Modular Minimalist Grammars – Part I

Edward P. Stabler*
stabler@ucla.edu
draft 2024-10-12

Abstract

Inspired by recent syntactic theory, minimalist grammar is modularized, simplified and extended here. Phrasal, 'core' syntax can be defined by one simple rule, with head movement, linearization, morphology and prosody separated and applied as transductions. When composed, these transductions can all apply simultaneously, or nearly so, to provide an incremental structural analysis. Some mutual interactions among modules is evidenced, but these breaches of modularity are restricted. They affect the granularity of synchronization but do not block the computation of incremental analyses with simple chart-free bottom-up methods. For illustration, a range of small but challenging examples are treated, with particular attention to head movement, relativized minimality and realizational morphology.

> I have had many difficulties to contend with in writing this . . . one of these is the proper arrangement of my chapters, inasmuch as the subjects they deal with interlock and overlap in the most bewildering way. – Jespersen (1924, p.10)
>
> . . . when faced with different concerns, we should separate them as completely as possible, and deal with them in turn. – Dijkstra (1982, p.324)
>
> For instance, you are asked what will have to be paid for six pounds of sugar at 3d. a pound. You multiply the six by the three. That is not because of any property of the sugar, or of the copper of which pennies are made. – Boole (1909, p.1)
>
> . . . much of Mathematics is dynamic, in that it deals with morphisms of an object L into another object of the same kind. – Mac Lane (1986, p.359)
>
> The point isn't just succinctness for the sake of saving space, it's about packing things up so that we can carry them around better. – Cheng (2023, p.176)

Early versions of minimalist grammars (MGs) have specialized rules for for first and non-first merges, for internal and external merge, and for internal and external merges with an element that is going to move again. Common subcomponents of each case are simply repeated (Stabler, 2011, 1997; Michaelis, 1998; Harkema, 2001). As suggested by Hunter (2011), Graf (2013), Kobele (forthcoming), Stabler and Yu (2023) and others, following similar directions in mainstream syntax, a simpler conception can be obtained by composing those rules from more basic parts, and by separating linearization and other largely distinct, 'post-syntactic' processes. A modular approach inspired by those proposals is formulated here, and the ease of generalization it allows is illustrated by extensions for unbounded branching, multiple wh movement, relativized minimal search, head movement and m-merger. Developing this perspective, some prominent recent proposals are explored which adjust the framework and breach the simple modularity one might have hoped for, making some processes both 'post-syntactic' and 'pre-syntactic'. Even with these complications, linguistically sophisticated performance models become conceivable. The modules can be composed so that, intuitively, analyses are extended as soon as the relevant ingredients become available.

* Parts of this were presented at the MG+2 meeting hosted by Stony Brook University. Thanks to the audience there and especially Meaghan Fowlie, Thomas Graf, John Hale, Greg Kobele, Miloš Stanojević, Kristine Yu.

1 Modular minimalist grammar

This work is inspired by an approach to linguistic theory that aims to derive many facts about human languages as minor embellishments of one basic combinatorial operation. Stabler (1997) defines derivations in six cases, each of which is rather complex. The first case presented there merges a head with a non-moving phrase it selects, matching and unlinking features. The similarity of all six cases indicates missed generalizations. And as recent work has extended minimalist grammars – following similar development in syntactic theories of head movement, adjunction, agreement, etc. – the number and complexity of cases increases quickly. This makes it hard, as a practical matter, to design linguistically sophisticated performance models. Understanding the whole theory becomes difficult. It also raises questions about linguistic and psycholinguistic plausibility. How can language learning toddlers manage to parse and pronounce structures that are definitionally so complex? What would a descriptively adequate model even look like?

In §1.1, as in Kobele (forthcoming), generative rules of earlier minimalist grammars are simplified by decomposing them into smaller parts, and then re-composing a single, more general rule. That rule derives 'workspaces'. Here, workspaces simply pair syntactic structures (and possibly some active substructures of those structures) with linguistically, derivationally relevant labels. Each derived workspace is, in effect, a moving window on a single active, developing structure, holding all the parts of the derived structure that may still extend their merge relations. Each workspace holds structures 'connected' by merge in this way. We begin with a match rule that only accommodates binary structures. In §1.2 additional properties are defined 'post-syntactically', in terms of their specific dependencies. These 'embellishments' can be computed one after another, with relatively independent properties specified relatively independently. And §1.3 extends minimalist grammars to provide preliminary analyses of unbounded external merge for coordination and unbounded internal merge for multiple-wh movement.

In all these derivations, every workspace is determined by the structure that is its head, and so the workspaces are really inessential. Consequently, in §1.2.1, an easy shift of perspective makes syntactic objects the products of the derivation, not workspaces. Workspaces are computationally convenient and finitely limited carriers of information about earlier calculations, as the name 'workspace' implies. This shift of perspective sets the stage for composing additional modules that slightly elaborate the merged structures, so that final structures can be built in one traversal. The simple picture of modules applied in sequence is slightly disrupted in §1.4, where we capture some recent proposals in which some post-syntactic mechanisms become also pre-syntactic. The modules become mutually recursive. Finally, in §2, some first steps are taken towards mechanisms that might underpin linguistic abilities. Extremely simple algorithms are capable of handling all the syntax and interfaces proposed in this paper.

1.1 Merge as the structure building operator

Partly just to clarify some relations between earlier grammars and the more recent ideas of §1.3, it is useful to recast earlier minimalist grammars in a more modular form.

1.1.1 Lexical items and grammars

Assume, to begin, that basic properties of expressions are indicated with 'selection' features: noun (N), verb (V), adjective (A), preposition (P), Wh-element (Wh),... To form a phrase, a verb V might need two determiner phrases (DPs), so we represent the requirements with the sequence D.D, using a dot to represent conjunction in a conditional written D.D $\multimap$ V. For uniformity, when there are no requirements, we assume there are conditionals with 'empty', trivially true antecedents, which we indicate with $\top$. So a noun with no requirements is labeled with the implication $\top \multimap$ N. For readability, we will often just leave out the $\top$ and $\multimap$, writing just N. And finally, an implication with no features at all, $\top \multimap \top$, will usually be written $\top$. Labels $\top$ with no features are syntactically inert.

The *labels* assigned to lexical items are all *implications*, $\alpha \multimap \beta$, where the antecedent α is a possibly empty conjunction of selection features for elements that are required, and the consequent β is a non-empty conjunction of features that the lexical item then possesses. So $\alpha \multimap \beta$ means roughly: 'if elements with the features α are merged, then the result has the features β'. The features in antecedent α are *negative*,

while those in consequent β are *positive*, as in the propositional calculus:[1]

$$\begin{array}{l}(\mathrm{f}_1 \wedge \ldots \wedge \mathrm{f}_i) \rightarrow (\mathrm{f}_{i+1} \wedge \ldots \wedge \mathrm{f}_{i+j}) \equiv \\ \neg(\mathrm{f}_1 \wedge \ldots \wedge \mathrm{f}_i) \vee (\mathrm{f}_{i+1} \wedge \ldots \wedge \mathrm{f}_{i+j}) \equiv \\ \neg\mathrm{f}_1 \vee \ldots \vee \neg\mathrm{f}_i \vee (\mathrm{f}_{i+1} \wedge \ldots \wedge \mathrm{f}_{i+j}) \equiv \\ \quad (\neg\mathrm{f}_1 \vee \ldots \vee \neg\mathrm{f}_i \vee \mathrm{f}_{i+1}) \wedge \\ \quad (\neg\mathrm{f}_1 \vee \ldots \vee \neg\mathrm{f}_i \vee \mathrm{f}_{i+2}) \wedge \\ \quad \ldots \\ \quad (\neg\mathrm{f}_1 \vee \ldots \vee \neg\mathrm{f}_i \vee \mathrm{f}_{i+j}).\end{array}$$

As we will see in §1.2.1, a single *modus ponens* rule applies to labels. This differs slightly from standard propositional calculus, so we use a dot . and $\multimap$ instead of $\wedge$ and $\rightarrow$.[2] Rather than labels like

$$(\mathrm{f}_1 \wedge \mathrm{f}_2 \wedge \mathrm{f}_3) \rightarrow (\mathrm{f}_4 \wedge \mathrm{f}_5 \wedge \mathrm{f}_6),$$

we have

$$\mathrm{f}_1.\mathrm{f}_2.\mathrm{f}_3 \multimap \mathrm{f}_4.\mathrm{f}_5.\mathrm{f}_6.$$

Features are matched and checked left to right: negative features (if any) first, then positive features.[3]

Let's also say that a lexical item is *positive* iff its label has a non-empty antecedent, so it has positive features. And a lexical item is negative iff it is not positive.

To set the stage for later developments, let's tentatively assume that each lexical item also can have person, number and gender features ϕ and case features κ. These can be complex, but here it will suffice to represent these with tags of the sort found in conventional glosses. For example, ϕ might be one of 1S, 3PL,... (i.e. 1st person singular, 3rd person plural,...), and κ might be one of NOM, ACC, ERG, ABS,... (i.e. nominative, accusative, ergative, absolutive,...).

Finally, a *vocabulary* Σ is a set of elements that somehow specify phonological formatives. These too may be complex but will tentatively just be represented with a string. Initially, we will indicate phonological content with standard English orthography, but allow ϵ in addition for an empty, silent specification.

A *lexical item* is a pair (w, features), where features is a pair (label, agr), where label is a pair (antecedent, consequent) and agr is a pair (ϕ, κ). Rather than writing (w,(α, β), (ϕ, κ)), we will write the slightly more readable (w,$(\alpha \multimap \beta)$)-ϕ-κ, and we will simply leave ϕ-κ off when they are empty or not relevant, rather than writing (w,$(\alpha \multimap \beta)$)-ϵ-ϵ. A *minimalist grammar* (MG) is a finite set of such lexical items, a lexicon.

Example. This grammar of 10 items is enough to illustrate some basics:

(1)				
	(ϵ, V $\multimap$ C)	(ϵ, V.Wh $\multimap$ C)		
	(Jo, D)	(the, N $\multimap$ D)	(which, N $\multimap$ D.Wh)	(who, D.Wh)
	(cat, N)	(food, N)		
	(likes, D.D $\multimap$ V)	(knows, C.D $\multimap$ V)		

In this grammar, we have not specified any ϕ or κ features. And the elements in the first line are not pronounced. That first element on the first line says that given a V phrase (a verb phrase, VP), the unpronounced ϵ makes a C phrase (a complementizer phrase, CP). The second pair on the first line says that, a VP and a Wh phrase can make a CP with an unpronounced ϵ. The 4 items in the next line say: *Jo* is

[1] These equivalences show that an implication between conjunctions of features corresponds to a set of propositional 'Horn clauses'. That is, each label corresponds to a conjunction of disjunctions, in which each disjunction has at most one positive disjunct. A connection between these formulas and context free grammars is described in Dowling and Gallier (1984), and Horn clause inference in various other logics is described in Lloyd (1984), Kanovich (2015).

[2] The . is not commutative. And there is no negation or disjunction or thinning or contraction. Compare '$\multimap$ elimination', the Law of Detachment, in various resource logics: Troelstra (1992, §6), Kanovich (2015).

[3] Notice that this representation reverses Kobele's names of the polarities. Kobele (forthcoming, 6) says: "The *locus principle* (attributed by Collins (2002a) to Chomsky (2000)) is one such, stating that (in our terms) no negative feature may be used if there are positive features available." Graf (2022a, 2023) proposes that features should be unordered, an idea left for future work.

a determiner phrase, a DP, and it is positive; given an NP, *the* forms a DP, so this item is negative; *which* takes an NP to make a phrase that is both a DP and a Wh phrase; and *who* is both a DP and a Wh phrase. The final two lines have entries N and V. Functions that will derive infinitely many CPs from this grammar, like the one pronounced *Jo knows which food the cat likes*, will be in the introduced next sections.

We split lexical features into 4 sorts – phonological features w, selection features in the labels, agreement features ϕ, and case features κ. More structure is certainly needed, but this simple idea will suffice for present purposes.[4]

1.1.2 Merge

Sets, trees and multidominance structures. In a computational implementation, the data is represented and transformed. What we will represent is relations among occurrences of lexical items. Collins and Stabler (2016) call them the 'tokens' of lexical items, and Chomsky et al. (2023) call them 'inscriptions':

> . . . lexical items are drawn as needed from the Lexicon and inscriptions of them are available for computation. Thus, a lexical item such as the noun *child* can be selected, and as many inscriptions of it as might be needed can be available, allowing such sentences as *One child slept while a second child played with another child*. Similarly, in mathematics there are multiple inscriptions of, say, the numeral three in an equation like 3x + 3y = 3. (p.2)

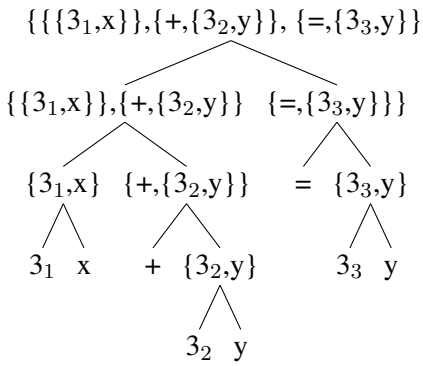
But, in the more common and better known terminology, it is more natural to say that there are multiple *occurrences* of the numeral 3 in '3x + 3y = 3'.[5] If we count positions and ignore the spaces, then the numeral 3 occurs in positions 1, 4, and 7, and we could refer to those occurrences as $3_1, 3_4$ and 3_7. Alternatively, we could just index those elements in order, $3_1, 3_2$ and 3_3.

Beginning with the elements of arithmetic formulas, we can build sets of those elements, and set of sets of those elements, and so on. In that way, we could parse the elements of '3x + 3y = 3' into semantically coherent subparts, distinguishing the respective occurrences of 3 like this:

(2) $\{\{\{3_1,x\}\},\{+,\{3_2,y\}\}, \{=,\{3_3,y\}\}\}$

A diagram of this set can make the relationships clearer. We can make a picture of the is-an-element-of, $\in$ relations, that define that set. Here is the membership graph of the set (2):

(3)



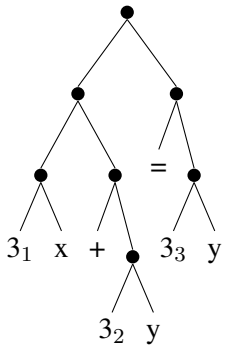
This is a tree, an unordered tree, since left-right order in the diagram does not matter. The tree is defined by pairs in the is-a-member-of relation, the $\in$ relation. It has 7 leaves which are the symbol occurrences. And it has 7 internal nodes, which are sets, for a total of 14 nodes.

Notice that the labels at the internal nodes are redundant in the previous diagram, since the graph itself shows what the elements of each set are. So we can represent the same structure of 14 nodes without showing the labels of internals nodes – we understand that each internal node is the set of the elements immediately below it. Here is set (2)=(3) again, more concisely:

[4] While verbs V select complements of different categories, functional categories like T and v are often assumed to exhibit little or no variation in the complements they require, so some linguists propose treating them differently. Each head could be associated with a number of functional elements with fixed requirements – its 'extended projection' (Abney, 1987; Grimshaw, 1991; Adger, 2010; Ramchand and Svenonius, 2014; Pietraszko, 2023a). A special kind of selection feature sharing could then allow minor variations in the requirements of these categories. But in the present framework, extended categories and other hierarchy universals (Cinque, 1999; Ramchand and Svenonius, 2014) would be formulated as lexical invariants. One issue here and in most traditional hierarchies is that D seems never to be selected (Thoms, 2022; Sportiche, 2005; Aboh, 2024; Bruening, 2020). A different kind of worry, discussed in §1.3.5, is that single lexical items can also have multiple phonological forms – allomorphs – (Embick and Noyer, 2001; Hewett, 2023; Murphy and Offer-Westort, 2024; Kalin and Rolle, 2024).

[5] On occurrences, and the related notions of type and token, see Wetzel (1993, 2009).

(4)



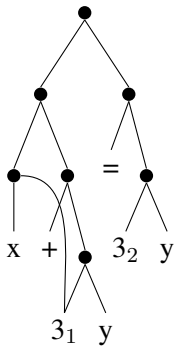
(4) is just another representation of the same set, (2)=(3) = (4), easier to read.

Since, in arithmetic, the different occurrences of 3 all refer to the same number, the previous tree is not arithmetically different from this one, in which we use 3_1 twice:

(5) {{{3_1,x}},{+,{3_1,y}}, {=,{3_2,y}}}

If we draw that set in the same way, indicating is-a-member-of relations with downward arcs, we see that the set is not a tree. It is rooted and directed and acyclic, but one element has two parents. The diagram of set (5) is this:

(6)



Like a tree, this graph is a directed, rooted, acyclic graph. But it differs from a tree in that one node has 2 mothers, so it is a 'multidominance structure'.[6] The difference between (4) and (6) does not matter in arithmetic.

In the syntax of human languages, distinctions among occurrences *do* matter. When some element occurs more than once in a tree, that can affect the pronunciation and semantic interpretation at the interfaces, and it affects what is available to 'move' in a syntactic structure, as discussed below.

In the syntax defined here, distinctions among occurrences will not usually be marked with indices, but will be determined by their structural context. Indexed elements are sometimes useful in discussions *about* syntactic structures and derivations, but they are not needed in the syntax itself. Syntactic mechanisms are defined by properties of expressions that do not include whatever properties correspond to occurrence-indices. That makes the syntax simpler, since standard indexing is defined with reference to the whole derivation – something that no operation in the syntax ever needs to be sensitive to. If we wanted to add a new occurrence of 3 to both sides of equation (2), we would want to avoid indices 0-2 to avoid confusion with previously indexed occurrences. But in minimalist grammar derivations, that kind of derivation-global mechanism should not be needed; syntactic operations are local. I agree with Chomsky (1995, 199n3): "Indices are basically the expression of a relationship, not entities in their own right. They should be replaceable without loss by a structural account of the relation they annotate."[7]

[6]Since the multidominance structures defined by set-membership lack left-right order, they are simpler than some other multidominance structures in the linguistic literature (Gärtner, 2002; Fowlie, 2011; Gärtner, 2014; Citko, 2011). The axiom of foundation in standard set theory (Levy, 1979; Kunen, 1980) ensures that no set S can contain S or any an element that contains S. That property is what we want here too. Similar structures can be established in the category of small sets (Shulman, 2008). Of course it is possible to define set-like objects with cycles (Moss, 2018), but they will not be relevant here.

[7]Three different ways to mark identical variables with the same scope are common in the literature: explicit coindexing to

Merge. Let merge be the function mrg that puts $n \geq 0$ distinct syntactic objects (SOs) together into a set. Given any alphabet V and features F, a (possible) lexical item is an element a (word,label) pair, an element of $(V^*, F^* \multimap F^*)$. A syntactic object SO is either a (possible) lexical item or any finite set built from those (possible) lexical items. The domain of the mrg is all tuples or sequences of SOs – it is the closure of all possible lexical items with respect to mrg. The mrg function maps any finite sequence of SOs to the (multi)set of those SOs:

$$\frac{SO_1, \ldots, SO_n}{\{SO_1, \ldots, SO_n\}} \text{ (mrg).}$$

With this formulation, mrg is a function from its domain into that same domain.

Note that the definition of mrg is not recursive, in the sense that mrg does not appear in its own definition. This is compatible with the commonly noted fact that mrg is recursive in the different sense that it can apply to its own results, and the fact that the structures in its domain and range are recursive in the sense that (multi)sets can contain (multi)sets.

The domain of mrg can be ranked. Rank 0 is the set of lexical elements that are not sets. Rank n>0 is the union of rank n-1 together with all subsets of rank n-1. The domain of mrg is the union of all those (finite) ranks, an infinite set of finite objects.[8]

When a syntactic object SO={X_1,...,X_i}, we say that each X_i is an *element* of SO, a *child* of SO. X is *contained* in Y if either X is an element of Y or X is an element of a set contained in X. Multiple occurrences of X may be contained in Y – if this happens and we want to specify one of them, this can be done by specifying a particular path from the root to the intended occurrence of X (Collins and Stabler, 2016, p.51). When (occurrence) X is contained in (occurrence) Y, we sometimes also say that X (properly) *dominates* Y, and that X is *in* or *inside* of Y, a *substructure* of Y.

1.2 Interfaces for the domain of merge: Binary formulation

1.2.1 Labeling and workspaces for binary structures

Here, a workspace is simply an association between SOs and labels. Each workspace has a largest SO, the head, with a non-empty label, possibly together with some substructures of that SO that also have non-empty labels. The workspace is eliminable, in the clear sense that the whole workspace is a function of its head SO. But it is convenient in calculating the labels of a complex to have relevant labels of its substructures available. As noted above, the workspace is a window, a focus of attention, on a head SO and any substructures it has that are derivationally relevant.

make identifiers globally unique (here, 'inscriptions' or indices in the glosses), implicit contextual representation (here, made explicit in our workspace labels), and explicit graphs (here, in our depictions of multidominant graphs). In the syntax we propose below: an occurrence of a constituent c in SO is a copy iff c and its label are linked in the workspace of the smallest constituent SO' in SO that properly contains c. Similar contextual representations are usually preferred in computer science too, and very many have been explored. Statman (1974) uses explicit graphs to represent same-variable same-binding relations among variables in the lambda calculus. But since explicit graphs are not ideal for processing, de Bruijn and many others propose contextual representations in the syntax, aiming to make them as local and easy to identify as possible (de Bruijn, 1972; Bird and Paterson, 1999; Altenkirch and Reuss, 1999; Bezem et al., 2002; Chilpala, 2008; Bernardy and Pouillard, 2013; van Antwerpen et al., 2018; McBride, 2018). As the Chomsky quote suggests, contextual representations are not only computationally more attractive than global identifiers, but they are psycholinguistically more plausible. The process that identifies copies in our structures is simple, fast and efficient, as in de Bruijn's. And unlike de Bruijn indices, our proposal does not require counting. Chomsky (2021), Chomsky et al. (2023, pp.24ff) propose a 'Form Copy' rule to identify copies, but I do not know how to state it precisely, so I do not compare it here. Form Copy is supposed to mark copies formed by movement and also yield a theory of control, but it seems to be absent in Marcolli et al. (2023c,a,b). Cf. Landau (2024) on its connection to control, van Gelderen (2024) on its connection to θ theory.

[8] Even when there is no vocabulary at all, as in the grammar g={}, the domain of mrg is still infinite, because when mrg applies to a sequence of 0 elements it yields the empty set, and then larger sets can be formed containing that set, without bound. In standard set theory, this is called the 'cumulative hierarchy of sets'. Cf. Levy (1979, §§6,7), Kunen (1980, §III.2).

Note that the ranks of derived structures correspond to the stages in the biological evolution of language hypothesized by Suzuki and Zuberbühler (2018) and Scott-Phillips and Heintz (2022), based on suggestions of Rizzi (2009), Schlenker et al. (2016) and others. One can imagine a linguistic system that has only a finite lexicon, with symbols linked to utterances and meanings – rank 0. At rank 1, lexical items are combined, and compositional determination of meanings becomes relevant. Some animals appear to have restricted 1-merge systems. At rank 2, merged elements begin to be combined with other elements, i.e. merge becomes recursive in the sense that it applies to its own output. Then arbitrarily higher ranks become available without any additional mechanism beyond the ability to remember and combine previously merged results.

To associate a head SO and substructures with labels, we pair a sequence of SOs with a sequence of labels, so that the i'th SO in the workspace is associated with the i'th label. Chomsky (2012, 4) also recommends keeping structures separate from labels: "...labeling should not be marked at all in generated structures but rather regarded as a property of [the generative mechanism] G."

Binary match. From each lexical item lex=(word,(label,agr)) we generate the workspace (lex,label), and then further elements are generated by merging and labeling those elements. From those lexical workspaces, we derive further workspaces that have complex SOs formed by merge. A workspace with a complex, merged head can be labeled when it is formed from workspaces that have matching features of differing polarity: the negative first feature of a head matches the positive first feature of another element.

Since one member of every derived workspace is a head that contains all other syntactic objects in that workspace, that head can be defined as the largest syntactic object in the workspace, the unique element that has more nodes than any other. So we can define a function:

$$\text{hd (SOs,labels)} = \max \text{ SOs.}$$

The uniqueness of the largest element is established in (23), below. This definition, though, is unlike every other function in the construction of minimalist grammars in that it requires actually examining a whole syntactic object. Neither mrg nor any other function in our definition of modular minimalist grammars needs to do this. We could define the head SO as the one with more leaves than any other, but that definition requires finding all the leaves – something unlike anything else in syntax.

In any grammar, let's say that f is a *category feature* iff it occurs as the first positive feature of some lexical item. And f is a *licensee* iff it occurs as a non-first positive feature in some lexical item. Then if category features and licensees are disjoint, an alternative definition is:

$$\text{hd } ((\text{SO}_1,\ldots,\text{SO}_n),(\text{label}_1,\ldots,\text{label}_n)) = \text{the SO}_i \text{ such that label}_i \text{ is either (i) negative or (ii) positive with a first feature that is a category.}$$

The uniqueness of the head defined by category feature is established in (24), below.

Another way to identify the head in a derived workspace is modify ck to augment labels with an indication of which is the head, or some way to compute that property.[9]

But the easiest, nearly cost-free way to identify a head is by derivational history: Each lexical item is a head, and in each merged objects is a head its workspace. Since the workspace is a sequence of SOs paired with a sequence of labels, we simply put the head first. So then we have:

$$\text{hd } ((\text{SO}_1,\ldots,\text{SO}_n),(\text{label}_1,\ldots,\text{label}_n)) = \text{SO}_1.$$

The question of how to identify heads will come up again, but for the moment, we use this derivational, history-based encoding. The head will always be the first SO in the workspace.

Given any feature f and a sequence of workspaces, match construct a pair of workspaces (WS,WS') The input to match is a sequence of workspaces $\text{WS}_1,\ldots,\text{WS}_i$ in which exactly 1 workspace has a negative head SO, with label beginning with (negative) f. Without loss of generality, let's assume that the negative workspace is WS_1. Then following Kobele (forthcoming) and others, we impose this requirement:

merge-over-move (MoM): if WS_1 also has a SO' with positive label' beginning with f, then (i) $i = 2$, (ii) the head SO' of other workspace WS_2 is identical to SO, and (iii) WS=((SO,SO'),(label,label')) and WS' is all the the non-matching elements of WS_1.

When WS_1 has a SO' with positive label' beginning with f and MoM applies, this is called *internal merge* (IM). On the other hand, if WS_1 does not have a SO' with positive label' beginning with f, then

Span: the head of each of $\text{WS}_2,\ldots,\text{WS}_i$ has a positive label with first feature f, WS contains all the matching heads and labels of $\text{WS}_1,\ldots,\text{WS}_i$, and WS' has is any elements that do not match.

When this case applies, it is called *external merge* (EM).

[9] Recent labeling theory explores different ideas about how to identify heads, left for future work.

The match rule applies to $i \geq 2$ workspaces when exactly 1 input workspace is negative with first feature f as follows:

$$\frac{\text{WS}_1, \quad \ldots, \text{WS}_i}{\text{WS,WS'}} \text{ (match)}, \qquad \begin{array}{l}\text{where, respecting MoM and Span,} \\ \text{(WS,WS') is a partition of elements of input WSs} \\ \text{according to whether the head has first feature f}\end{array}$$

The MoM condition guarantees that when the unique workspace with a negative element also has a matching positive element, IM is the only possibility. Some linguists have instead assumed a merge-over-move preference (Epstein et al., 2012b; Chomsky, 2000, 106), but there are empirical problems with those proposals (Shima, 2000; Castillo et al., 2009; Abels, 2012, §4.3.1). A large literature touches on these issues, some of which come up again in §1.4.1 below.

Feature checking. After feature matches are found in the match step, we may want to 'forget them', or 'remember that they have been found'. This is done by 'checking' them, detaching them from the label of the result. (See fn. 2.) The first negative feature of the head is 'detached', along with the matched first feature of the complement.

$$\frac{\text{f}.\alpha \multimap \beta_1, \quad \text{f}.\beta_2}{\alpha \multimap \beta_1, \quad \beta_2} \text{ (ck)}.$$

Note that this function is reducing the number of features to be remembered in the label of the current derivation step. So it should not be be regarded as deleting anything, but rather as unlinking it from one of its positions.

A workspace is a pair (SOs,labels) where SOs is a sequence of syntactic objects – SO_1,...,SO_i – with a corresponding sequence of labels – label_1,...,label_i. In derived workspaces, SO_1 is the head and label_1 is its label. By (19) below, any other structures in a derived workspace are substructures of the head, so the workspace can be regarded as a kind of 'locus of attention' over which the derivations are defined, with a memory of just those lexical features that are still unchecked.

An SO is *negative* iff its label is, and *positive* otherwise. And a workspace is *negative* iff it contains a negative label, and *positive* otherwise. So a negative workspace can contain positive SOs, but a positive workspace is one that has only positive SOs.

Workspace simplification: Rule t. The previous step 'checks' and 'detaches' features from a label, so eventually we calculate labels with no features at all. When that happens, the (sub)structures and their empty labels need not be remembered any more. Rule t simply unlinks all SOs and corresponding labels of those structures when the labels are empty:

$$\frac{\text{WS}}{\text{WS - } \{(\text{SO}_i,\text{label}_i) \mid \text{label}_i = \top\}} \text{ (t)}.$$

Structures with no features in their labels are syntactically inert, so there is no need to keep them linked in the workspace. When unlinked from the workspace, they of course remain in their original positions in the SO, so this not a deletion, but an unlinking from the locus of attention.

A shortest move constraint. Initially, we will assume that no workspace can have two positive elements with the same initial feature. Intuitively, the derivation can only remember one structure of each kind, where the relevant notion of 'kind' is given by a first (positive) feature occurrence.

$$\frac{\text{WS}}{\text{WS}} \text{ (smc), if no 2 positive SOs have a first feature in common.}$$

So smc is formulated as the identity function on workspaces in which no two SOs have first features that are positive and identical. Stabler (1997, p.75) notes that this condition could be regarded as a local 'economy condition' that (in our current terms) prevents two positive occurrences of a feature from competing for a checking relation with a higher negative occurrence of that feature.[10]

[10] Compare the 'shortest move' proposals in Chomsky (1995, §4.5.5). In §1.3.6, we consider eliminating this condition.

Composing merge and label into one derivational step d. Now, with 3 additional, simple functions on sequences, we can define the subset of the domain of merge that can be labeled.[11] Define

: ('cons') puts an element onto the front of a sequence:

1:(2,3) = (1,2,3)

tail maps a non-empty sequence to all but its first element

tail 1,2,3 = 2,3

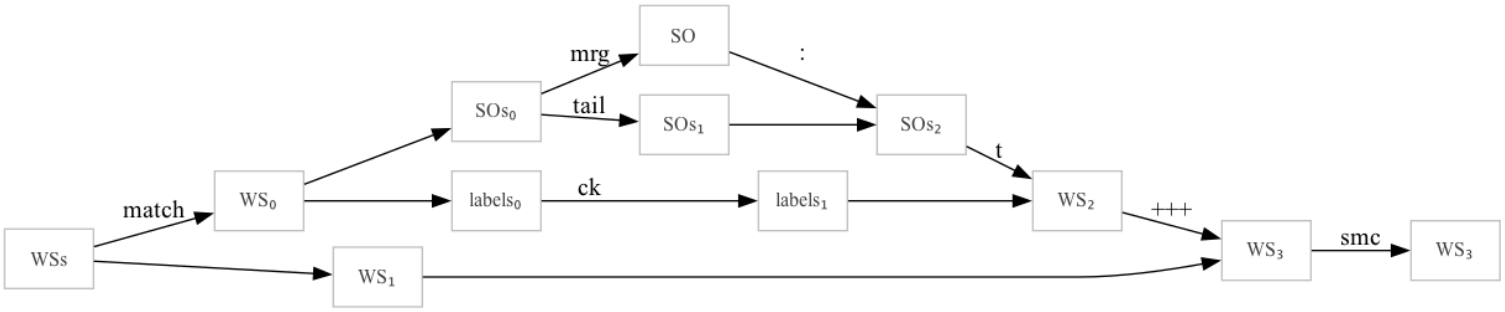
+++ concatenates pairs of sequences

(1,2),(a,b) +++ (3,4),(c,d) = (1,2,3,4),(a,b,c,d).

Our basic derivational step d then maps a sequence of WSs to a WS if it can be labeled, as follows:[12]

(7) d WSs = let ((SOs,labels),others) = match WSs in
smc (t (mrg SOs:tail SOs) (ck labels) +++ others)

This calculation, deriving a new workspace from a sequence of 2 workspaces, can be depicted like this:



Following the paths from left to right, match applies to a sequence of workspaces (initially required to be a sequence of length 2), and forms new workspaces: WS_0 with SOs that have matching first features, and WS_1 contains any (possibly zero) sub-SOs with non-matching features (the 'movers'). Then mrg applies to the matching SOs, and ck to matching labels. The cons function : puts the newly merged element first in a new sequence followed by the non-head element(s) that were merged, and t unlinks inert elements. Finally +++ combines the newly merged structure with the non-matching movers, and smc crashes if two positive elements (i.e. two 'movers') have the same first feature.

Since match and smc will be undefined on some sequences of workspaces, d is partial too, as expected. We cannot label everything.[13] Also note that this definition of d is not recursive, in the sense that d is not used in its own definition, nor is it used in definitions of any of the functions in the definition.

[11] Here and throughout, notice that we write the application of a function f to an argument a in the 'functional style' (f a), rather than writing f(1). When a function takes two arguments, like the addition function +, we write either (+ 1 2) or 1+2. Outermost parentheses are omitted when no confusion will result.

[12] This definition of d can be implemented in Haskell exactly as it is given here. Implementations of all our definitions and examples are available at https://github.com/epstabler/mgt. With some extra work, a Python implementation can be similar, and that is provided as well. These calculators are useful for checking claims about derived structures.

The `let...in...` construction is removable but enhances readability, and in programming languages it can define a 'staged computation' that avoids recalculation of results: cf. https://en.wikipedia.org/wiki/Let_expression, Davies and Pfenning (2009), Vytiniotis et al. (2010). The `...where...` construction is a stylistic variant, sometimes more readable.

The definition indicates that SOs (SOs_0 in the diagram) is used twice, as inputs for both mrg and tail. This allows 'moved' elements to play a role at multiple positions in the structure. But looking more closely at what is really happening here, we see that different features are used in the different positions. This is highlighted in the 'matching graph' that is the middle structure of Figure 1. Computation in which each intermediate result is used exactly once eliminates a kind of redundancy and has appealing properties (Walker, 2002; Bernardy et al., 2018; Gallot et al., 2020; Milewski, 2024).

[13] Even when a minimalist grammar labels every *pronounced string*, it will still be true that not all *sequences of workspaces* can be labeled – like any sequence of length $n \neq 2$ or any sequence (WS_1,WS_2) where $WS_1 = WS_2$. Recall that, in most or possibly all human languages, any sequence of words can be a name (Pullum and Gazdar, 1982; Manaster-Ramer, 1983, 1987, 1988; Stabler, 2019). In particular, in most and possibly all human languages, any sequence of words can be used as a quote name for that sequence of words itself. But names and parts of names do not have the same syntactic requirements as other parts of speech, as will be reflected in the features that drive their composition (Potts, 2007; Pagin and Westerståhl, 2010).

The set of *derived workspaces* of grammar g is the closure of $\{((w,(l,agr)),l)\,|\,i \in g\}$ with respect to d. If WS = (SO,label), with exactly one SO and exactly one positive feature f in its label, we say SO is a *complete structure of category* f, and we say SO is an f phrase, an fP. Conventionally,, one category in each grammar is taken as the basic *start category* – the category of a complete sentence, a clause. In our examples here, we assume the start category is C.

Example, continued From grammar (1), we can merge a verb with a direct object, not as an instance of one case among many of *merge* as in Stabler (1997), but in 8 simpler steps:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{((\text{likes},\text{D}\multimap\text{V}),\text{D}\multimap\text{V})\quad((\text{Jo},\text{D}),\text{D})}{(\,(\text{likes},\text{D}\multimap\text{V}),\ (\text{Jo},\text{D},\text{D})),(\text{D}\multimap\text{V},\ \text{D})\,),\ \ ()}\,\text{match}}{((\text{likes},\text{D}\multimap\text{V}),(\text{Jo},\text{D}))}\,\text{mrg}\quad\dfrac{((\text{likes},\text{D}\multimap\text{V}),(\text{Jo},\text{D}))}{(\text{Jo},\text{D})}\,\text{tail}}{\{(\text{likes},\text{D}\multimap\text{V}),(\text{Jo},\text{D})\},(\text{Jo},\text{D})}:\quad\dfrac{\text{D}.\text{D}\multimap\text{V},\text{D}}{\text{D}\multimap\text{V},\top}\,\text{ck}}{(\{(\text{likes},\text{D}\multimap\text{V}),(\text{Jo},\text{D})\},\ \text{D}\multimap\text{V})}\,\text{t}\quad()}{(\{(\text{likes},\text{D}\multimap\text{V}),(\text{Jo},\text{D})\},\ \text{D}\multimap\text{V})}\,{+}{+}{+}}{(\{(\text{likes},\text{D}\multimap\text{V}),(\text{Jo},\text{D})\},\ \text{D}\multimap\text{V})}\,\text{smc}$$

The same 8 steps combine *likes* and *who*, but because *who* has two positive features, the effect of step t is different, and as a result, the resulting workspace has 2 SOs instead of just 1.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{((\text{likes},\text{D}\multimap\text{V}),\text{D}\multimap\text{V})\quad((\text{who},\text{D.Wh}),\text{D.Wh})}{(\,(\text{likes},\text{D}\multimap\text{V}),\ (\text{who},\text{D.Wh})),(\text{D}\multimap\text{V},\ \text{D.Wh})\,),\ \ ()}\,\text{match}}{((\text{likes},\text{D}\multimap\text{V}),(\text{who},\text{D.Wh}))}\,\text{mrg}\quad\dfrac{((\text{likes},\text{D}\multimap\text{V}),(\text{who},\text{D.Wh}))}{(\text{who},\text{D.Wh})}\,\text{tail}}{\{(\text{likes},\text{D}\multimap\text{V}),(\text{who},\text{D.Wh})\},(\text{who},\text{D.Wh})}:\quad\dfrac{\text{D}.\text{D}\multimap\text{V},\text{D.Wh}}{\text{D}\multimap\text{V},\text{Wh}}\,\text{ck}}{(\{(\text{likes},\text{D}\multimap\text{V}),(\text{who},\text{D.Wh})\},(\text{who},\text{D.Wh})),\ (\text{D}\multimap\text{V},\text{Wh})}\,\text{t}\quad()}{(\{(\text{likes},\text{D}\multimap\text{V}),(\text{who},\text{D.Wh})\},(\text{who},\text{D.Wh})),\ (\text{D}\multimap\text{V},\text{Wh})}\,{+}{+}{+}}{(\{(\text{likes},\text{D}\multimap\text{V}),(\text{who},\text{D.Wh})\},(\text{who},\text{D.Wh})),\ (\text{D}\multimap\text{V},\text{Wh})}\,\text{smc}$$

This would be accomplished with a different rule in many early minimalist grammars, but here it is exactly the same pattern of 8 steps. Since these steps are exactly d, and d is the only rule, we can simply write:

$$\dfrac{((\text{likes},\text{D}\multimap\text{V}),\text{D}\multimap\text{V})\quad((\text{who},\text{D.Wh}),\text{D.Wh})}{(\{(\text{likes},\text{D}\multimap\text{V}),(\text{who},\text{D.Wh})\},(\text{who},\text{D.Wh})),\ (\text{D}\multimap\text{V},\text{Wh})}\ .$$

Even in little examples like this, it is a nuisance reading those sets! The standard set notation is redundant because each lexical item gets listed over and over again, in every set that contains (a set that contains...) it. We lose nothing at all by removing the labels from the sets, the internal nodes, with the understanding that the elements of each set are its children, as in the graph on the left in Figure 1. That graph represents a complete syntactic object for *Jo knows which food the cat likes*, a CP, where every derived set is represented by a dot with downward arcs to its members. The middle graph of Figure 1 is a dependency graph for the same derivation, showing the feature checking relations. And on the right, the same structure is shown again with a conventional but less explicit X-bar-like notation.

Extending d through the domain of merge. Given anything in the domain of merge – any lexical item or finite set built from lexical items – it is easy to check whether it is derived from the grammar. Taking the SO on the left in Figure 1, for example, since it is in fact derivable from the grammar, we can compute the workspace that has that derived structure as its head by recursively applying d, first to the leaves, and then to the results of that step, and so on, for a bottom-up transduction from the syntactic object to its workspace.

This computation is almost identical to the deterministic multi tree transductions which have implemented previous versions of minimalist grammar (Kobele et al., 2007; Kobele, 2011; Graf, 2011). But here, we operate not on trees but on (multi)sets. Rather than forcing linguistic structures into the standard

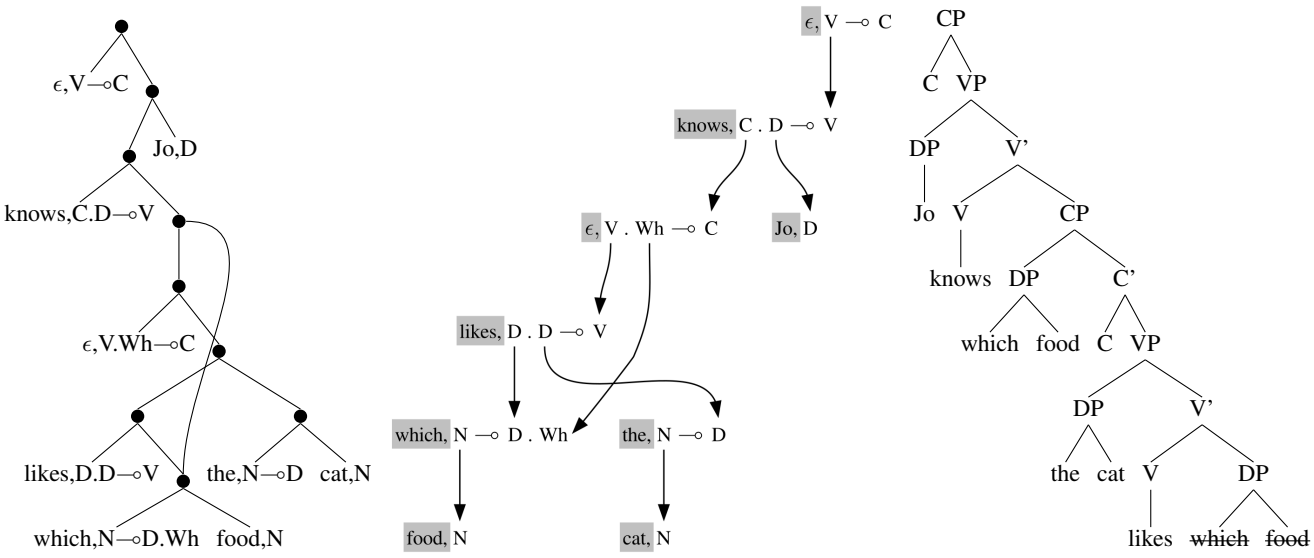


Figure 1: 3 representations for *Jo knows which food the cat likes*. Left, the diagram of a set. The labeling ℓ assigns C to this set by computing the matching graph, middle. Right, an X-bar tree. Compare Figure 7 on page 20.

notation of tree transductions, our transductions can be defined directly on syntactic objects, i.e. on lexical items and finite sets. Formally, define transduction ℓ, for 'label' as in (8):

$$\ell\,\mathrm{SO} = \begin{cases} \{(\mathrm{SO},\ \mathrm{label})\} & \text{if SO is a lexical item (w,label)} \\ \mathrm{d}\,(\ell\,\mathrm{SO}_1)\ldots(\ell\,\mathrm{SO}_i) & \text{if SO=}\{\mathrm{SO}_1,\mathrm{SO}_2,\ldots,\mathrm{SO}_i\}. \end{cases} \tag{8}$$

This definition is recursive. Applied to a non-lexical item, a set, ℓ applies to the elements of that set. In this way, the recursive case of ℓ 'calls itself' immediately, going right down to the leaves – the base case. Then, beginning at the leaves, d builds each workspace from the lower ones, bottom-up.

1.2.2 Generalized head movement

In English and other languages, heads often seem to be displaced from the rest of their projections. There are many views about how this happens. A number of different influences converge on lexical heads, and, as Chomsky suggests, quoting Jespersen (1924), there seems to be considerable variation across languages in how those various pressures are negotiated.[14] The idiosyncratic syntactic, morphological and phonological properties of heads must be learned by each speaker and associated with lexical items. But Harley (2013) observes that "theories of the morphology-syntax interface are even more contentious and disparate than theories of syntax proper." And Kalin and Weisser (2021) note that "The Minimalist Program has a fraught relationship with morphology, especially as represented in Chomsky's own works." Here we show how one recent and possibly representative view can be integrated into the modular MG framework.

The 'm-merger' introduced in §1.3.7 moves heads for morphological reasons, and the prosodic restructuring in §1.3.8 moves elements for phonological reasons, while the head movement in this section is more syntactic, applying only to (possibly non-adjacent) constituents related by selection. Thus we follow Arregi and Pietraszko (2021, p.261) and Branigan (2023, 17-18) in assuming head movement applies before linearization, because the conditions on head movement (esp. when slightly elaborated for do-support below), are syntactic ones. Like Arregi and Pietraszko (2021, 2024) and many others, we assume that head movement can have the effect of both raising and lowering heads.[15] The structures built by this mechanism capture the observation of Baker (1985) that the syntactic structures like (C

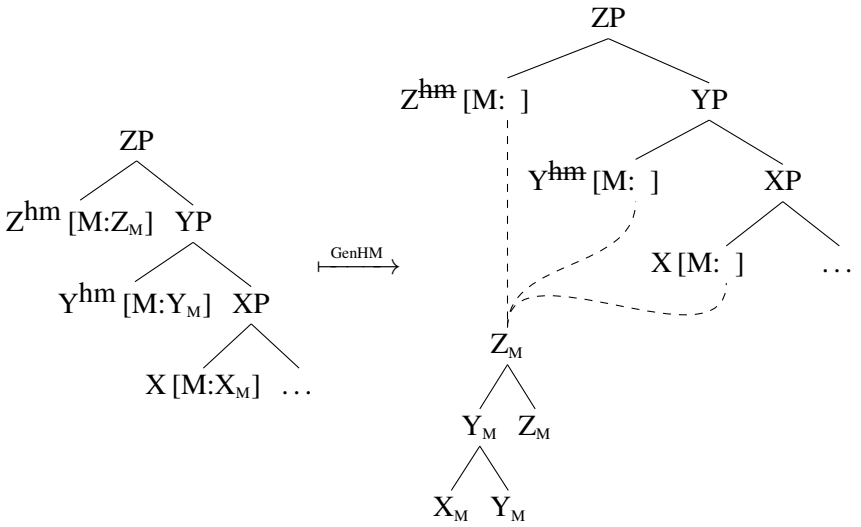
[14] Chomsky (1995, p.3); Chomsky (1975, p.5). What Jespersen actually says is discussed in §3.2 below.

[15] Arregi and Pietraszko (2021) note that their proposal is quite similar to mirror theory and related proposals (Brody, 2000; Adger et al., 2009; Svenonius, 2016), which also get the effect of raising and lowering with one mechanism.

(T (v (V. . .)))) are often mirrored, reversed in morphology by affix sequences like V-v-T-C. But like Branigan (2023) and Collins (2002b), we extend this mechanism to 'multiple head movement', deriving certain exceptional affix orders. Later, §1.3.5 considers realizational alternatives that provide different perspectives on all these phenomena.

Arregi and Pietraszko (2021) hypothesize that generalized head movement takes both the morphological content of Y_M of a lexical item marked for head movement, Y^{hm}, and the morphological content X_M of the head of its complement, and replaces both with a shared structure that adjoins X_M to Y_M. With multiple head-marked elements, this happens recursively, as illustrated in (their (4)):

(9)



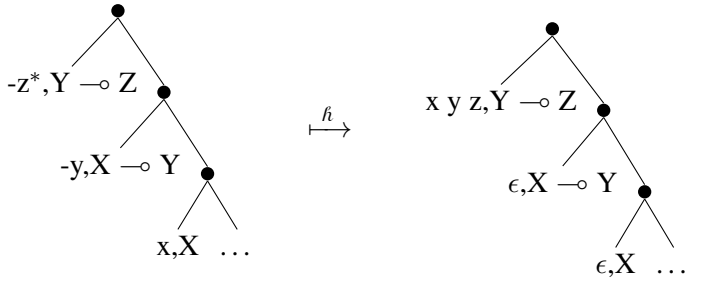
Note that the adjunction of heads respects the mirror principle. And since the operation is triggered by heads in a selection relation, it respects the 'head movement constraint' (Travis, 1984; Baker, 1985). In a second step, assuming that some heads may also be marked 'strong', the adjunction structure is placed by this delinking rule:

(10) Unlink the complex from all positions except the position of the highest strong head, else if none are strong, unlink from all positions except the highest.

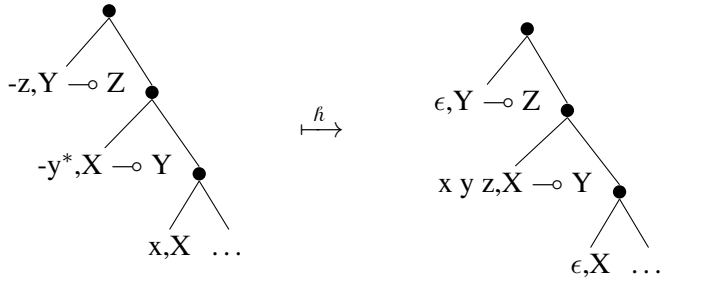
We define a head movement transduction $\hbar$ but since we have no processes sensitive to head adjunction structure of the sort shown in (9), our head complexes are simply concatenated. Head complexes are formed from *complement sequences*, that is, sequences of heads $(h_1,\ldots,h_n)$ in which each nonfinal h_i selects h_{i+1} as its first argument, hence immediately c-commanding it. Complement sequences related by head movement are called *head chains*. We require that all nonfinal elements of head chains are marked not hm as in (9), but -. Intuitively, these heads are dependent in some way.[16] In (11), the final heads in the complex-forming sequences are not marked -. But, in the special cases triggering do-support, we allow final heads marked -, discussed below. Morphologically strong heads are marked *, as in Arregi and Pietraszko (2021). And $\hbar$ also does the delinking, placing the head complex into its final position. So here is our analog of (9) and (10):

[16] See e.g. the proposal in Branigan (2023, §2.2) that the heads marked - require association with a morphological stem.

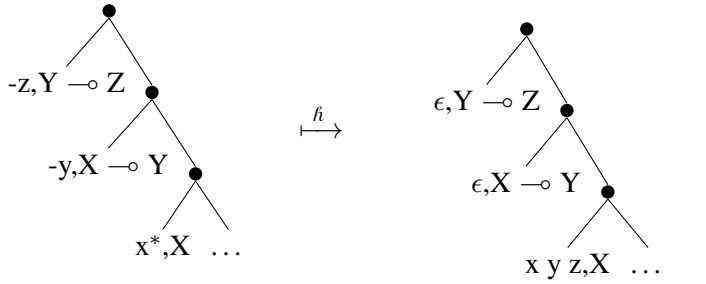
(11) a. (z strong) head chain: -z* ← -y ← x



b. (y strong) head chain: -z → -y* ← x



c. (x strong) head chain: -z → -y → x*



(We add one more kind of case, case (11d) on p. 17, below.)

As rule (10) prescribes, if none of x, y and z in (11) are strong, or if all of them are strong, the result is the same as in (11a).

All of the input trees in (11) are labeled Z by ℓ. We define the head movement transduction $\hbar$ so that it takes these trees and builds head complexes as needed in a downward pass through the structure (so, in effect, the dashed links in the Arregi and Pietraszko diagram (9) indicate the head complex being held by the transduction).[17] Then the head complexes are placed according to the delinking rule in an upward pass through the structure. Downward, the heads of in the complement sequences are collected, and upward, the concatenated complexes are placed. Other than the prescribed changes in the morphophonology of the heads, there is no other change. The syntactic structure remains the same. No syntactic features are touched.

[17] Compare 'multi' tree transductions that build tuples of intermediate trees (Engelfriet et al., 2009).

English tense and auxiliaries. We can formulate something like the familiar treatment of English verbal complexes in Figure 2. Head movements are indicated with dashed lines for readability, but those dashed lines are not part of the structure.[18] Input and output structures differ only in morphophonological contents at the leaves.

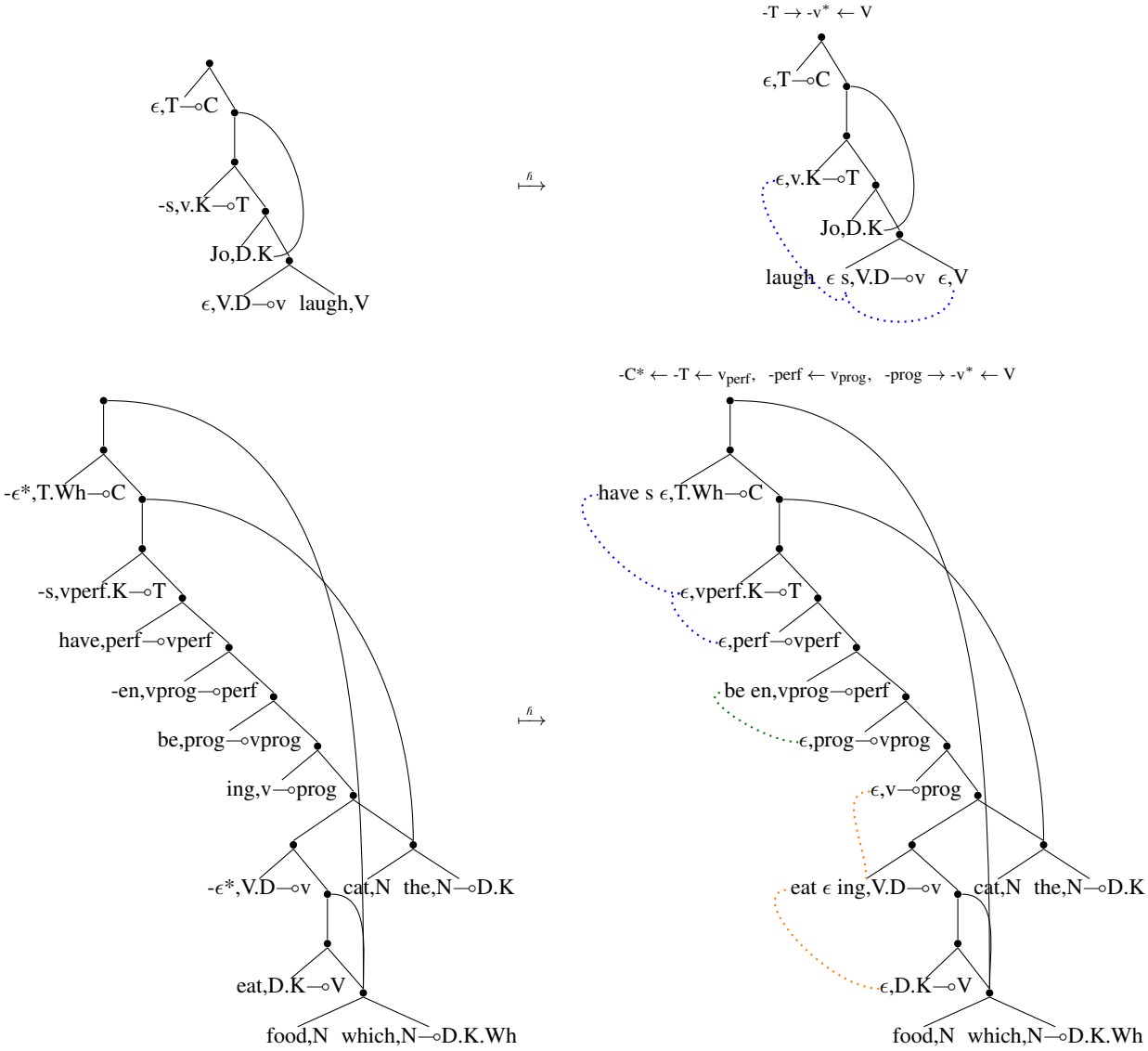


Figure 2: Head movement applies to structures for *Jo laughs* and *Which food has the cat been eating*. In the input structures, each D has moved to a position that checks K ('EPP'), and in the latter, the wh-phrase has moved a second time to spec,C to check Wh. So ℓ labels the input SOs as C. Then h applies to build and place the chains. The latter structure has 3 distinct head chains. See the linearizations of these structures in Figure 5 on p.18. (Also, notice that the same result would be obtained in the second structure here even if the C of English subject-auxiliary inversion were not strong, but we have evidence that it is strong from examples like the second structure in Figure 3, where the subject intervenes between C* and v*, attracting DO-s and not the verb after the chain is split.)

[18] Ramchand and Svenonius (2014, p.155) point out:

> Linguists differ with respect to whether they simply represent Perf, Prog and Voice as functional heads (Bjorkman, 2011; Sailor, 2012) and handle the inflectional facts via 'affix lowering' or AGREE, or whether they in addition assume separate functional heads hosting -en and -ing (Bošković, 2014; Harwood, 2014). The minimalist assumption seems to be that some kind of selection is at work, and does not represent a universal functional sequence, and these projections are left out even for English when the literal perfect or progressive forms are not expressed in the sentence.

Figure 2 roughly follows Harwood (2014, (17)) and Arregi and Pietraszko (2021). Compare the different hypothesis in §1.3.5.

Do-support in English. What explains the need for *do* in (12)?

(12) a. Jo does not laugh / *Jo not laughs

b. who does he see / *who sees he?

c. Jo DOES laugh / ?Jo does laugh

Here we formulate an intervention-based proposal based largely on Arregi and Pietraszko (2019, 2021, §4.1). They argue that do-support of the sort we see in (12) is triggered by intervention or ellipsis, citing Pollock (1989, pp.409–422), Bobaljik (1995, pp.72–73) as precedents. We leave ellipsis for future work, and focus on the intervention-based proposal. Given (occurrences) X, Y, Z in an SO, define as usual:

(13) X *c-commands* Y iff X has a sister that dominates Y.

(14) Y *intervenes* between X and Z iff X c-commands Y and Z, and Y c-commands Z.

Then using the categories assumed in Figure 2, together with the Σ category sometimes posited for negation (Laka, 1990), the claim about the degraded cases in (12) is:

(15) a. In (12a), negation in spec,Σ intervenes between T and v in T-Σ-v*-V.

b. In (12c), non-moving *he* in spec,T, intervenes between C and v in C*-T-v*-V.

c. In (12c), verum focus in spec,Σ intervenes between T and v in T-Σ-v*-V.

The reference to 'non-moving' in (15b) is required to avoid breaking chains with non-final positions of moving elements – as in the head chain -prog$\rightarrow$-v*$\leftarrow$V in the structure on the right in Figure 2.[19]

To require do-support in (12), transduction $\hbar$ is slightly elaborated as follows:

(16) (**Head complexes**) Complexes are built from complement sequences in which

i. all non-final heads are marked -,

ii. no non-moving specifier intervenes between the first head and attachment position, and

iii. no two heads in the sequence have the same category.[20]

(**English do-support**) If a chain ending in V has an intervening non-moving specifier, split at v* and provide DO as the stem for higher affixes.

(**Attach**) The attachment position of head chains, including each piece of a split chain, is defined by (10).

This leaves the analysis of structures in the previous Figure 2 unchanged, but allows those with do-support in Figure 3. After splitting the chain, Arregi and Pietraszko (2021, §4.1) say that the higher chain has an 'orphan' V and the lower chain has an 'orphan' T, where the higher orphan V becomes DO and the lower orphan T is deleted. But their analyses suggest that, across languages, intervention-sensitivity is not universal and there is substantial variation in the exponence of orphaned elements.

No mention is made of 'extended projections' in (16). So on this kind of approach, when N is part of a verbal complement sequence, there is no problem allowing noun-incorporation as an instance of head movement (Baker, 1988).

[19] Arregi and Pietraszko (2021, p.261) propose a feature P to mark non-moving positions, suggesting that moving XP[+P] to a higher position leaves XP[-P] in the lower position. But we do not need the special feature P, since our labels already indicate unambiguously and efficiently whether a constituent is moving. And our labels are created by ℓ, not by merge, and so the labels are part of the workspaces, not part of any SO. Consequently, we have no violation of the hypothesized 'no tampering' condition on merge – cf. Chomsky (2007, p.13), Collins and Stabler (2016, p.58).

[20] Compare the non-recursiveness condition in Kobele (2002, §4.2).

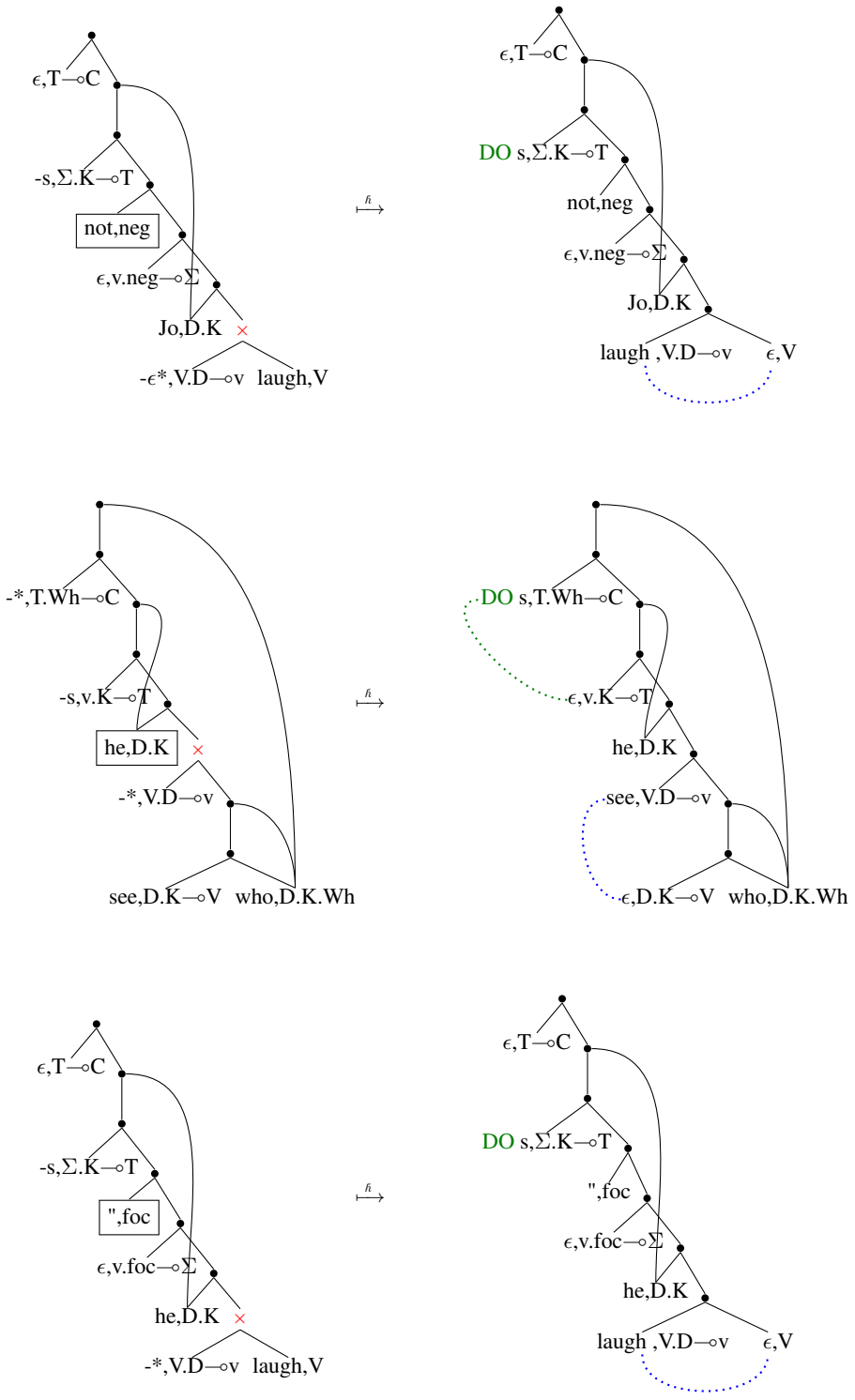
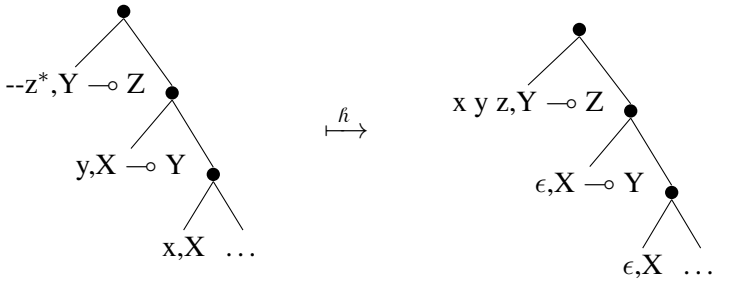


Figure 3: Structures for (12a,b,c), using the IPA 'extra stress' symbol " for verum focus in the last one. The interveners in the input trees are the (highest occurrences of the) boxed elements. The resulting break points, at v*, are marked ×. These structures are linearized in Figure 6 on page 19.

Multiple head movement in Javanese. A number of authors have proposed that head movement sometimes creates a non-standard order of heads as the result of 'tucking in' additional heads when one head has multiple dependencies. We make a simple extension in the mechanism above to allow heads that probe for multiple lower heads, allowing n heads not marked - to occur below a head with n probes. So we add this kind of case to those listed in (11):

(11) continued d. head chain: --z* ← y ← x



Cole et al. (2008) propose that some complexes of Peranakan Javanese might be derived like this, as in these examples, their (105a,107a,115,114):[21]

(17) a. Dheen gelem isa ngomong Inggris
he want can speak English

b. Gelem isa dheen ngomong Inggris?
want can he speak English?

(18) a. Tono pernah gelem isa ngomong Inggris
Tono PERF want can speak English

b. Pernah gelem isa Tono ngomong Inggris?
PERF want can Tono speak English?

Branigan (2023) proposes that multiple head movement happens when a single head has multiple 'probes'. Our analog of a probe for head movement is the morphological feature -, so let's extend the head movement operation $\hbar$ so that a chain-initial --x attracts both the next 2 heads, the head ---x attracts 3 heads, and so on.[22] This approach derives the structure in Figure 4 (where we use the English glosses). In that structure, as in the similar structures of Cole et al. (2008, (121)) and Branigan (2023, p.56:(52)), the subject clearly intervenes. However, our rule (16) for DO-support specifies that chains with interveners only break if they end at V. (The formulation in Arregi and Pietraszko (2021, (39)) is similarly specific.) So if we have ---ϵ* in C, looking for 3 heads, and if (the Javanese elements for) *want*, *can* and *speak* are not marked as dependent, then the chain ends not in V but at the v *can*. So perhaps intervention does matter but just not in this case, or perhaps Javanese head movement is not sensitive to intervention.

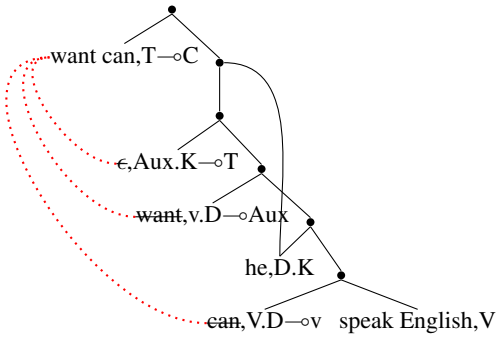


Figure 4: Multiple head movement in the Javanese (17b): *Gelem isa dheen ngomong Inggris?*

[21] Compare analyses of Serbo-Croatian, Standard Dutch, and Polish (Bošković, 1999), ǂHoan (Collins, 2002b), Javanese (Cole et al., 2008; Vander Klok, 2015), and Russian and Innu-aimûn (Branigan, 2023).

[22] With multiplicity allowed only on initial elements, the number of heads i to be incorporated at any point is bounded by the maximum number of triggering features on any single element in the lexicon. So this proposal requires no adjustment in proposed noncounting and tier strictly local hypotheses about human language. Cf. Heinz (2018); Graf (2022a,b, 2023). However, this proposal does conflict with the common idea that human languages cannot count past two (Newmeyer, 2005, (2)). It even conflicts with the claim of Berwick and Weinberg (1984, pp.158-9) that no rule in the grammar could possibly refer to "the seventh S domain above the current one." Quite possibly, no lexical item has seven morphological probes; even two is unusual, but that is plausibly the consequence of more general constraints on memory, etc. Reghizzi and Braitenberg (2003) hypothesize, as we do, that the bound in human language, and also in music and other cognitive domains, is not strictly two, but finite and small. It is plausible that in these domains we find only counting modulo n for small n.

1.2.3 Linearization

We will tentatively assume that linearization is a transduction that maps multisets to sequences, mapping SO = {SO_1,..., SO_i} to a sequence SO' = (SO_1,..., SO_i). As usual, sometimes outermost parentheses will be left out when no confusion will result.

One very simple ordering function inspired by Kayne (2020, 1994), Chomsky (1995), Cinque (2023) and others unlinks moving elements from non-final positions, puts first-selected elements on the right of the head, and later-selected elements on the left. Let's call that function o_{svo}. It maps (multi)sets to sequences. Letting each sequence be a node whose daughters are its elements, in order, the output of o_{svo} is always an ordered tree. Each node has at most one parent. In the minimalist jargon: although constituents can be 'repeated', there are no 'copies'. See examples in Figures 5 and 6.

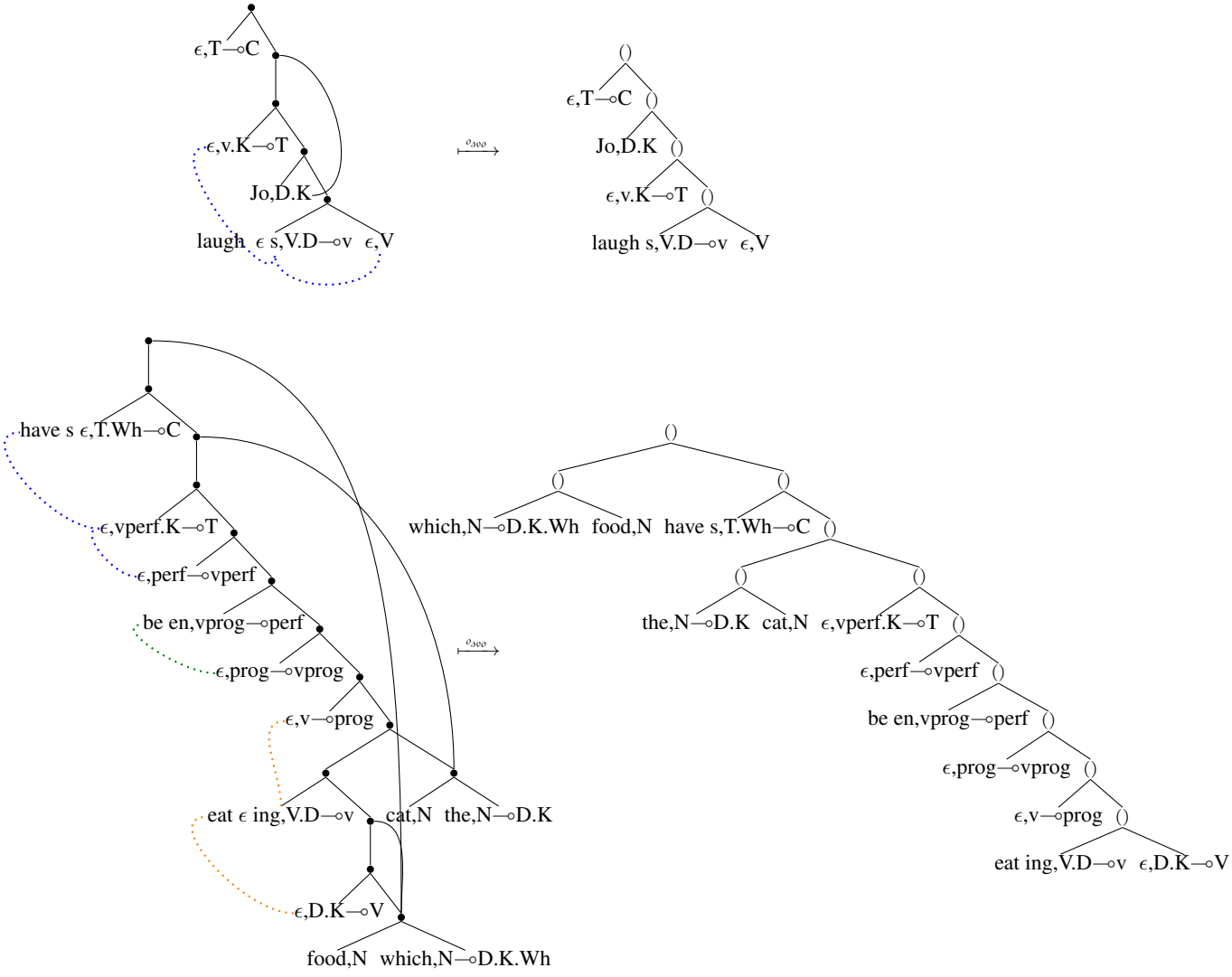


Figure 5: **Linearizing the structures of Figure 2.** Each node labeled () is a sequence, whose elements are its children, in order. In the former structure, in addition to imposing a linear order, the 2 copies of the subject are reduced to 1 by unlinking all but the highest position. In the latter structure, 3 copies of *which food* are reduced to 1, and 2 copies of *the cat* are reduced to 1. The output of linearization is an ordered tree, but note that the number of lexical items is unchanged by this step, and the syntactic features are untouched.

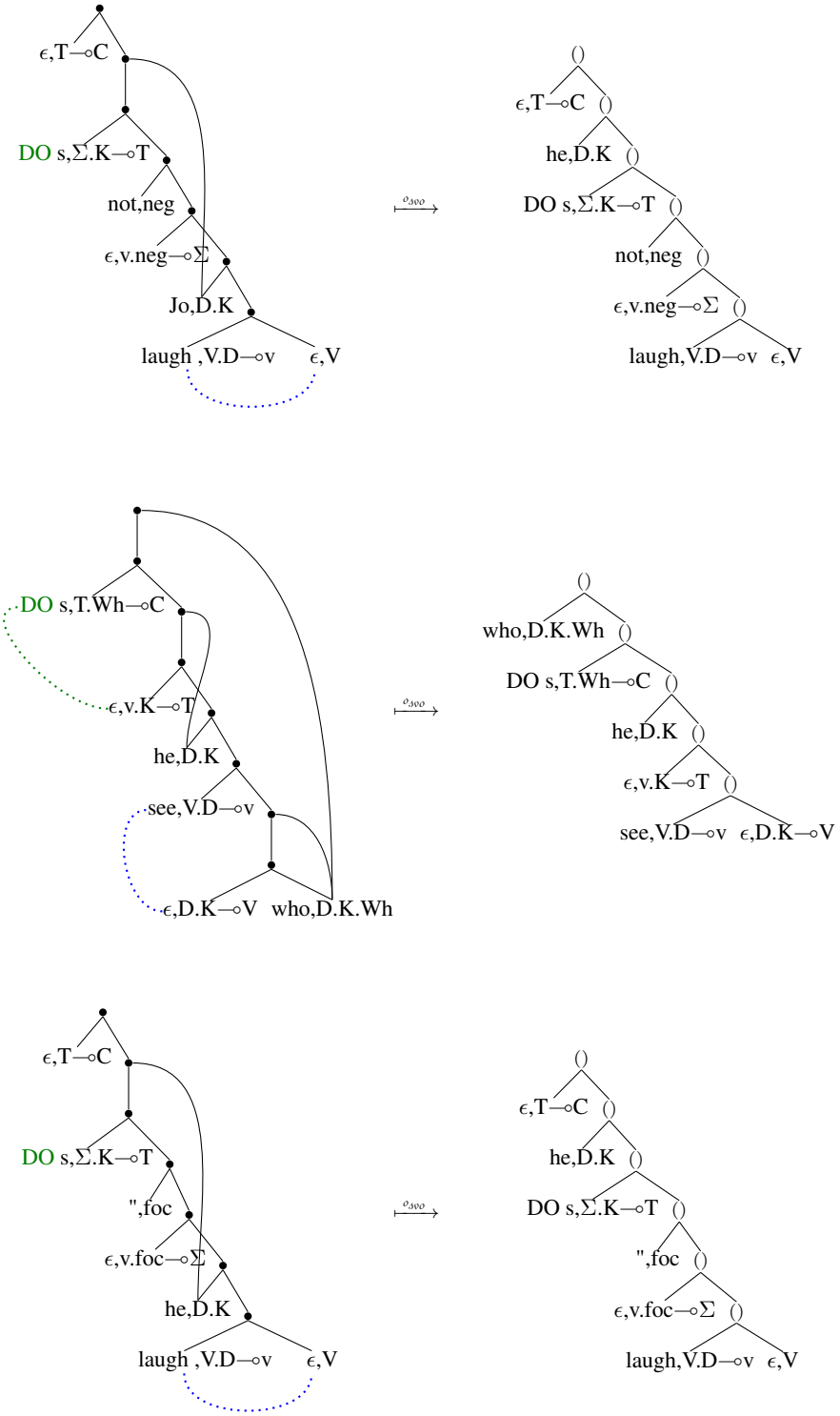


Figure 6: **Linearizing the structures of Figure 3.** As in Figure 5: the output of linearization is an ordered tree; the number of lexical items is unchanged; syntactic features are untouched.

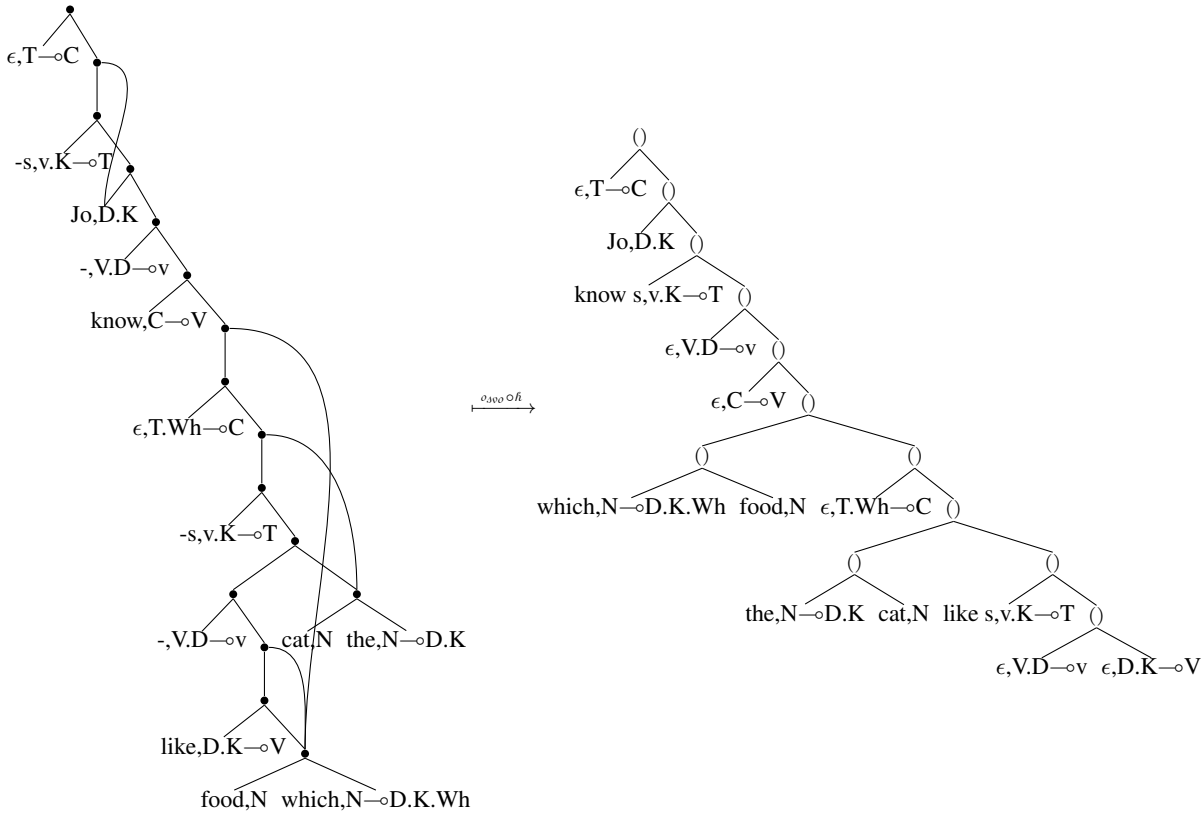


Figure 7: **Elaborating the structure of Figure 1 with head movement and linearization.** Labeling ℓ assigns C to the set on the left. Head movement adjusts morphophonology. Then linearization orders and simplifies. I use $\circ$ for function composition, as usual. Here the composition of linearization and head movement, $o_{svo} \circ \hbar$, is naturally read as "o_{svo} after $\hbar$".

1.2.4 Properties of binary minimalist grammars

We can easily establish some basic results about the minimalist grammars defined with d:

(19) **Workspace connectedness**. The elements of every derived workspace (SOs,labels) are connected by the merge relation. In particular, letting SOs=SO_1,...,SO_i ($i \geq 1$), SO_1 is the head and, if $i > 1$, every other element – SO_2,...,SO_i – is a substructure of SO_1.

Proof: This can be established by an induction on rank. (Rank 0) For each lexical item i, the workspace $(i, label)$ is trivially connected. As the induction hypothesis, assume that the result holds for derived workspaces up to rank n. (Rank n+1) As the inductive hypothesis, assume workspaces are connected up to rank n. If an element of rank n+1 is formed, connectedness holds for the result since the new, merged element connects the largest structures in each of the input workspaces.

(20) **Effective enumerability**. Given any grammar g, the complete structures of g are effectively enumerable.[23]

Proof: Even if the set of derived workspaces is infinite, as in the case of the example grammar just considered, we can provide an algorithm that lists every derived workspace at some finite point in its computation.

A naive way to enumerate derived structures is by rank. Rank 0 is the set $\{\{i\} | \ i \in g\}$. That set is finite because g is, and so of course it can be listed. Then at rank 1, we consider all possible inputs for Match from the finite set that is the union of $\{\{i\} | \ i \in g\}$ with the results from rank 0. Those results are also finite. And so on for ranks 2, 3,....

Since this naive enumeration recomputes the same results many times, it is common to use the 'semi-naive' closure algorithm familiar from chart parsing (Shieber et al., 1995), which avoids repetitions simply by adding elements to each new rank only if they do not occur in an earlier one.

In particular linguistic tasks, like parsing a sequence of words or generating a sequence of words, it is common to further restrict useless matches by considering only arguments for match that could possibly yield a useful new result. For each such method, a 'soundness' result guarantees that only actual derived structures will be computed, and a 'completeness' result guarantees that all the derived structures for the task input will be found.

(21) **Finite categorization**. For many grammars, like the example grammar in §1.2.1, the set of derived structures in the closure of $\{\{\text{i}\} | \ \text{i} \in \text{g}\}$ with respect to d is infinite. But if, for each derived structure, we take the multiset of its labels, that set is finite. In a sense, this means that there are only finitely many 'categories' of derived syntactic objects, even when there are infinitely many derived syntactic objects.

Proof: The argument offered by Michaelis (1998) for a slightly different version of MGs applies here too. The result follows from the fact that (i) the grammar is finite and so has finitely many labels, (ii) the labels in every derived structure are obtained from lexical labels by detachment, and (iii) by the smc, the number of derived structures in each derived structure cannot be more than 1 + the number of different features that occur positively in the lexicon.

(22) **Binarity**. Every derived structure is either a lexical item or a set containing exactly 2 elements. (Later, we generalize the grammar to derive non-binary multisets in which a single element can occur multiple times.) Head movement changes only morphophonological content, and so the images of derived structures under $\hbar$ are also binary. Linearization does not delete anything but only unlinks copies, and when this is done, any resulting unary branches are removed, so the result of applying linearization are also binary.

Proof: Immediate from the definition d, head movement, and linearization.

(23) **Workspace headedness 1**. For any grammar g, every derived workspace has a unique maximal element, which we will call its head.

[23] On effective enumerability, see e.g. Boolos et al. (2007, §1).

Proof: This can be established by an induction on rank. (Rank 0) For each lexical item i, the workspace $(i, label)$ trivially has i as its unique head. As the induction hypothesis, assume that the result holds for derived workspaces up to rank n. (Rank n+1) Since mrg applies to maximal elements, forming a strictly larger set, the result holds for n+1.

(24) **Workspace headedness 2**. Consider any derived workspace WS of a grammar g in which no initial positive feature occurs as a non-initial positive feature. Then if WS has a head with a negative label, it has exactly one and it contains no positive SO with a label whose first feature is initial. And if WS has no negative SO, then it contains exactly one positive SO whose first feature is initial.

Proof: This can be established by an induction on rank. (Rank 0) For each lexical item i, the workspace $\{i\}$ trivially satisfies the condition. As the induction hypothesis, assume that the result holds for derived workspaces up to rank n. (Rank n+1) Every step combines a negative and positive SO. If the negative SO has more than one negative feature, then after ck it will be the unique negative SO in the result. If it has only one negative feature, then after ck it will be the unique SO in the result with an first feature that is initial.

(25) **Workspace headedness 3**. The first element of every derived workspace is its head.

Proof: Immediate from the definition of d.

(26) **Derivation determinacy, efficiency**. For any grammar g and any set SO in the domain of mrg, if SO has a derivation, it is unique and can be efficiently calculated from SO.

Proof: The uniqueness of the derivation is immediate since d is a (partial) function. Efficiency with respect to the size of the input is clear, since we need 1 step of d per internal node, each step is polynomial, and for any g, there is a fixed finite bound on the number of steps needed to compute d.

(27) **Relations among workspace elements**. There are grammars with derivations in which an SO is unlinked from the workspace by t even though SO contains a substructure SO' that remains in the workspace.

Proof: The derivation of Figure 1 establishes (27). When the empty C combines with *which food the cat likes*, rule t unlinks the VP from the workspace even though that workspace still contains the Wh element corresponding to *which food*.

(28) **Multiple SO occurrences in workspaces**. There are grammars with derivations in which, at certain points, there is more than one occurrence of the same substructure in the workspace.

Proof: Figure 8 shows a complete derivation of a CP in which two occurrences of *the man* appear in the workspaces highlighted in green. At those points, the different occurrences have different labels, as required by smc, so they cannot be confused. It is easy to tell which substructures of the head those elements are.

(29) **Remnant movement**. There are grammars with derivations in which an SO moves after a SO' inside SO has moved.

Proof: The derivation of Figure 9 establishes (29), providing a simplified movement structure for the structure:[24]

I wonder [how likely to t_1 win]$_2$ John$_1$ is t_2.

John is extracted from the adjectival phrase before it moves to C. A simple check, applying d recursively from the leaves up to the root, shows that the SO in Figure 9 is, in fact, a complete structure with the label C.

[24]This kind of remnant movement example is very widely discussed. Cf. Müller (1996, (53b)), Saito (2015, (17b)), Collins and Sabel (2015, (71a)), Collins and Stabler (2016, (20)), Epstein et al. (2018, (4)), *inter alia*. It comes up again in §1.4.1, below.

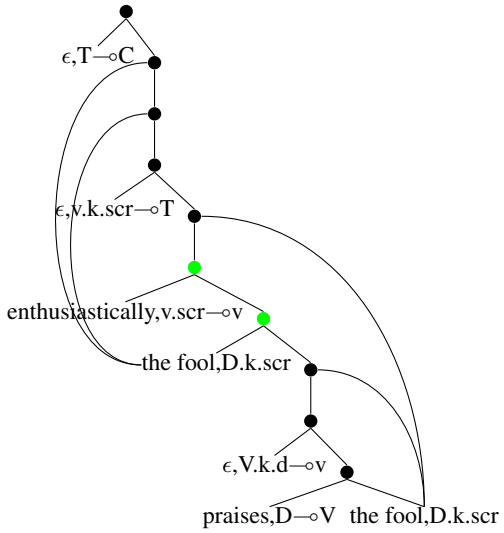


Figure 8: Assuming moved elements pronounced on the left, this syntactic object for *the fool the fool enthusiastically praises* establishes (28). It is crafted to establish that point simply and not intended as a proposal about anything in an actual human language. But compare, e.g., the scrambled structures in Frey and Gärtner (2002).

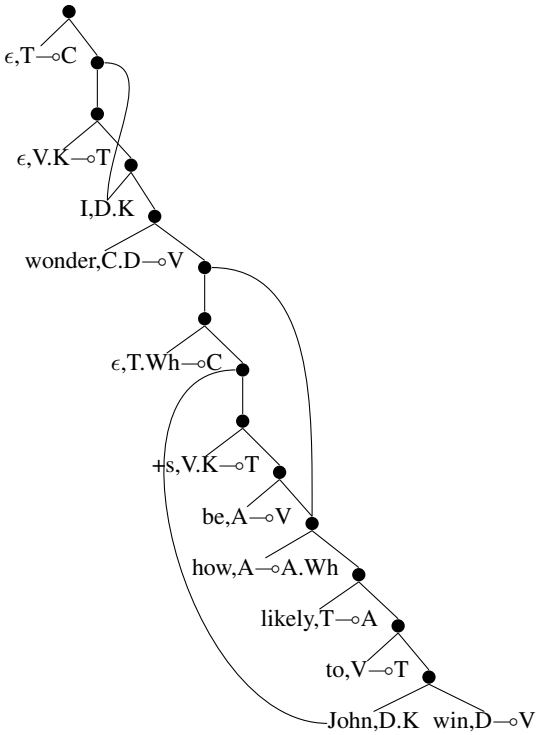


Figure 9: A simple structure for *I wonder how likely to win John is*, establishing (29).

(30) **The labeling function ℓ is an automorphism on the domain of mrg.**[25]

(i) Whenever defined, ℓ maps an so to a workspace whose head is that same so. That is, the composition h$\circ\ell$ is a (partial) identity function, an automorphism on the domain of mrg. That is,

$$\text{so} = \text{h}(\ell\ \text{so}),$$

for all values of so for which ℓ is defined.

(ii) ℓ is a (partial) isomorphism from the domain of mrg to workspaces, preserving mrg and d in the sense that

$$\ell(\text{mrg}\ \vec{\text{so}}) = \text{d}(\ell\ \vec{\text{so}}),$$

for all values of $\vec{\text{so}}$ for which these functions are defined. For values at which the functions are defined, the equation can be depicted like this:

$$\begin{array}{ccc} \vec{\text{so}} & \xrightarrow{\ \ell\ } & \vec{\text{ws}} \\ \Big\downarrow{\scriptstyle \text{mrg}} & & \Big\downarrow{\scriptstyle \text{d}} \\ \text{so} & \xrightarrow{\ \ell\ } & \text{ws} \end{array}$$

ℓ is a deterministic, multi, nondeleting bottom-up transduction.

Proof: (i) By the definition of ℓ, it is trivially true that so $=$ h(ℓ so) when so is lexical. And in all other cases, the merged element is placed first.

(ii) is almost immediate from our definitions too. 1-1-ness is guaranteed by (i). And for $\vec{so} = \text{so}_1, \ldots, \text{so}_i$ in the domain of mrg and labelable,

$$\begin{array}{rcll} \ell\ (\text{mrg so}_1 \ldots \text{so}_i) & = & \ell\ \{\text{so}_1, \ldots, \text{so}_i\} & (\text{def mrg}) \\ & = & \text{d}\ (\ell\ \text{so}_1) \ldots (\ell\ \text{so}_i) & (\text{def } \ell). \end{array}$$

Function ℓ is deterministic, a function, since d is. It is multi in the sense that an output workspace can have more than one tree in it (up to a fixed, finite bound determined by the lexicon and the smc). And it is nondeleting in the sense that every element of the input is contained in the output.

(31) **The syntactic domain.** All aspects of syntactic structure depend directly on merge relations. Labeling, head movement and linearization map the domain of merge into itself. In particular, labeling is an automorphism. Head movement alters morphophonological content at the leaves, but makes no change in structure or syntactic features. Linearization adds order and unlinks non-final copies.

In a sense, this is the central idea of the modularization of minimalist grammars. A wide range of linguistic properties are defined in terms of the basic merged structure and the features at the leaves.[26]

[25] Given any grammar (i.e. finite set of lexical items), the domain L of mrg is the lexical items together with all finite sets that can be built from them. Function $f : \text{L}^* \to \text{M}$ is a *(partial) homomorphism* on L iff it preserves mrg in the sense that, for mrg' an operation on M, for any $s_0, \ldots, s_i \in \text{L}$, $f\ (\text{mrg}\ s_0 \ldots s_i) = \text{mrg'}\ (f\ s_0 \ldots s_i)$, if f is defined on those arguments. If homomorphism f is 1-1, it is an *isomorphism*. If isomorphism f maps (tuples from) L back into L, it is an *automorphism*. For an introduction to these notions with attention to category-theoretic foundations, see e.g. Aluffi (2009, §I). For an introduction that is carefully extended to partial morphisms, see e.g. Grätzer (1979, §2). For connections to tree automata, see e.g. Shieber (2014).

[26] Hornstein (2024, pp.7-8) proposes "All grammatical relations are merge-mediated" as the "Fundamental Principle of Grammar" (FPG). That FPG sounds very much like (31), but Hornstein extends his FPG with assumptions that are much stricter than (31). In fact, he says that his extended FPG entails that there can be no modularity in the faculty of language. But that latter claim uses 'modularity' in a different sense than ours – recall our informal definition of that notion in §1 above. For example, Hornstein says that specifier-head case assignment is compatible with his extended FPG (pp.82-3). But that theory is modular in our sense, since Hornstein's definition of merge (p.49), like ours, says nothing at all about case; and of course specifiers merge with {head, complement} structures, not directly with the head. Nevertheless, Hornstein rejects many of the proposals considered here as not 'merge-mediated' in a strict enough sense to fall under the FPG as he intends it.

Levine (2024, p.144) quotes the similar observation of Nöth that, even for C.S. Peirce in the 1800's, "the key to syntactic structure is the predicate and its valence" – an idea preserved in contemporary type-logical grammar and implemented here in the labeling rules match and t. In the revised 'unbound' grammar of §1.3 this idea is modified but preserves (31).

Another idea close to (31) that will be increasingly prominent below is that that syntax is the basic generative component of language, so that morphology, in particular, is derived from syntactic structure (Halle and Marantz, 1993, 1994).

(32) *Conjecture:* **Mild context sensitivity.** For any unbounded grammar g, for any category, the string language of complete structures of that category, linearized with ord_{svo}, is mildly context sensitive in the sense that it is defined by a multiple context free grammar (Seki et al., 1991).

Proof: These MGs appear to be weakly equivalent to the systems with binary merge studied by Michaelis (1998). Our head movement is similar to the mirror theory formalized by Kobele (2002), expressively equivalent as long as the non-recursiveness condition in (16) is respected. A rigorous check of these claims is left for future work.

References

Michael Abbott, Thorsten Altenkirch, and Neil Ghani. 2005. Containers: Constructing strictly positive types. *Theoretical Computer Science*, 342:3–27.

Klaus Abels. 2003. *Successive Cyclicity, Anti-Locality, and Adposition Stranding*. Ph.D. thesis, University of Connecticut.

Klaus Abels. 2012. *Phases: An Essay on Cyclicity in Syntax*. Linguistische Arbeiten.

Steven Abney. 1987. *The English Noun Phrase in its Sentential Aspect*. Ph.D. thesis, MIT.

Enoch O. Aboh. 2024. D is not a syntactic parameter. In Anke Himmelreich, Daniel Hole, and Johannes Mursell, editors, *To the left, to the right, and much in between: A Festschrift for Katharina Hartmann*. Goethe-Universität.

David Adger. 2010. Minimalist theory of feature structure. In A. Kibort and G.G. Corbett, editors, *Features*, page 185–219. Oxford University Press.

David Adger, Daniel Harbour, and Laurel Watkins. 2009. *Mirrors and Microparameters*. Cambridge University Press.

David Adger and Peter Svenonius. 2012. Features in minimalist syntax. In C. Boeckx, editor, *Oxford Handbook of Linguistic Minimalism*. Oxford University Press.

Brian Agbayani and Chris Golston. 2010. Phonological movement in Classical Greek. *Language*, 86:133–167.

Brian Agbayani and Chris Golston. 2016. Phonological constituents and their movement in Latin. *Phonology*, 33:1–42.

Kazimierz Ajdukiewicz. 1935. Syntactic connexion. *Studia Philosophica*, 1:1–27. English translation in Storrs McCall, ed., *Polish Logic: 1920-1939*. Oxford: Clarendon Press, 1967.

T. Altenkirch and B. Reuss. 1999. Monadic presentations of lambda terms using generalized inductive types. *Computer Science Logic*, 13:453–468.

Paolo Aluffi. 2009. *Algebra: Chapter 0*. American Mathematical Society.

Karlos Arregi and Asia Pietraszko. 2019. Do-support as spellout of split head chains. *LingBuzz*, 004584.

Karlos Arregi and Asia Pietraszko. 2021. The ups and downs of head displacement. *Linguistic Inquiry*, 52:241–289.

Karlos Arregi and Asia Pietraszko. 2024. The relation between head movement and periphrasis. *University of Chicago. Forthcoming*.

Lennart Augustsson, Joachim Breitner, Koen Claessen, Ranjit Jhala, Simon Peyton Jones, Olin Shivers, Guy L. Steele Jr., and Tim Sweeney. 2023. The verse calculus. *Procs. ACM on Programming Languages*, 7:417–447.

Patrick Bahr. 2012. Modular tree automata. In *Mathematics of Program Construction*, pages 263–299. Springer LNCS 7432.

Mark C. Baker. 1985. The mirror principle and morphosyntactic explanation. *Linguistic Inquiry*, 16:373–416.

Mark C. Baker. 1988. *Incorporation: A Theory of Grammatical Function Changing*. University of Chicago Press.

Peter beim Graben, Ronald Römer, Werner Meyer, Markus Huber, and Matthias Wolff. 2020. Reinforcement learning of minimalist grammars. *Computing Research Repository*, arXiv:2005.00359.

Ryan Bennett, Emily Elfner, and James McCloskey. 2015. Pronouns and prosody in Irish. In *Procs. XIV International Congress of Celtic Studies*, page 19–74. Dublin Institute for Advanced Studies, School of Celtic Studies.

Jean-Philippe Bernardy, Mathieu Boespflug, Ryan R. Newton, Simon Peyton Jones, and Arnaud Spiwack. 2018. Linear Haskell: Practical linearity in a higher-order polymorphic language. *Procs. ACM on Programming Languages, Volume 2*, article 5.

Jean-Philippe Bernardy and Nicolas Pouillard. 2013. Names for free: Polymorphic views of names and binders. In *Procs. ACM SIGPLAN Symposium on Haskell*, pages 13–24.

Robert C. Berwick. 1982. *Locality Principles and the Acquisition of Syntactic Knowledge*. Ph.D. thesis, MIT.

Robert C. Berwick and Amy Weinberg. 1984. *The Grammatical Basis of Linguistic Performance*. MIT Press.

Marc Bezem, Dimitri Hendriks, and Hans de Nivelle. 2002. Automated proof construction in type theory using resolution. *Journal of Automated Reasoning*, 29:253–275.

Richard Bird and Ross Paterson. 1999. de Bruijn notation as a nested datatype. *Journal of Functional Programming*, 9:77–91.

Bronwyn Bjorkman. 2011. *BE-ing Default: The Morphosyntax of Auxiliaries*. Ph.D. thesis, MIT.

Wayne D. Blizard. 1989. Multiset theory. *Notre Dame Journal of Formal Logic*, 30:36–66.

Jonathan David Bobaljik. 1995. *Morphosyntax: The syntax of verbal inflection*. Ph.D. thesis, MIT.

Ingrid Bondevik and Terje Lohndal. 2023. Extraction from finite adjunct clauses. *Glossa*, 8.

Mary Everest Boole. 1909. *Philosophy and Fun of Algebra*. C.W. Daniel.

George S. Boolos, John P. Burgess, and Richard C. Jeffrey. 2007. *Computability and Logic, 5th edition*. Cambridge University Press.

Željko Bošković. 1999. On multiple feature checking: Multiple Wh-fronting and multiple head movement. In Samuel David Epstein and Norbert Hornstein, editors, *Working Minimalism*, pages 159–187. MIT Press.

Željko Bošković. 2014. Now i'm a phase, now i'm not a phase. *Linguistic Inquiry*, 45:27–89.

Željko Bošković. 2020. On the coordinate structure constraint, across-the-board-movement, phases, and labeling. In J. van Craenenbroeck, C. Pots, and T. Temmerman, editors, *Recent Developments in Phase Theory*. De Gruyter.

Phil Branigan. 2023. *The Grammar of Multiple Head Movement*. Oxford University Press.

Michael Brody. 2000. Mirror theory: Syntactic representation in perfect syntax. *Linguistic Inquiry*, 31:29–56.

Benjamin Bruening. 2020. The head of the nominal is N, not D. *Glossa*, 20.

Michael Burke, Aoife Cahill, Mairéad Mccarthy, Ruth O'Donovan, Josef Van Genabith, and Andy Way. 2004. Evaluating automatic LFG F-structure annotation for the Penn-II Treebank. *Research on Language and Computation*, 2:523–547.

William E. Byrd. 2009. *Relational programming in miniKanren*. Ph.D. thesis, Indiana University.

Juan Carlos Castillo, John E. Drury, and Kleanthes Grohmann. 2009. Merge over move and the extended projection principle. *Iberia*, 1:53–114.

Zhong Chen and John T. Hale. 2010. Deforesting logical form. In *The Mathematics of Language*. Springer LNCS 6149.

Eugenia Cheng. 2023. *Is Math Real?* Basic Books.

Cristiano Chesi. 2015. On directionality of phrase structure building. *Journal of Psycholinguistic Research*, 44:65–89.

Adam Chilpala. 2008. Parametric higher-order abstract syntax for mechanized semantics. In *Procs. ACM SIGPLAN International Conference on Functional Programming*, pages 143–156.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton.

Noam Chomsky. 1961. On the notion 'rule of grammar'. In *Structure of Language and its Mathematical Aspects: Procs. 12th Symposium in Applied Mathematics*, pages 6–24.

Noam Chomsky. 1963. Formal properties of grammars. In R.D. Luce, R.R. Bush, and E. Galanter, editors, *Handbook of Mathematical Psychology, Volume II*, pages 323–418. Wiley.

Noam Chomsky. 1975. *Questions on Form and Interpretation*. Peter de Ridder Press.

Noam Chomsky. 1995. *The Minimalist Program*. MIT Press.

Noam Chomsky. 2000. Minimalist inquiries: The framework. In R. Martin, D. Michaels, and J. Uriagereka, editors, *Step by Step: Essays on Minimalism in Honor of Howard Lasnik*, pages 89–155. MIT Press.

Noam Chomsky. 2003. Replies. In L.M. Antony and N. Hornstein, editors, *Chomsky and His Critics*. Blackwell.

Noam Chomsky. 2004. Beyond explanatory adequacy. In A. Belletti, editor, *Structures and Beyond: The Cartography of Syntactic Structures Volume 3*. Oxford University Press.

Noam Chomsky. 2007. Approaching UG from below. In U. Sauerland and H.-M. Gärtner, editors, *Interfaces + Recursion = Language? Chomsky's Minimalism and the View from Syntax-Semantics*, pages 1–30. Mouton de Gruyter.

Noam Chomsky. 2008. On phases. In R. Freidin, C.P. Otero, and M.L. Zubizarreta, editors, *Foundational Issues in Linguistic Theory: Essays in Honor of Jean-Roger Vergnaud*. MIT Press.

Noam Chomsky. 2012. Foreword. In A.J. Gallego, editor, *Phases*. De Gruyter.

Noam Chomsky. 2018a. Syntactic structures. some retrospective comments. In N. Hornstein, H. Lasnik, P. Patel-Grosz, and C. Yang, editors, *Syntactic Structures after 60 Years*. De Gruyter Mouton.

Noam Chomsky. 2018b. Two notions of modularity. In R.G. de Almeida and L.R. Gleitman, editors, *On Concepts, Modules, and Language*. Oxford University Press.

Noam Chomsky. 2020a. Puzzles about phases. In L. Franco and P. Lorusso, editors, *Linguistic Variation: Structure and Interpretation*, pages 163–168. De Gruyter.

Noam Chomsky. 2020b. The UCLA lectures. *LingBuzz*, 005485.

Noam Chomsky. 2021. Minimalism: Where are we now, and where can we hope to go. *Gengo Kenkyu*, 160:1–41.

Noam Chomsky, T. Daniel Seely, Robert C. Berwick, Sandiway Fong, M.A.C. Huybregts, Hisatsugu Kitahara, Andrew McInnerney, and Yushi Sugimoto. 2023. *Merge and the Strong Minimalist Thesis*. Cambridge University Press.

Sandra Chung. 2020. *Chamorro Grammar*. University of California.

Guglielmo Cinque. 1999. *Adverbs and Functional Heads: A Cross-Linguistic Perspective*. Oxford University Press.

Guglielmo Cinque. 2004. Issues in adverbial syntax. *Lingua*, 114:683–710.

Guglielmo Cinque. 2010. *The Syntax of Adjectives: A Comparative Study*. MIT Press.

Guglielmo Cinque. 2023. *On Linearization: Toward a Restrictive Theory*. MIT Press.

Barbara Citko. 2011. Multidominance. In Cedric Boeckx, editor, *Oxford Handbook of Linguistic Minimalism*, pages 119–142. Oxford University Press.

Barbara Citko and Martina Gracanin-Yuksek. 2021. *Merge: Binarity in (Multidominant) Syntax*. MIT Press.

Peter Cole, Yurie Hara, and Ngee Thai Yap. 2008. Auxiliary fronting in Peranakan Javanese. *Journal of Linguistics*, 44:1–43.

Chris Collins. 2002a. Eliminating labels. In Samuel D. Epstein and T. Daniel Seely, editors, *Derivation and Explanation*. Blackwell.

Chris Collins. 2002b. Multiple verb movement in ǂHoan. *Linguistic Inquiry*, 33:1–29.

Chris Collins and Joachim Sabel. 2015. A C/I-interface constraint on remnant movement. In G. Grewendorf, editor, *Remnant Movement*. De Gruyter.

Chris Collins and Edward P. Stabler. 2016. A formalization of minimalist syntax. *Syntax*, 19:43–78.

Duncan Coutts, Roman Leshchinskiy, and Don Stewart. 2007. Stream fusion: From lists to streams to nothing at all. *Procs. ACM SIGPLAN International Conference on Functional Programming*, 12:315–326.

Rowan Davies and Frank Pfenning. 2009. A modal analysis of staged computation. *Journal of the ACM*, 48:555–604.

N. G. de Bruijn. 1972. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indagationes Mathematicae*, 34:381–392.

Mark de Vries. 2002. Three-dimensional grammar. *Linguistics in the Netherlands 2003*, 20:201–212.

Mark de Vries. 2017. Across-the-board phenomena. In *Wiley Blackwell Companion to Syntax, 2nd Ed*. Wiley.

Andrew M. Devine and Laurence D. Stephens. 2000. *Discontinuous Syntax: Hyperbaton in Greek*. Oxford University Press.

Alan D. Dewar and John G. Cleary. 1984. A low-cost high accuracy intelligent backtracking algorithm. In J.A. Campbell, editor, *Implementations of Prolog*, pages 216–233. Ellis Horwood.

Edsger W. Dijkstra. 1982. *Selected Writings on Computing*. Springer.

William F. Dowling and Jean H. Gallier. 1984. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 1:267–284.

Frances Egan. 2003. Naturalistic inquiry: Where does mental representation fit in? In L.M. Antony and N. Hornstein, editors, *Chomsky and His Critics*. Wiley.

Frances Egan. 2025. *Deflating Mental Representation*. MIT Press.

David Embick and Rolf Noyer. 2001. Movement operations after syntax. *Linguistic Inquiry*, 32:555–595.

Joost Engelfriet, Eric Lilin, and Andreas Maletti. 2009. Extended multi bottom–up tree transducers: Composition and decomposition. *Acta Informatica*, 46:561–590.

Samuel D. Epstein, Erich M. Groat, Ruriko Kawashima, and Hisatsugu Kitahara. 2012a. *A Derivational Approach to Syntactic Relations*. Oxford University Press.

Samuel D. Epstein, Hisatsugu Kitahara, and T. Daniel Seely. 2012b. Labeling by minimal search. *Linguistic Inquiry*, 45(13):463–481.

Samuel D. Epstein, Hisatsugu Kitahara, and T. Daniel Seely. 2018. Can determinacy + PIC explain descriptions of remnant movement asymmetries? In *Procs 157th Linguistic Society of Japan*. Paper F-7.

Marina Ermolaeva and Gregory M. Kobele. 2023. Formal properties of agreeing minimalist grammars. *Procs. Society for Computation in Linguistics*, 6:385–388.

Thomas Ernst. 2022. The adjunct condition and the nature of adjuncts. *Linguistic Review*, 39:85–128.

Marcelo Ferreira. 2009. Null subjects and finite control in Brazilian Portuguese. In *Minimalist Essays on Brazilian Portuguese Syntax*, page 17–50. John Benjamins.

Sebastian Fischer, Oleg Kiselyov, and Chung-Chieh Shan. 2011. Purely functional lazy nondeterministic programming. *Journal of Functional Programming*, 21:413–465.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6:15–28.

Sandiway Fong. 1991. *Computational Properties of Principle-Based Grammatical Theories*. Ph.D. thesis, MIT.

Suzana Fong. 2019. Proper movement through Spec-CP: An argument from hyperraising in Mongolian. *Glossa*, 4(1):30.

Michael Fortescue, Marianne Mithun, and Nicholas Evans. 2017. Introduction. In M. Fortescue, M. Mithun, and N. Evans, editors, *The Oxford Handbook of Polysynthesis*, pages 1–16. Oxford University Press.

Meaghan Fowlie. 2011. Multidominant minimalist grammars. UCLA Master's thesis.

Meaghan Fowlie. 2014. Adjunction and minimalist grammars. *Procs. Formal Grammar*, 19:34–51.

Robert Frank and Tim Hunter. 2021. Variation in mild context-sensitivity: Derivational state and structural monotonicity. *Evolutionary Linguistic Theory*, 3:181–214.

Steven Franks. 1993. On parallelism in across-the-board dependencies. *Linguistic Inquiry*, 24:509–529.

Werner Frey and Hans-Martin Gärtner. 2002. On the treatment of scrambling and adjunction in minimalist grammars. *Procs. Formal Grammar*, 7:41–52.

Hiroki Fujita and Shravan Vasishth. 2024. Cue-based memory retrieval in garden-path sentences. *Procs. Cognitive Science Society*, 60:1538–1545.

Paul D. Gallot, Aurélien Lemay, and Sylvain Salvati. 2020. Linear high-order deterministic tree transducers with regular look-ahead. *International Symposium on Mathematical Foundations of Computer Science*, 45:38.

Hans-Martin Gärtner. 2002. *Generalized Transformations and Beyond*. Akademie Verlag.

Hans-Martin Gärtner. 2014. Strange loops: Phrase-linking grammar meets Kaynean pronominalization. *Linguistische Berichte*, 239:383–395.

Hans-Martin Gärtner and Jens Michaelis. 2005. A note on the complexity of constraint interaction. In *Procs. Logical Aspects of Computational Linguistics*, pages 114–130. Springer LNAI 3492.

Hans-Martin Gärtner and Jens Michaelis. 2007. Some remarks on locality conditions and minimalist grammars. In U. Sauerland and H.-M. Gärtner, editors, *Interfaces + Recursion = Language? Chomsky's Minimalism and the View from Syntax-Semantics*, pages 161–196. Mouton de Gruyter.

Hans-Martin Gärtner and Jens Michaelis. 2010. On the treatment of multiple-wh interrogatives in minimalist grammars. In Thomas Hanneforth and Gisbert Fanselow, editors, *Language and Logos*, pages 339–377. Akademie Verlag.

Doreen Georgi. 2017. Patterns of movement reflexes as the result of the order of merge and agree. *Linguistic Inquiry*, 48:585–626.

Doreen Georgi and Mary Amaechi. 2024. Subextraction from subjects in Igbo: New evidence for an antilocality constraint on ā-movement. *LingBuzz*, 008358.

Donna B. Gerdts and Zachary Gilkison. 2018. NP coordination, lists, etc. in Hul'q'umi'num' Salish. In *Society for the Study of the Indigenous Languages of the Americas*.

Edward Gibson. 2024. Composition, not infinity: The irrelevance of recursion to theories of language. In E. Gibson and M. Poliak, editors, *From Fieldwork to Linguistic Theory*, pages 1–22. Language Science Press.

Andrew Gill, John Launchbury, and Simon L. Peyton Jones. 1993. A short cut to deforestation. *Procs. Functional Programming Languages and Computer Architecture*, 93:223–232.

Grant Goodall. 1987. *Parallel Structures in Syntax*. Cambridge University Press.

Thomas Graf. 2011. Closure properties of minimalist derivation tree languages. *Logical Aspects of Computational Linguistics*, 11:96–111. Springer LNAI 6736.

Thomas Graf. 2013. *Local and Transderivational Constraints in Syntax and Semantics*. Ph.D. thesis, UCLA.

Thomas Graf. 2014. Models of adjunction in minimalist grammars. *Procs. Formal Grammar*, 19:52–68. Springer LNCS 8612.

Thomas Graf. 2018. Why movement comes for free once you have adjunction. *Procs. Chicago Linguistic Society*, 53:117–136.

Thomas Graf. 2022a. Subregular linguistics: Bridging theoretical linguistics and formal grammar. *Theoretical Linguistics*, 48(3-4):145–184.

Thomas Graf. 2022b. Typological implications of tier-based strictly local movement. *Procs. Society for Computation in Linguistics*, 5:184–193.

Thomas Graf. 2023. Subregular tree transductions, movement, copies, traces, and the ban on improper movement. *Procs. Society for Computation in Linguistics*, 6:289–299.

George Grätzer. 1979. *Universal Algebra, 2nd Edition*. Springer.

Jane Grimshaw. 1991. Extended projection. Technical report, Brandeis University.

Marjan Grootveld. 1992. On the representation of coordination. *Linguistics in the Netherlands 1992*, 9:61–73.

Jonas Groschwitz, Alexander Koller, and Mark Johnson. 2016. Efficient techniques for parsing with tree automata. *Annual Meeting of the Association for Computational Linguistics*, 54:2042–2051.

Bill Haddican and Anders Holmberg. 2019. Object symmetry effects in Germanic. *Natural Language and Linguistic Theory*, 37:91–122.

John T. Hale. 2014. *Automaton Theories of Human Sentence Comprehension*. CSLI.

Morris Halle and Alec Marantz. 1993. Distributed morphology and the pieces of inflection. In K. Hale and S.J. Keyser, editors, *The View from Building 20*, pages 111–176. MIT Press.

Morris Halle and Alec Marantz. 1994. Some key features of distributed morphology. *MIT Working Papers in Linguistics*, 21:275–288.

Peter Hallman. 2021. Explaining Siewierska's generalization. *Journal of Comparative Germanic Linguistics*, 24:145–184.

Claire Halpert and Hedde Zeijlstra. 2024. Off phases: It's all relative(ized). *LingBuzz*, 008323.

Daniel Harasim, Chris Bruno, Eva Portelance, Martin Rohrmeier, and Timothy J. O'Donnell. 2021. A generalized parsing framework for abstract grammars. *Computing Research Repository*, arXiv.1710.11301.

Henk Harkema. 2001. *Parsing Minimalist Languages*. Ph.D. thesis, University of California.

Heidi Harley. 2013. Getting morphemes in order. In L.L.-S. Cheng and N. Corver, editors, *Diagnosing Syntax*. Oxford University Press.

Thomas Harper. 2011. A library writer's guide to shortcut fusion. *ACM SIGPLAN Notices*, 46:47–58.

William Harwood. 2014. Rise of the auxiliaries: A case for auxiliary rasing vs. affix lowering. *Linguistic Review*, 21(2):295–362.

Martin Haspelmath. 2004. Coordinating constructions: An overview. In M. Haspelmath, editor, *Coordinating constructions*. John Benjamins.

Johannes Hein. 2020. *Verb Doubling and Dummy Verb: Gap Avoidance Strategies in Verbal Fronting*. De Gruyter.

Johannes Hein and Andrew Murphy. 2016. Case matching and syncretism in ATB dependencies. In K. Barnickel, M. Guzmán N., J. Hein, S. Korsah, A. Murphy, L. Paschen, Z. Puškar, and J. Zaleska, editors, *Replicative Processes in Grammar*, page 301–350. Universität Leipzig, Linguistische Arbeits Berichte 93.

Jeffrey Heinz. 2018. The computational nature of phonological generalizations. In L.M. Hyman and F. Plank, editors, *Phonological Typology*. De Gruyter.

Jeffrey Heinz, Chetan Rawal, and Herbert G. Tanner. 2011. Tier-based strictly local constraints in phonology. *Procs. Association for Computational Linguistics*, 49:58–64.

Matthew Hewett. 2023. Allomorphy in Semitic discontinuous agreement: Evidence for a modular approach to postsyntax. *Natural Language and Linguistic Theory*, 41:1091–1145.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33:355–396.

Anders Holmberg, Michelle Sheehan, and Jenneke van der Wal. 2019. Movement from the double object construction is not fully symmetrical. *Linguistic Inquiry*, 50:677–721.

Matthew Honnibal and James R. Curran. 2009. Fully lexicalising CCGbank with hat categories. *Procs. Conference on Empirical Methods in Natural Language Processing*, 3:1212–1221.

Norbert Hornstein. 2024. *The Merge Hypothesis: A Theory of Aspects of Syntax*. Cambridge University Press.

John Hughes. 1989. Why functional programming matters. *Computer Journal*, 32:98–107.

Tim Hunter. 2011. Insertion minimalist grammars: Eliminating redundancies between merge and move. *Procs. Mathematics of Language*, 12:90–107. Springer LNAI 6878.

Tim Hunter. 2015. Deconstructing merge and move to make room for adjunction. *Syntax*, 18:266–319.

Tim Hunter and Robert Frank. 2021. Comparing methods of tree-construction across mildly context-sensitive formalisms. In *Procs. Society for Computation in Linguistics*, volume 4, pages 355–358.

Clara Huttenlauch, Marie Hansen, Carola de Beer, Sandra Hanne, and Isabell Wartenburger. 2023. Age effects on linguistic prosody in coordinates produced to varying interlocutors. In *Prosodic Boundary Phenomena*. Language Science Press.

Adam Jardine and Jeffrey Heinz. 2016. Learning tier-based strictly 2-local languages. *Transactions of the Association for Computational Linguistics*, 4:87–98.

Adam Jardine and Kevin McMullin. 2017. Efficient learning of tier-based strictly k-local languages. In Frank Drewes, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications*, pages 64–76. Springer LNCS 10168.

Otto Jespersen. 1924. *Philosophy of Grammar*. George Allen and Unwin.

Mark Johnson. 1988. *Attribute Value Logic and The Theory of Grammar*. CSLI.

Mark Johnson and Martin Kay. 1994. Parsing and empty nodes. *Computational Linguistics*, 20:289–300.

Aravind K. Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Volume 3: Beyond Words*, pages 69–124. Springer.

Kohei Kajikawa, Ryo Yoshida, and Yohei Oseki. 2024. Dissociating syntactic operations via composition count. *Procs. Cognitive Science Society*, 46:297–305.

Laura Kalin and Nicholas Rolle. 2024. Deconstructing subcategorization. *Linguistic Inquiry*, 55:197–218.

Laura Kalin and Philipp Weisser. 2021. Minimalism and morphology. In *The Cambridge Handbook of Minimalism*. Cambridge University Press. Manuscript, forthcoming.

Makoto Kanazawa. 2009. Second order abstract categorial grammars. Manuscript, ESSLLI.

Max Kanovich. 2015. Horn linear logic and Minsky machines. *Computing Research Repository*, arXiv:1512.04964.

Richard S. Kayne. 1994. *The Antisymmetry of Syntax*. MIT Press.

Richard S. Kayne. 2020. Antisymmetry and externalization. *LingBuzz*, 005554.

Alan Hezao Ke. 2023a. Can agree and labeling be reduced to minimal search? *Linguistic Inquiry*.

Alan Hezao Ke. 2023b. Search downward. *Glossa*, 8.

Edward L. Keenan and Edward P. Stabler. 2010. Language variation and linguistic invariants. *Lingua*, 120:2680–2685.

Maayan Keshev, Mandy Cartner, Aya Meltzer-Asscher, and Brian Dillon. 2024. A working memory model of sentence processing as binding morphemes to syntactic positions. *Procs. Cognitive Science Society*, 46:452–458.

Boram Kim. 2024. Not every finite CP is a phase. *Glossa*, 9:1–43.

Oleg Kiselyov, Aggelos Biboudis, Nick Palladinos, and Yannis Smaragdakis. 2019. Stream fusion, to completeness. *Priciples of Programming Languages*, 17:285–299.

Oleg Kiselyov, Chung chieh Shan, Daniel P. Friedman, and Amr Sabry. 2005. Backtracking, interleaving, and terminating monad transformers. *Procs. ACM SIGPLAN International Conference on Functional Programming*, 10:192–203. Code.

Filipe Hisao Kobayashi. 2020. Proper interleaving of A- & A'-movement: A Brazilian Portuguese case study. *LingBuzz*, 005609.

Gregory M. Kobele. 2002. Formalizing mirror theory. *Grammars*, 5:177–221.

Gregory M. Kobele. 2007. Parsing elliptical structure. Manuscript, Leipzig University.

Gregory M. Kobele. 2011. Minimalist tree languages are closed under intersection with recognizable tree languages. In *Logical Aspects of Computational Linguistics, LACL'11*, pages 129–144.

Gregory M. Kobele. 2012. Ellipsis: Computation of. *WIREs Cognitive Science*, 3:411–418.

Gregory M. Kobele. forthcoming. Minimalist grammars and decomposition. In K.K. Grohmann and E. Leivada, editors, *Cambridge Handbook of the Minimalist Program*. Cambridge University Press.

Gregory M. Kobele, Christian Retoré, and Sylvain Salvati. 2007. An automata-theoretic approach to minimalism. In *Model Theoretic Syntax at 10, ESSLLI'07*.

Hilda Koopman. 2005. Korean (and Japanese) morphology from a syntactic perspective. *Linguistic Inquiry*, 36:601–633.

Yusuke Kubota and Robert D. Levine. 2021. *Type-Logical Syntax*. MIT Press.

Vipin Kumar and Yow-Jian Lin. 1988. A data-dependency-based intelligent backtracking scheme for prolog. *Journal of Logic Programming*, 5:165–181.

Kenneth Kunen. 1980. *Set Theory*. North-Holland.

Stepan Kuznetsov. 2021. Complexity of the infinitary Lambek calculus with Kleene star. *Review of Symbolic Logic*, 14:946–972.

Itziar Laka. 1990. *Negation in Syntax*. Ph.D. thesis, Massachusetts Institute of Technology.

Idan Landau. 2024. Empirical challenges to the form-copy theory of control. *Glossa*, 9.

Jonathan Him Nok Lee, Mark Liberman, and Martin Salzmann. 2024. Do we EXPECT TO find phonetic traces for syntactic traces? *Interspeech*, 2024:4258–4262.

Robert Levine. 2024. A Peircean turn in linguistics: Syntactic-semantic composition as logical inference. In E. Gibson and M. Poliak, editors, *From Fieldwork to Linguistic Theory*, pages 143–177. Language Science Press.

Azriel Levy. 1979. *Basic Set Theory*. Springer.

Mark Liberman. 2021. The distribution of wanna-contraction. University of Pennsylvania. Lecture notes for LING 620, accessed September 2024.

Mark Y. Liberman and Janet B. Pierrehumbert. 1984. Intonational invariance under changes in pitch range and length. In M. Aronoff and R.T. Oehrle, editors, *Language Sound Structure*, pages 157–233. MIT Press.

John W. Lloyd. 1984. *Foundations of Logic Programming*. Springer.

Saunders Mac Lane. 1986. *Mathematics: Form and Function*. Springer-Verlag.

Travis Major. 2021. *On the Nature of "Say" Complementation*. Ph.D. thesis, UCLA.

Andreas Maletti and Giorgio Satta. 2010. Parsing and translation algorithms based on weighted extended tree transducers. *Procs. Workshop on Applications of Tree Automata in Natural Language Processing*, 2010:19–27.

Alexis Manaster-Ramer. 1983. The soft formal underbelly of theoretical syntax. In *Papers from the Nineteenth Regional Meeting*, pages 256–262, Chicago. Chicago Linguistics Society.

Alexis Manaster-Ramer. 1987. Dutch as a formal language. *Linguistics and Philosophy*, 10:221–246.

Alexis Manaster-Ramer. 1988. Review of W. Savitch, E. Bach, W. Marsh, and G. Safran-Naveh, eds., 1987. *Computational Linguistics*, 14:98–103.

M. Rita Manzini and Diego Pescarini. 2022. The clitic string as a pair merge sequence. *Glossa*, 45:6571.

Alec Marantz. 1988. Clitics, morphological merger, and the mapping to phonological structure. In M. Hammond and M. Noonan, editors, *Theoretical Morphology*. Academic Press.

Matilde Marcolli, Robert C. Berwick, and Noam Chomsky. 2023a. Old and new minimalism. *Computing Research Repository*, arXiv:2306.10270.

Matilde Marcolli, Robert C. Berwick, and Noam Chomsky. 2023b. Syntax-semantics interface: An algebraic model. *Computing Research Repository*, arXiv:2311.06189.

Matilde Marcolli, Noam Chomsky, and Robert C. Berwick. 2023c. Mathematical structure of syntactic merge. *Computing Research Repository*, arXiv:2305.18278.

Mitchell Marcus. 1974. Wait-and-see strategies for parsing natural language. *MIT Artificial Intelligence Laboratory, Working paper*, 76.

Mitchell Marcus. 1976. A design for a parser for English. *Procs. ACM*, pages 62–68.

Mitchell Marcus. 1980. *A Theory of Syntactic Recognition for Natural Language*. MIT Press.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn treebank. *Principles and Parameters in Comparative Grammar*, 19:313–330.

Gabriela Matos. 1995. Estruturas binárias e monocêntricas em sintaxe. In *Actas do X Encontro Nacional da Associação Portuguesa de Linguística*, page 301–315. APL e Colibri.

Ora Matushansky. 2006. Head movement in linguistic theory. *Linguistic Inquiry*, 37:69–109.

Conor McBride. 2018. Everybody's got to be somewhere. *Electronic Proceedings in Theoretical Computer Science*, 295:53–69.

Thomas McGee and Idan Blank. 2024. Evidence against syntactic encapsulation in large language models. *Procs. Cognitive Science Society*, 46.

Andrew McInnerney. 2022a. Against the argument/adjunct distinction. *Procs. 45th Annual Penn Linguistics Conference*, 28:13.

Andrew McInnerney. 2022b. *The Argument/Adjunct Distinction and the Structure of Prepositional Phrases*. Ph.D. thesis, University of Michigan.

Jason Merchant. 2011. Aleut case matters. In E. Yuasa, T. Bagchi, and K.P. Beals, editors, *Pragmatics and Autolexical Grammar*. John Benjamins.

Jens Michaelis. 1998. Derivational minimalism is mildly context-sensitive. In *Proceedings, Logical Aspects of Computational Linguistics, LACL'98*, pages 179–198. Springer.

Bartosz Milewski. 2024. Linear lenses in Haskell. BartoszMilewsky.com.

Frederike Moltmann. 1992. *Coordination and Comparatives*. Ph.D. thesis, MIT.

Richard Moot. 2015. A type-logical treebank for French. *Journal of Language Modelling*, 3:229–264.

Lawrence S. Moss. 2018. Non-wellfounded set theory. In *Stanford Encyclopedia of Philosophy*. Stanford University.

Gueron Müller. 1996. A constraint on remnant movement. *Natural Language and Linguistic Theory*, 14:355–407.

Andrew Murphy and Bob Offer-Westort. 2024. The locality of allomorphy: Insights from Bidhaawyeet. *LingBuzz*, 008292.

Ad Neeleman, Joy Philip, Misako Tanaka, and Hans van de Koot. 2023. Subordination and binary branching. *Syntax*, 26:1–44.

Ad Neeleman and Misako Tanaka. 2024. Extraction asymmetries show that type A coordination is adjunction. *Language*, 100.

Frederick J. Newmeyer. 2005. *Possible and Probable Languages: A Generative Perspective on Linguistic Typology*. Oxford University Press, Oxford.

Miki Obata and Samuel David Epstein. 2011. Feature-splitting internal merge. *Syntax*, 14:122–147.

Hiromune Oda. 2021. Decomposing and deducing the coordinate structure constraint. *Linguistic Review*, 38:605–644.

Peter Pagin and Dag Westerståhl. 2010. Pure quotation and compositionality. *Linguistics and Philosophy*, 33:381–415.

Phoevos Panagiotidis and Vitor Nóbrega. 2022. Why we need roots in minimalism. In *The Cambridge Handbook of Minimalism*. Cambridge University Press. Manuscript, forthcoming.

Ivan Perez and Angel Herranz. 2024. Logic programming with extensible types. NASA technical memorandum TM-20240010266.

Colin Phillips. 2003. Linear order and constituency. *Linguistic Inquiry*, 34(1):37–90.

Asia Pietraszko. 2023a. Cyclic selection: Auxiliaries are merged, not inserted. *Linguistic Inquiry*, 54.

Asia Pietraszko. 2023b. Timing-driven derivation of a NOM/ACC agreement pattern. *Glossa*, 8:1–55.

Christer Platzack. 2012. Cross Germanic variation in the realm of support verbs. In P. Ackema, R. Alcorn, C. Heycock, D. Jaspers, J. van Craenenbroeck, and G. Vanden Wyngaerd, editors, *Comparative Germanic Syntax*. John Benjamins.

Jeffrey Poland. 2003. Chomsky's challenge to physicalism. In L.M. Antony and N. Hornstein, editors, *Chomsky and His Critics*. Wiley.

Jean-Yves Pollock. 1989. Verb movement, universal grammar, and the structure of IP. *Linguistic Inquiry*, 20:365–424.

Emil L. Post. 1947. Recursive unsolvability of a problem of Thue. *Journal of Symbolic Logic*, 12:1–11.

Christopher Potts. 2007. The dimensions of quotation. In C. Barker and P. Jacobson, editors, *Direct Compositionality*. Oxford University Press.

Ljiljana Progovac. 1998. Structure for coordination, parts I,II. *Glot International*, 3(7,8).

Geoffrey K. Pullum and Gerald Gazdar. 1982. Natural languages and context free languages. *Linguistics and Philosophy*, 4:471–504.

Michael O. Rabin and Dana Scott. 1959. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125.

Gillian Ramchand and Peter Svenonius. 2014. Deriving the functional hierarchy. *Language Sciences*, 46:152–174.

Stefano Crespi Reghizzi and Valentino Braitenberg. 2003. Towards a brain compatible theory of syntax based on local testability. In C. Martin-Víde and G. Păun, editors, *Grammars and Automata for String Processing*, pages 17–32. CRC Press.

Norvin Richards. 1997. *What moves where when in which language?* Ph.D. thesis, MIT.

Norvin Richards. 2014. *Contiguity Theory*. MIT Press.

Luigi Rizzi. 1990. *Relativized Minimality*. MIT Press, Cambridge, Massachusetts.

Luigi Rizzi. 2009. Some elements of syntactic computations. In Derek Bickerton and Eörs Szathmáry, editors, *Biological Foundations and Origin of Syntax*. MIT Press.

Luigi Rizzi. 2013. Locality. *Lingua*, 130:169–186.

J. A. Robinson. 1965. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12:23–41.

J. A. Robinson. 1979. *Logic: Form and Function*. Edinburgh University Press.

Catherine Rudin. 1988. On multiple questions and multiple wh-fronting. *Natural Language and Linguistic Theory*, 6:445–501.

Pavel Rudnev. 2024. Participles, auxiliaries and the insertion approaches to verbal periphrasis. HSE University, forthcoming.

Jerrold M. Sadock. 2000. Aleut number agreement. In *Procs. Berkeley Linguistic Society*.

Craig Sailor. 2012. Inflection at the interface. Technical report, UCLA. Slightly expanded version appears as §§1-2 of Sailor (2014).

Mamoru Saito. 2015. Remnant movement, radical reconstruction, and binding relations. In G. Grewendorf, editor, *Remnant Movement*. De Gruyter.

Arto Salomaa. 1980. Morphisms on free monoids and language theory. In R.V. Book, editor, *Formal Language Theory*. Academic Press.

Philippe Schlenker, Emmanuel Chemla, Anne M. Schel, James Fuller, Jean-Pierre Gautier, Jeremy Kuhn, Dunja Veselinović, Kate Arnold, Cristiane Cäsar, Sumir Keenan, Alban Lemasson, Karim Ouattara, Robin Ryder, and Klaus Zuberbühler. 2016. Formal monkey linguistics. *Theoretical Linguistics*, 42:1–90.

Stephan Schuhmann, Klaus Herrmann, and Kurt Rothermel. 2008. Direct backtracking. In *Procs. Architecture of Computing Systems*. Springer LNCS 4934.

Luise Schwarzer and Philipp Weisser. 2024. In defense of cyclic coordinate structures. *Procs. North East Linguistic Society*, 54.

Thom Scott-Phillips and Christophe Heintz. 2022. Animal communication in linguistic and cognitive perspective. *Annual Review of Linguistics*, 9:93–111.

Nina Seemann, Fabienne Braune, and Andreas Maletti. 2015. String-to-tree multi bottom-up tree transducers. In *Procs. Association for Computational Linguistics*, page 815–824.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.

Margret Selting. 2019. Lists as embedded structures and the prosody of list construction as an interactional resource. *Journal of Pragmatics*, 39:483–526.

Stuart M. Shieber. 2014. Bimorphisms and synchronous grammars. *Journal of Language Modelling*, 2:51–104.

Stuart M. Shieber, Yves Schabes, and Fernando C.N. Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of Logic Programming*, 24:3–36.

Esturo Shima. 2000. A preference for move over merge. *Linguistic Inquiry*, 32:375–385.

Michael A. Shulman. 2008. Set theory for category theory. *Computing Research Repository*, ArXiv:0810.1279.

Raymond M. Smullyan. 1961. *Theory of Formal Systems*. Princeton University Press, Princeton.

M.K. Snigaroff. 2024. Head movement from non-complements: Evidence from Aleut. *Natural Language and Linguistic Theory*, Forthcoming.

Dominique Sportiche. 2005. Division of labor between merge and move: Strict locality of selection and apparent reconstruction paradoxes. *LingBuzz*, 000163.

Dominique Sportiche. 2024. The coordinate structure constraint: Not a constraint on movement. *LingBuzz*, 007934.

Jon Sprouse, Ivano Caponigro, Ciro Greco, and Carlo Cecchetto. 2016. Experimental syntax and the variation of island effects in English and Italian. *Natural Language and Linguistic Theory*, 34:307–344.

Edward P. Stabler. 1991. Avoid the pedestrian's paradox. In R.C. Berwick, S.P. Abney, and C. Tenny, editors, *Principle-based Parsing: Computation and Psycholinguistics*, pages 199–238. Kluwer.

Edward P. Stabler. 1997. Derivational minimalism. In C. Retoré, editor, *Logical Aspects of Computational Linguistics*, pages 68–95. Springer LNCS 1328.

Edward P. Stabler. 2001. Recognizing head movement. In P. de Groote, G. Morrill, and C. Retoré, editors, *Logical Aspects of Computational Linguistics*, pages 254–260. Springer LNAI 2099.

Edward P. Stabler. 2011. Computational perspectives on minimalism. In C. Boeckx, editor, *Oxford Handbook of Linguistic Minimalism*, pages 617–641. Oxford University Press.

Edward P. Stabler. 2013. Two models of minimalist, incremental syntactic analysis. *Topics in Cognitive Science*, 5:611–633.

Edward P. Stabler. 2019. Three mathematical foundations for syntax. *Annual Review of Linguistics*, 5.

Edward P. Stabler and Kristine M. Yu. 2023. Unbounded recursion in two dimensions, where syntax and phonology meet. *Procs. Society for Computation in Linguistics*, 6.

Miloš Stanojević. 2017. Minimalist grammar transition-based parsing. In *Logical Aspects of Computational Linguistics*, pages 273–290. Springer LNCS 10054.

Miloš Stanojević. 2019. On the computational complexity of head movement and affix hopping. *Procs. Formal Grammar*, 24:101–116. Springer LNCS 11668.

Miloš Stanojević and Edward P. Stabler. 2018. A sound and complete left-corner parsing for minimalist grammars. *Workshop on Cognitive Aspects of Computational Language Learning and Processing*, 8:1–10.

Richard Statman. 1974. *Structural Complexity of Proofs*. Ph.D. thesis, Stanford University.

Toshitaka N. Suzuki and Klaus Zuberbühler. 2018. Animal syntax. *Current Biology*, 29:R669–R671.

Peter Svenonius. 2016. Spans and words. In H. Harley and D. Siddiqi, editors, *Morphological Metatheory*, page 199–220. John Benjamins, Amsterdam.

Gary Thoms. 2022. Preserving the locality of selection with layering derivations. *Generative Linguistics in the Old World*, 45:421–455.

Axel Thue. 1914. Probleme über Veränderungen von Zeichenreihen nach gegebenen Regeln. *Norske Videnskabers Selskabs Skrifter I Mathematisch-Naturwissenschaftliche Klasse, Christiana*, 7. English translation by Jean Berstel, *Axel Thue's papers on repetitions in words: A translation*, 1994.

John Torr. 2017. Autobank: a semi-automatic annotation tool for developing deep minimalist grammar treebanks. *Procs. European Chapter of the Association for Computational Linguistics*, 15:81–86.

John Torr. 2019. *Wide-Coverage Statistical Parsing with Minimalist Grammars*. Ph.D. thesis, Edinburgh University.

John Torr and Edward P. Stabler. 2016. Coordination in minimalist grammars. In *Procs. 12th Workshop on Tree-Adjoining Grammars and Related Formalisms*.

Esther Torrego. 1984. On inversion in Spanish and some of its effects. *Linguistic Inquiry*, 15:103–129.

Lisa Travis. 1984. *Parameters and Effects of Word Order Variation*. Ph.D. thesis, MIT, Cambridge, Massachusetts.

A.S. Troelstra. 1992. *Lectures on Linear Logic*. CSLI.

David Vadas and James R. Curran. 2011. Parsing noun phrases in the Penn Treebank. *Computational Linguistics*, 37:753–809.

Astric van Alem. 2024. Complementizer agreement is clitic doubling. *Natural Language and Linguistic Theory*.

Hendrik van Antwerpen, Casper Bach Poulsen, Arjen Rouvoet, and Eelco Visser. 2018. Scopes as types. In *Procs. ACM on Programming Languages*, pages 1–30.

Rob van den Berg, Carlos Gussenhoven, and Rietveld Toni. 1992. Downstep in Dutch. In G.J. Docherty and D.R. Ladd, editors, *Papers in Laboratory Phonology II*, page 335–358. Cambridge University Press.

Julie A. Van Dyke and Richard L. Lewis. 2003. Distinguishing effects of structure and decay on attachment and repair. *Journal of Memory and Language*, 49:285–316.

Elly van Gelderen. 2024. Review of Merge and the Strong Minimalist Thesis. *Biolinguistics*, 18.

Coppe van Urk. 2015. *A Uniform Syntax for Phrasal Movement*. Ph.D. thesis, MIT, Cambridge, Massachusetts.

Jozina Vander Klok. 2015. The dichotomy of auxiliaries in Javanese: Evidence from two dialects. *Australian Journal of Linguistics*, 35:142–167.

Shravan Vasishth, Titus von der Malsburg, and Felix Engelmann. 2013. What eye movements can tell us about sentence comprehension. *WIREs Cognitive Science*, 4:125–134.

Job Vranish. 2012. MiniKanrenT. *Gitub*.

Dimitrios Vytiniotis, Simon Peyton Jones, and Tom Schrijvers. 2010. Let should not be generalized. *Procs. ACM SIGPLAN workshop on types in language design and implementation*, 5:39–50.

Philip Wadler. 1992. Comprehending monads. *Mathematical Structures in Computer Science*, 2(4):461–493.

Michael Wagner. 2005. *Prosody and Recursion*. Ph.D. thesis, MIT.

David Walker. 2002. Substructural type systems. In B.C. Pierce, editor, *Advanced Topics in Types and Programming Languages*, pages 3–43. MIT Press.

Philipp Weisser. 2024. *A Typology of Shifting Coordinators and what they can tell us about clitics and their formal modelling*. Ph.D. thesis, Universität Leipzig.

Linda Wetzel. 1993. What are occurrences of expressions? *Journal of Philosophical Logic*, 22:215–219.

Linda Wetzel. 2009. *Types and Tokens: On Abstract Objects*. MIT Press.

Zhixuan Yang and Nicolas Wu. 2022. Fantastic morphisms and where to find them: A guide to recursion schemes. *Procs. Mathematics of Program Construction*, 14:222–267.

Michelle Yuan. 2018a. *Dimensions of Ergativity in Inuit: Theory and Microvariation*. Ph.D. thesis, MIT.

Michelle Yuan. 2018b. M-merger and copy spell-out in Inuktitut noun incorporation. In *Procs. NELS 48*.

Michelle Yuan. 2022. Ergativity and object movement across Inuit. *Language*, 98:510–551.

Michelle Yuan and Claudia Juárez Chávez. 2024. Lexical DP Blocking in the person-case constraint as a licensing failure. *LingBuzz*, 008174.

Lingling Zhou, Suzan Verberne, and Gijs Wijnholds. 2024. Tree transformer's disambiguation ability of prepositional phrase attachment and garden path effects. *Procs. Association for Computational Linguistics*, 12:12291–12301.

A. Merge and interfaces: A succinct Haskell implementation of the binary formulation

All our definitions, examples, and a Python version, will be at https://github.com/epstabler/mgt.

```
module MgBin where
import Data.MultiSet (MultiSet, toList, fromList) -- Multiset needed. E.g., use: ghci -package multiset
import Data.List ( partition )
import Data.Bifunctor (Bifunctor, bimap, first, second)

type Label = ([String], [String])   -- (neg features, pos features)
type Agr = (String, String)         -- (phi, case)
type Lex = ([String], (Label, Agr))
data SO = L Lex | S (MultiSet SO) | O PhTree deriving (Show, Eq, Ord)
type WS = ([SO],[Label])
data PhTree = Pl Lex | Ps [PhTree] | Pz deriving (Show, Eq, Ord)

-- merge a sequence of SOs into one SO, one multiset
mrg :: [SO] -> SO
mrg sos = S (fromList sos)

-- append two pairs of lists, coordinate-wise
(+++) :: Bifunctor bf => ([a1], [a2]) -> bf [a1] [a2] -> bf [a1] [a2]
(xs,ys) +++ pairOfLists = bimap (xs++) (ys++) pairOfLists

-- partition a pair of lists (xs,ys) according to whether each (x_i,y_i) has property p
ppartition :: (a1 -> a2 -> Bool) -> ([a1], [a2]) -> (([a1], [a2]), ([a1], [a2]))
ppartition _ ([],[]) = (([],[]),([],[]))
ppartition p (x:xs, y:ys) = let (ps,nonps) = ppartition p (xs,ys) in
  if p x y then (bimap (x:) (y:) ps, nonps) else (ps, bimap (x:) (y:) nonps)

-- partition sequence of WSs to separate WS elements that have a negative element
wssNeg :: [WS] -> ([WS],[WS])
wssNeg = partition ((/= []).fst.head.snd)

-- partition WS elements to separate WS elements with labels beginning with positive feature f
wsPosMatch :: String -> WS -> (WS,WS)
wsPosMatch f = ppartition (\_ y -> ((== f).head.snd) y)

-- check/delete already matched features of head and complement
ck :: [Label] -> [Label]
ck [h,c] = [first tail h, second tail c]

-- delete inert, trival elements of a workspace
t :: WS -> WS
t = snd. ppartition (\_ y -> y == ([],[]))

-- partition list of workspaces into WS with neg f and matching pos f's, and WS of non-matching elements
match:: [WS] -> (WS,WS)
match wss = let ([(so:sos, label:labels)], poswss) = wssNeg wss in
  let f = (head.fst) label in case (wsPosMatch f (sos, labels), poswss) of
    ((([so'],[label']), imOthers), [(so'':_,_)]) ->          -- IM
      if so'' == so' then ( ([so,so'],[label,label']), imOthers ) else error "merge-over-move"
    (([],[]), imOthers), [ws]) -> case wsPosMatch f ws of   -- EM
      (([so'],[label']), emOthers) -> ( ([so,so'],[label,label']), imOthers +++ emOthers )

-- return workspace if it respects smc, else error
smc :: WS -> WS
smc (sos, labels) = if smc' [] labels then (sos, labels) else error "smc violation" where
  smc' _ [] = True
  smc' sofar ((f:_):labels) = (f `notElem` sofar) && smc' (f:sofar) labels
  smc' sofar (_:labels) = smc' sofar labels

-- derivational step, derives WS with a new merged SO and its new label
d :: [WS] -> WS
d wss = let ((sos, labels), others) = match wss in smc (t (mrg sos: tail sos, ck labels) +++ others)

-- extend d (partially) through the domain of merge
ell :: SO -> WS
ell (L lx) = ([L lx], [(fst.snd) lx])
ell (S s) = d (map ell (toList s))
```

A.1.1. Haskell: Head movement with do-support

```
module MgBinH where
import Data.MultiSet (MultiSet, fromList , toList ) -- Multiset needed. E.g. use: ghci -package multiset
import MgBin (SO(S,L), ell , wssNeg, wsPosMatch )
import Data.Bifunctor (first )

hfeats :: [ String ] -> (Int, Bool, [ String ]) -- returns (noOfProbes, strong ?, head without feats )
hfeats [] = (0, False , [] )
hfeats (s:ss) = let (i,s') = deps s in if length s' > 0 && [last s'] == "*" then (i,True, init s':ss) else (i,False,s':ss)
  where deps s = case s of {('-':s') -> first (1+) (deps s') ; _ -> (0,s)}

heng :: SO -> SO -- h' (noOfHeadsNeededAbove,strong?,split?,headsFromAbove) = (break?,headsFromBelow,so)
heng so = case h' 0 False False [] so of { (_, [], so') -> so' } where
  h' :: Int -> Bool -> Bool -> [String] -> SO -> (Bool, [String], SO)
  h' 0 _ _ [] (L lex) = (False, [], L lex)
  h' 1 hiStrong sp hs (L (w,fs)) = let (i',isStrong,w') = hfeats w in let cat = (head.snd.fst) fs in
    if isStrong && not hiStrong then (cat == "V", [], L(w'++hs,fs)) else (cat == "V", w'++hs, L([],fs))
  h' i hiStrong sp hs (S s) = let ([nws],pws:pwss) = wssNeg (map ell (toList s)) in case (head.fst) nws of
      L (w,fs) -> let cat = (head.snd.fst) fs in let (i',strong,w') = hfeats w in
        let i'' = i' + max 0 (i-1) in case (i,i'') of
          (0,0) -> case h' 0 False False [] ((head.fst) pws) of                    -- no chain
            { (br, [], pso) -> (br, [], S (fromList (L (w, fs) : pso : [] ))) }
          (0,1) -> let (br,hs',pso) = h' i'' strong False w' ((head.fst) pws) in -- chain begins
            (br, [], S (fromList (L (hs', fs) : pso : [] )))
          (0,_) -> let (br,hs',pso) = h' i'' strong False [] ((head.fst) pws) in -- chain begins
            (br, [], S (fromList (L (w' ++ (foldl (\acc x -> x : acc) [] hs'), fs) : pso : [] )))
          (1,0) -> let (br,hs',pso) = h' 0 False False [] ((head.fst) pws) in     -- chain ends
            if strong && not hiStrong
            then (cat == "V", [], S (fromList (L (w' ++ hs, fs) : pso : [] )))
            else (cat == "V", w' ++ hs, S (fromList (L ([], fs) : pso : [] )))
          (_,_) -> if sp && (head.snd.fst) fs == "v" && strong
                   then let (br, hs',pso) = h' i'' strong False w' ((head.fst) pws) in
                     if br
                     then (br, ["DO"] ++ hs, S (fromList (L (hs' , fs) : pso : [] )))
                     else if strong && not hiStrong
                          then (br, [], S (fromList (L (hs' ++ hs , fs) : pso : [] )))
                          else (br, hs' ++ hs , S (fromList (L ([], fs) : pso : [] )))
                   else let (br, hs',pso) = h' i'' (max strong hiStrong) sp (w' ++ hs) ((head.fst) pws) in
                     if strong && not hiStrong
                     then (br, [], S (fromList (L (hs', fs) : pso : [] )))
                     else (br, hs', S (fromList (L ([], fs) : pso : [] )))
      nso -> if not sp && nonMovingPso nws pws
        then let (br,hs',nso') = h' i hiStrong True hs nso in let psos = map (head.fst) (pws:pwss) in
          (br, hs', S (fromList (nso' : psos)))
        else let (br,hs',nso') = h' i hiStrong sp hs nso in let psos = map (head.fst) (pws:pwss) in
          (br, hs', S (fromList (nso' : psos)))

nonMovingPso nws pws = let (nso:nsos, nlabel : nlabels ) = nws in
  let f = (head.fst) nlabel in case wsPosMatch f (nsos, nlabels ) of
    (([pso],[plabel]), _) -> if (length (snd plabel)) > 1 then False else True
    (([],[]), _) -> case wsPosMatch f pws of {(([pso],[plabel]), _) -> if (length (snd plabel)) > 1 then False else True}
```

A.1.2. Haskell: Linearization

```
module MgBinO where
import Data.MultiSet (MultiSet, toList ) -- Multiset needed. E.g. use: start ghci -package multiset
import MgBin (SO(S,L,O), PhTree(Pl,Ps), WS, ell, wssNeg, wsPosMatch)

o_svo :: SO -> SO
o_svo (L lex) = O (Pl lex)
o_svo (S s) = let ([(nso:nsos, nlabel : nlabels )], poswss) = wssNeg (map ell (toList s)) in
              let f = (head.fst) nlabel in case (wsPosMatch f (nsos, nlabels ), poswss) of
                ((([pso],[plabel]), _), _) -> formatPhTree (length (snd plabel)) nso pso -- IM
                (((([],[]), _), [ws]) -> case wsPosMatch f ws of
                  (([pso],[plabel]), _) -> formatPhTree (length (snd plabel)) nso pso -- EM
  where formatPhTree noOfPsoPosFeats nso pso = case (noOfPsoPosFeats, nso, o_svo nso, o_svo pso) of
          (1, L _, O pht, O pht') -> O (Ps [pht, pht']) -- first merge, nonmoving complement
          (1, S _, O pht, O pht') -> O (Ps [pht', pht]) -- nonfirst merge, nonmoving complement
          (_,  _, O pht, _) -> O pht                    -- moving complement
```