kondrashova, N., & Zec, D. (eds.), *Annual workshop on Formal Approaches to Slavic Linguistics: The Cornell meeting*, 210–239. Ann Arbor, MI: Michigan Slavic Publications.

Greenberg, J. (1963). Some universals of grammar with particular reference to the order of meaningful elements. In Greenberg, J. (ed.), *Universals of language*, 73-113. Cambridge, MA: MIT Press.
Harizanov, B., & Gribanova, V. (2019). Whither head movement? *Natural Language & Linguistic Theory* 37(2), 461-522.
Haspelmath, M. (2007). In Shopen, T. (ed.), *Language typology and syntactic description Vol. II: complex constructions*, 1-51. Cambridge, MA: Cambridge University Press
Hauser, M., Chomsky, N., and Fitch, T. (2002). The faculty of language: what is it, who has it, and how did it evolve? *Science*. 2002 (5598):1569-79.
Henson, R. (1998). Short-term memory for serial order. The start-end model. *Cognitive Psychology* (36), 73–137.
Holmberg, A. (2000a). Deriving OV order in Finnish. in Svenonius, P., (ed.), *The Derivation of VO and OV*. Philadelphia: John Benjamins.
Holmberg, A. (2000b). Scandinavian stylistic fronting: How any category can become an expletive. *Linguistic Inquiry* 31:445–484.
Holmberg, A. (2006). Stylistic fronting. In *The Blackwell companion to syntax*, eds. M. Everaert and H. van Riemsdijk, 532–565. Oxford: Blackwell.
Hrafnbjargarson, G. H. (2004). Stylistic fronting. *Studia Linguistica* 58:88–134.
Huybregts, R. (1976). Overlapping dependencies in Dutch. *Utrecht Working Papers in Linguistics* 1:24–65.
Jónsson, J. (1991). Stylistic fronting in Icelandic. *Working Papers in Scandinavian Syntax* 48:1–43.
Joshi, Aravind (1985). "How much context-sensitivity is necessary for characterizing structural descriptions". In Dowty, D., Karttunen, L., and Zwicky, A. (eds.). *Natural Language Processing: Theoretical, Computational, and Psychological Perspectives.* New York: Cambridge University Press, 206–250.
Kayne, R. (1994). *The Antisymmetry of Syntax*. Cambridge, MA: MIT Press.
Koopman, H., and Szabolcsi, A. (2000) Verbal Complexes. Cambridge, MA: MIT Press.
Kural, M. (2005). Tree traversal and word order. *Linguistic Inquiry* 36(3):367-387.
Laka, I. (1990). *Negation in Syntax: on the nature of functional categories and projections*. Ph.D. dissertation, MIT.
Lema and Rivero, M.-L. (1991). Types of verbal movement in Old Spanish: Modal & futures, and perfects. *Probus* 3(3).
Lenneberg, E.H. (1967). *Biological foundations of language*. Wiley.
Lewis, R. L., & Vasishth, S. (2005). An activation-based model of sentence processing as skilled memory retrieval. Cognitive Science, 29, 375–419.
Lewis, R. L., Vasishth, S., & Van Dyke, J. A. (2006). Computational principles of working memory in sentence comprehension. Trends in Cognitive Sciences, 10, 44–54.
Maling, J. (1990). Inversion in embedded clauses in Modern Icelandic. *Modern Icelandic syntax*, eds. J. Maling and A. Zaenen, 71–91. San Diego, CA: Academic Press.
McCloskey, J. (2017). Ellipsis, polarity and the cartography of verb-initial orders in Irish. In Aboh, A., Haeberli, E., Puskás, G., & and Schönenberger, M., (eds.), *Elements of Comparative Syntax: Theory and Description*, 99–151. Berlin: De Gruyter.

McElree, B. (2000). Sentence comprehension is mediated by content-addressable memory structures. *Journal of Psycholinguistic Research* (29), 111–123.
Medeiros, D.P. (2018). ULTRA: Universal Grammar as a Universal Parser. *Frontiers In Psychology: Language Sciences* 9.
Nchare, A. (2011). The Syntax of Agreement in Shupamem and Greenberg's Universal 20. *NYU Working Papers in Linguistics* 3, 136-198.
Nchare, A. (2012). *The Grammar of Shupamem*. Ph.D. thesis, New York University.
Ott, D. (2018). Stylistic Fronting as Remnant Movement. *Studia Linguistica* 72(1): 1-38.
Pearson, M. (2000). Two types of VO languages. In Svenonius, P. (ed.), *The derivation of VO and OV.* Philadelphia: John Benjamins.
Perlmutter, D. (1970). Surface structure constraints in syntax. *Linguistic Inquiry* 1(2), 187-255.
Platzack, C. (1987). The Scandinavian languages and the null-subject parameter. *Natural Language and Linguistic Theory* 5:377–401.
Rivero, M-L. (1991). Long head movement and negation: Serbo-Croatian vs. Slovak. *The Linguistic Review* 8(2–4). 319–351.
Rizzi, Luigi. (1990). Relativized Minimality. Cambridge, Mass.: MIT Press.
Rizzi, L. (1997). The fine structure of the left periphery. In Haegeman, L. (ed.), *Elements of Grammar: Handbook of Generative Syntax*. Dordrecht: Kluwer.
Roberts, I. (2010). *Agreement and head movement: clitics, incorporation, and defective goals*. Cambridge, MA: MIT Press.
Roberts, I. (2017). The Final-over-Final Condition in DP: Universal 20 and the Nature of Demonstratives. In Sheehan, M., Biberauer, T., Roberts, I., & Holmberg, A. (eds.) *The Final-Over-Final Condition: A Syntactic Universal* (Vol 76), 151-186. Cambridge, MA: MIT Press.
Roberts, I. (2019). *Parameter Hierarchies and Universal Grammar*. New York: Oxford University Press.
Rögnvaldsson, E. & Thráinsson, H. (1990). On Icelandic word order once more. *Modern Icelandic syntax*, eds. J. Maling and A. Zaenen, 3–40. San Diego, CA: Academic Press.
Sheehan, M., Biberauer, T., Roberts, I., & Holmberg, A. (eds.). (2017). *The Final-Over-Final Condition: A Syntactic Universal* (Vol. 76). Cambridge, MA: MIT Press.
Shieber, S.M. (1985). Evidence against the context-freeness of natural language. *Linguistics & Philosophy* 8(3), 333-344.
Stabler, E. (2004). Varieties of crossing depemdencies: structure dependence and mild context sensitivity. *Cognitive Science* 28: 699-720
Steddy, S., and Samek-Lodovici, V. (2011). On the ungrammaticality of remnant movement in the derivation of Greenberg's Universal 20. *Linguistic Inquiry* 42 445–469.
Steedman, M. (2000). *The syntactic process* (Vol. 24). Cambridge, MA: MIT press.
Steedman, M. (2020). A formal universal of natural language grammar. *Language* 96(3), 618-660.
Townsend, D., and Bever, T.G. (2001). *Sentence Comprehension: The Integration of Habits and Rules*. Cambridge, MA: MIT Press.
Travis, L. (1984). *Parameters and effects of word order variation*. Ph.D. Dissertation, MIT.
Zwart, Jan-Wouter. (2009). The relevance of typology to minimalist inquiry. *Lingua* 119: 1589-1606.