A Categorial Grammar View of Syntactic Categories:

Some Unexpected Evidence from Coordination

Pauline Jacobson
Linguistics Program, Brown University
pauline_jacobson@brown.edu

1. Introduction

The goal of this paper is to show that under a rather natural view of syntactic categories compatible with the general theory of Categorial Grammar (hereafter, CG), some surprising and hitherto undiscussed (to my knowledge) facts about coordination fall out. It is standard in CG to maintain that there is a small set of primitive categories and a recursive definition of others, the latter are often referred to as 'function' categories (not to be confused with the notion of 'functional categories' in other theories). While expressions of one of these function categories have meanings which are functions, the use of the term 'function' in the syntax is sometimes metaphorical, or sometimes meant as functions from categories to categories (e.g., Steedman 2023). But here we advocate a view in which each such category corresponds (oversimplifying for the moment) to a function from strings to strings. We then show that combining this with a view making use of additional operations such as function composition (see, e.g. Steedman 1987, Dowty 1988) we make some striking predictions about what compositions are allowed and what aren't - predictions borne out by some subtle empirical facts surrounding coordination.

To give a spoiler alert: the view advocated here effortlessly handles the existence of surprising sentences like (1):

1. a. Barry plays the dulcimer, the standup bass, and (he) does magic tricks.

b. Sarah studies syntax in the morning, semantics in the afternoon, and (she) plays violin in the evening .

c. Barry plays the dulcimer, the standup bass, does magic tricks, and he builds treehouses. .

Sentences of the form in (1b) were first noted in Maxwell and Manning 1996 but have received little attention, and (1a) is a simplified version making the same point. It is surprising to see an ordinary object NP[1] coordinated with a full VP or even a full S; (1b) has the extra fanciness of including an instance of 'nonconstituent coordination'. And note that there are cases where the last S doesn't have the same subject; imagine (2) in the context of a discussion about a performing duo:

2. Barry plays the dulcimer, the standup bass, does magic tricks, and Patty plays the crumhorn.

It should be stressed that these sentences (and others in this paper) crucially use what we can call 'comma' or 'silent' *and* for all but the last one. (When we have a series of these I will refer to them as comma chains). Note that we also get comma *or* chains (with *or* semantics throughout) as in (3). (4) shows the same surprise with *or* chains:

3. Lee, Sandy, or Kim will win the graduation prize.

4. a. At the concert tomorrow if we want to spiff things up, Barry can play the washboard, the jug, or do magic tricks. (*NP, NP, or VP)* (see fn. 1 on "NP")

b. At the concert tomorrow to spiff things up , Barry can play the washboard, the jug, do magic tricks, or Heidi can put on a silly hat. (*NP, NP, VP, or S*)

The reason I crucially use comma chains is that if we take (1a), for example, and replace the 'silent' conjuncts with overt *and,* the analysis would be quite mundane:

5. Barry plays the hammered dulcimer and the bass and (he) does magic tricks.

(5) simply involves coordination of NPs (actually, generalized quantifiers (GQs), see fn. 1), which means we have the complex NP (or, GQ) *the hammered dulcimer and the bass* which is object of *plays.* That VP in turn coordinates with *does magic tricks.* (In the case where the last conjunct is *he does magic tricks* we have S coordination.) Indeed one referee suggests that that is also the correct analysis of (1a) and that - as such - there is nothing remarkable about (1) nor about most of the facts in Sec. 5. But one cannot simply assume that the cases above reduce to similar ones with overt *and* and *or*. This is because the silent versions have additional constraints on their distribution and meaning. For example, if we assumed that *the hammered dulcimer, the standup bass* was just ordinary NP (GQ) conjunction - exactly as if it were *the hammered dulcimer and the standup bass* - its well-formedness and interpretation would be independent of the fact this expression is followed by *and (he) does magic tricks.* We would thus incorrectly predict **Barry plays the standup bass, the hammered dulcimer* to be perfect, as well as **Tom, Dick left.* While some speakers do marginally accept comma chains with no overt *and,* these require more than two expressions in the chain as in *I saw Tom, Dick, Harry* or an example like *Tom, Dick, Harry left.* (These also have a prosody which suggests an incomplete list.). And other speakers find these quite marginal. More tellingly, even for speakers who do accept some of these, no one interprets these as having the same meaning as *Tom or Dick or Harry left,* but nothing would preclude that interpretation if 'silent' *or* were just like overt *or.* Equally

strikingly, (1a) has no interpretation where the interpretation of the conjuncts is mixed; it cannot for example be understood as *Barry plays the dulcimer and the standup bass or does magic tricks* or as *Barry plays the dulcimer or the standup bass and does magic tricks.* The empirical generalization is that a comma chain requires the presence of an overt *and* or *or* as the rightmost part, and the interpretation of the whole is governed by which is the overt conjunction. Thus (1) - (4) do not trivially reduce to ordinary coordination of likes.[2]

There is a further surprise not noticed by Manning and Maxwell (who did not explore the domain in any detail). This is the fact that when there is this type of comma chain, the 'bigger' categories can only be on the right. Thus notice that in all the above cases the constituents 'grow' as we move right. For example, (1c) and (2) are of the form NP, NP, VP, and S. But we cannot mix these to give the order VP, VP and NP with the obvious interpretation of his playing the standup bass (it does have a pragmatically infelicitous interpretation whereby he "does the standup bass", see Sec. 5.4.)

6. *Barry plays the dulcimer, does magic tricks, and the standup bass.

Thus this paper shows among other things that: (1) that the existence of the Maxwell/Manning (hereafter MM) cases is unsurprising, and (2) that the 'Grow Only Rightward' phenomenon also follows from the view of syntactic categories here. In addition to these results, we show (Sec. 4) some interesting facts about Right Node Raising (RNR) which also follow from this view of categories.

By way of a roadmap: Sec. 2 elucidates the background. Most of that (up to Sec. 2.6) is not new and is based on the framework outlined in Jacobson 2014, but is included here for those unfamiliar with CG. Sec. 3 discusses the status of the Coordinate Structure

Constraint (CSC) - arguing with many others (e.g., Lakoff, 1986, Kuno, 1987, Kehler 1996) that the effects are not due to syntax. There we also develop an account of comma chains. Sec. 4 turns to the interaction of comma chains with RNR. We show that while there are chans of conjuncts - some with gaps and some without - in RNR (akin to facts noticed in Lakoff, 1986 for "*wh*-extraction cases) the rightmost one has to have a gap. This turns out to also follow from the view of syntactic categories advocated here. But this also contrasts interestingly with cases standardly going under the rubric of *wh*-extraction, and so a secondary result here is to provide new evidence that 'gaps' in RNR type cases need to be treated differently from those in *wh*-extraction. (See, among others, McCloskey 1986 for previous evidence.) Sec. 5 turns from the RNR situation to a parallel one on the left - i.e. to the MM cases and the Grow Only Rightward generalization. Appendix 1 discusses (and argues against) a 'flat' analysis of comma chains; and Appendix 2 considers an alternative ellipsis analysis of the MM cases, pointing out some ways in which it is problematic.

2. Background Pieces

2.1. Some basics of Categorial Grammar

We begin with some basics of the syntax (and semantics) of Categorial Grammar (CG). While there actually are many varieties of CG and related work in the Type Logical tradition, I base most of this on a version elucidated in Jacobson 2014. This is similar in some ways to what is known as Combinatory Categorial Grammar (CCG) as in Steedman, 1987, 2023, and many papers in between, but also departs from that in some significant ways; those central to the present account will be noted as appropriate.

Our point of departure is hardly unique to CG: we take any linguistic expression (of any size) to be a triple of <[sound] (represented here with orthography), Category, meaning>. By meaning we mean a model-theoretic object – not a representation in a symbolic language, although such representations are used as ways to *name* model theoretic objects. One might wonder wherein is encoded the structure of an expression - i.e., its tree representation. Under the view here a tree is nothing more than a representation of how an expression is put together and plays no role in the grammar itself. Put another way: the grammar does not 'see' trees. (With one caveat: below we explore the possibility that a small bit of internal structure needs to be kept track of for infixation operations.)

Second, we can think of syntactic categories as encoding an expression's distribution. Moreover, in most versions of CG (including the version here) the category also encodes the type of meaning that an expression has; we return below. Thus there are a few primitive categories, and a recursive definition of all others. For the primitives, we begin with the set {NP (or, DP - see fn. 1), S, N, PP, CP}; there could be a few others. Other categories are defined recursively. As a first pass we can say that if A is a category and B is a category than A/B is a category; call this a 'function' category. The interpretation of this is that an expression of category A/B wants to combine with an expression of category B to give as a result an expression of category A. (So, in the usual tree terms, the B-expression will be a sister of the A/B expression, and the mother will be A.). For example, we can think of a "VP" as being actually S/NP (or, perhaps S/X if categories besides NP occur as subjects), though I often use "VP" as an abbreviation for this category.

As is standard, we also assume that there is a set of features that can occur on these categories. An interesting question arises within CG, which will be relevant later. This is the question of whether features should always be on (or part of) some basic category, or whether a feature can 'span' a function category such as A/B. It turns out both to simplify how features pass and to have added bonuses if we assume that in those cases where we might intuitively think of a feature as 'spanning' a category, it is actually encoded on the result. By the 'result' category, we mean that basic category that we would get when an expression with a complex category takes all of its arguments. For example, take the category ((S/(S/NP))/N) (arguably the category for *every*), here the leftmost S is the 'result'. Notice that if even the so-called 'basic' categories like NP, S, etc. are actually themselves feature bundles - a view first put forth in Chomsky 1970 - then having features always be a part of a basic category is exactly what we would expect. The extra features are really no different than the ordinary basic category features (again see Chomsky, 1970, Gazdar et al, 1985 for explicit discussion). That said, wherever we aren't spelling out a full category but instead using some label "X" as an abbreviation of a complex category, we write X[F] for convenience, with the understanding that the feature F is actually on the result category. And also for expository convenience we will treat the basic category labels (e.g, NP) differently from the additional features. This is fairly customary.

Returning to more detail about the CG categories, note that merely saying something is of category A/B does not specify how this expression combines with a B to give A. We therefore elaborate the recursive definition of the 'function' categories as follows: If A is a category and B is a category then $A/_RB$ is a category and $A/_LB$ is a

category. The intuition behind these categories is that each encodes how the relevant expression combines with its 'argument'. An expression of category $A/_RB$ takes a B to its right to give as result an expression of category A, and an expression of category $A/_LB$ takes its argument to the left. (Other authors use other notations.)

This is all combined with a semantics. As noted above, the syntactic category of an expression also encodes its semantic type. We assume that all expressions of a single category have the same type of meaning. Beginning with the basic categories, assume that any expression of some category A has as its extension a member of some set - call that a. All NPs, for example, have as their extension some individual; call the set of individuals e. Their full meaning is a function from worlds and times to individuals but we ignore intensions throughout - this again is just for expository convenience. Note that this means that a generalized quantifier like *every duck* is not an NP see Sec. 2.5.1. In addition, we take any category of the form A/B to have as its meaning some function from the set of things denoted by B-type expressions (call that set b) to some member of the set denoted by A-type things (all that a). Using standard notation, we say that an expression of category A/B denotes a function 'of type' <b,a> - that is, some function in bXa. Thus assuming that the extension of any S is a truth value (a member of the 2-member set t), any S/NP (a VP) denotes some <e,t> (a function from individuals to truth values). This holds for both single word expressions like *walked* and more complex ones like *chewed gum.* (CG makes no distinction between lexical and phrasal categories; but should the grammar need to refer to cases where the category consists of a single word this could be done in various ways.) Thus *walks,* has as its category $S/_LNP$ and a meaning of type <e,t>. A transitive verb *likes*

is of category $(S/_{L}NP)/_{R}NP$ and thus has as its meaning some function in $\langle e,\langle e,t\rangle\rangle$. Given that the meaning of any A/B is of type <b,a> and the meaning of its sister B is of type b, it is quite unsurprising that the way the semantics works is that the meaning of the A/B expression takes the meaning of the B expression as argument.

To make this explicit, we temporarily posit two rule schemas:

(7) a. Given an expression α of the form $\langle[\alpha]; A/_{R}B; [[\alpha]]\rangle$ and an expression β of the form $\langle[\beta]; B; [[\beta]]\rangle$, there is an expression of the form $\langle[\alpha\beta], A, [[\alpha]]([[\beta]])\rangle$.

b. Given an expression a of the form $\langle[\alpha], A/_{L}B, [[\alpha]]\rangle$ and an expression β of the form $\langle[\beta], B, [[\beta]]\rangle$, there is an expression of the form $\langle[\beta\alpha], A,[[\alpha]]([[\beta]])\rangle$.

By way of illustration, (7a) ensures that *walks* - of category $S/_{L}NP$ - can combine with *Lee* of category NP to give *Lee walks* of category S and whose meaning is [[walks]]([[Lee]]).

But the two schemas in (7) look surprisingly alike – the only difference is in the directionality of the two expressions. This suggests that they should be collapsed. Of course, if we are just talking about having one collapsed rule vs. two rules, the difference is negligible. But if there are additional operations such as the case of a function taking its argument as an infix (Bach 1979) or the reverse - where the function itself is an infix - then there would be more rules to collapse. And the crucial result in this paper is that the tool needed to collapse these is exactly what leads to the main predictions below.

Hence, we posit that what have been called "function categories" really do correspond not just to functions in the semantics (or functions from categories to categories) but - in parallel fashion - to actual functions on strings.[3] The intuition here is that when the expression *walks* (a triple) combines with *Lee* (another triple) we have not

only the result that in the semantics [[walks]] applies to some individual l, but we also will have a function that maps the string [walks] to the string [Lee walks].

A few steps are needed for this. First, we don't want the categories themselves to actually be functions; rather each function category corresponds to a function. One reason for this is that basic categories obviously cannot be functions - in fact we can think of each such category as the name of a set of strings. This then should also be true of 'function' categories; $S/_{L}NP$ names a set of strings (e.g,{[walks], [chews gum], [grunts], ...}. But it also corresponds to a function. Take the function corresponding to the category $S/_{L}NP$. While one might think at first that that should be a function mapping NP strings to S strings, we need one more step. This is because the function corresponding to the category $S/_{L}NP$ can map [lee] to [lee walks] only if it first has access to [walks]. This is perfectly simple: a function corresponding to any category A/B is a function in <string, <string, string>>; it takes two strings as arguments (one at a time) to return a third. The first is the string category A/B, and the second is the string of category B, yielding a string of category A.

By way of example, there is a function corresponding to the category $S/_{L}NP$. That function applies to the string [walks] to give the function mapping [lee] to [lee walks], [sally] to [sally walks], etc. Some notation will be useful. For any (function) category X, we refer to the corresponding function as F:X. (read that as 'the function corresponding to the category X'). Similarly, for every function F which corresponds to a category, we refer to the category as CAT:F. We then define the relevant functions as follows:

(8) a. $F:A/_{R}B([x])([y]) = [xy]$ b. $F:A/_{L}B([x])([y]) = [yx]$

In prose, (8a) says that the relevant function first applies to a string [x] (the sound part of an expression of category $A/_RB$) and then to a string [y] (of category B) to give [xy]. The converse is (8b). With this, we collapse (7) into a single schema as follows:

(9) Given an expression α of the form <[α], A/B, [[α]]> and an expression β of the form <[β], B, [[β]]>, there is an expression of the form <[F:CAT_α ([α])([β])], A, [[α]]([[β]])>.

(CAT_α means the category of α.) To illustrate, *walks* is a triple of the form <[walks], $S/_LNP$, [[walks]]>. (9) says this combines with a triple like <[Lee], NP, l> to give <[Lee walks], S, [[walks]](l)>. The syntactic category of the result and the semantics are familiar; new here is that the actual string results from applying $F:S/_L$ NP to [walks] and then to [Lee].

2.2. Word order rules

The ultimate word order in a sentence is determined by the categories of the lexical items. But obviously these should not be listed on a case-by-case basis. For example, we clearly don't what every intransitive verb in English to just happen to be listed in the lexicon with category $S/_LNP$ as opposed to $S/_RNP$. To this end, Jacobson 2014 suggests that lexical items are (in general) 'born' in an underspecified form and thus without directional slashes, and those are added by general rules. A first pass at these rules would be as follows (from Jacobson 2014); the .... on either side of the category means that these rules apply even when that category is embedded in others. The intent also is that the directional slashes are only added where there are none; no rule overrides already assigned directional features:

(10) a. ...(S/X)... → ... ($S/_LX$)...

b. ...(X/X)... → ...($X/_LX$)...

c. The default rule: ...(A/B) → …($A/_RB$)…

Being the default, the rule in (c) has to apply after (a) and (b).

2.3. Infixation

What about 3-place verbs as in *give the bone to Fido*? Space precludes a thorough treatment, but some remarks are in order as this interacts with the material in Sec. 4. Following a long tradition begun originally in Chomsky 1957 and then generalized and extended (using various formalisms) in, e.g. Bach 1979, 1980, Dowty 1982 (both of those within CG); Pollard 1984 within an extension of GPSG which later became HPSG, Jacobson 1987 within GPSG, and Larson 1988 within a movement based theory - it is possible that *give* first combines with *to Fido* and that subsequently *the bone* is introduced. Rather than using movement to account for the expressions which contain a 'discontinuous constituent', Bach proposed that *the bone* is directly introduced as an infix; he referred to this as Wrap. If this is correct, we need another directional feature - call it I - indicating the argument is taken as an infix. Thus *give* would have the category $((S/_{L}NP)/_{I}NP)/_{R}PP$ (borrowing familiar terms from morphology, this means that *give to Fido* is a circumfix).

To fully work out the details of a system with infixation, we also need to assume that expressions are not just strings, but strings with a dedicated infixation point or, as in Pollard 1984, strings with additional information from which can be derived an infixation point. (Hence when *give to Fido* combines with *the bone,* the grammar needs to know that the latter is infixed after the verb). One system is in Pollard 1984, who takes each string to be a <u>headed</u> string, If we borrow this idea, we can suppose that infixes in English go to the right of the head. Hence, *give to Fido* is a string headed by *give,* so *the bone* is infixed in to the right of *give.* Also needed are conventions as to whose infixation point - or head -

is inherited in the new expression. We assume in general this is inherited from the function category (but see the remarks below on Lift). And in a full system with infixation we will also want a 'circumfix slash' ($A/_{C}B$) indicating that the expression with this category is itself an infix - i.e., it takes its argument as a circumfix. There remain pieces to be fully incorporated into the general view of categories in this paper, some of which are mentioned later when relevant. But a full formulation of a grammar with infixation is beyond the present scope (see Aus, in preparation).

Nonetheless, there is one immediate question of crucial relevance to the material in Sec. 4. This is as follows: if there are infixation slashes, then after the word order rules apply, what is the actual category of a simple transitive verb like *likes?* The most general statement of the rule adding a I-feature to a slash is for the rule to map any item of category ((S/X)/NP)/... to ((S/X)/$_{I}$NP)/... That means that even a simple transitive verb takes its object as an 'infix', but in the case where there is no other argument, the infixation is vacuous (assuming that the infixation point is to the right of the verb, 'infixation' yields the same result as does placing the argument to the right). That said, the remarks below are greatly simplified if we continue to treat an ordinary transitive verb as being of category ($S/_{L}NP)/_{R}NP$ - which is what we would get if the infix slash is only specified on items of category ((S/X)/NP)/Y for Y not null (i.e., only on 3-place verbs). Since this is a slight complication in the statement of the word order rules, one might wonder whether treating transitive verbs as being marked with R rather than I on the object position is justified.

I believe it is. First, it is certainly not a major complication to restrict the infixation rule to only 3 place verbs. Second, everything of category (($S/_{L}X)/_{I}NP$)/... also has a

version where the I slash is an R slash - this is needed for the case of "Heavy NP Shift". As such, there is no harm in working with the R-slash version of transitive verbs. ("Heavy NP Shift" would be vacuous in the case of simple transitive verbs.) In fact using the R-slash version with transitive verbs plays a role here primarily in the analysis of Right Node Raising (RNR), and that has the same basic prosody as does Heavy NP Shift. Third - and most importantly - even leaving aside the Heavy NP Shift version of the category and even if we do make the assumption that ordinary transitive verbs take their objects by vacuous infixation, I suspect everything being said in Secs. 4 using our simplification would carry over directly to a more fully articulated theory with infix (and circumfix) slashes.

2.4. *and* and *or*

We turn next to the category for *and* (and *or*). We assume that *and* is listed in the lexicon as a cross-categorial item: (X/X)/X, with the generalized semantics given in Partee and Rooth 1983; call this Boolean *and* (and *or*).[4] (We slightly refine its category in Sec. 3.3.) Note that the proposed directional features predict that this will be fleshed out as $(X/_{L}X)/_{R}X$. There is, of course, nothing novel about this; most theories of word order principles make the analogous prediction. See, e.g, Munn 1993. Aside from theoretical considerations, empirical support for this is found in, e.g., Munn 1993 (See Appendix 1.)

2.5. Additional combinatory rules and applications to conjunction

2.5.1. Lift

Consider the syntactic category of generalized quantifiers like *every dog.* Under the view of the syntax/semantics match here, these cannot be NPs as they do not denote individuals. Using the fairly standard wisdom since Montague 1973, their semantic type

is <<e,t>,t>. In subject position they take the VP as argument which leads us to the conclusion that their syntactic category is $S/_R(S/_LNP)$, and a quantificational determiner like *every* is of category $(S/_R(S/_LNP))/_RN$. (This says nothing about generalized quantifiers in object position; see Hendriks 1993 and Jacobson 2014 for relevant discussion). But note an anomaly about this category: the directional slashes are not expected. The L-slash is not surprising and follows from the word order rules given in (7), and the directional feature on the final slash in *every* is the unsurprising default R feature. What is surprising is the first (reading left to right) R slash. Should this not be an L-slash, since in general whatever combines with any X to give an S takes that S to the left? We return to this directly.

Partee and Rooth 1983 proposed that ordinary NP meanings (individuals) can also lift to generalized quantifier meanings (type <<e,t>,t>) (i.e., the meaning in Montague 1973) such that if the name *Lee* denotes the individual l, then the lifted meaning of *Lee* is $\lambda P[P(l)]$ – i.e. function mapping any <e,t> function P to the value that P assigns to l (in set terms, the set of sets with l as member). This was motivated by the fact that *Lee*, for example, can conjoin with a generalized quantifier (*Lee and every candidate for schoolboard will speak at the meeting*). But rather than simply list the meaning of *Lee* in the lexicon as the 'fancier' meaning (as Montague did), Partee and Rooth argued that it is 'born' with the simpler meaning (of type e). We adopt the Partee and Rooth semantics, and given the rest of the premises of CG we, the lift rule also must also map the syntactic category NP to the generalized quantifier category $S/_R(S/_LNP)$.

Moreover, we assume that the rule is much more general (for some motivation see Dowty, 1988), hence we allow any expression to lift. By way of formalizing this, notice

that lift is an operation which is defined perfectly generally and is independent of its use in grammar. Take any set A, any member a of this set, and any set of functions <A,B> (i.e., the set of functions in AXB). Then we define (Lift(a)) as that function taking as argument any function f in <A,B> and mapping f to that value that f assigns to a. The result is thus something in B, so Lift(a) is a function in <<A,B>,B>. We need to be a bit more precise: Lift(a) does not yield a unique function because it depends what set we are lifting 'over'; hence when needed we will write $\text{Lift}_{<A,B>}$(a) to mean the function whose domain is the set of functions <A,B>. Restating this informally so as to make the intuition clear, we start with some object a. The lifted a is now the 'boss', it takes some function f (that could have taken a as argument), and gives back just what would have resulted had f applied a.

Given the rest of this version of CG, we also want to tie this in to the syntax. Hence we can write a rule mapping a single triple into a new one, differing in meaning and category but not in sound. This is an instance of what is often called a type shift rule, but we avoid that terminology as the rule it also affects the syntactic category. Note further that there can be no notion of 'type shift as a last resort' in the architecture of the grammar assumed here. That makes sense only if the syntax precomputes representations which are 'sent' to the semantics whose job is to compositionally interpret these (so if it encounters a type mismatch, it has at its disposal a set of type shift rules to 'repair' this). Here the syntax and semantics are computed in tandem so these notions are inapplicable. The general lift rule is (preliminarily) given in (11). Note that because there are two possible outputs in terms of the syntactic category, this is actually two rules which we exhibit together:

(11) Given a linguistic expression of the form <[α], A, [[α]]>, there is an expression β of the form <[α], $B/_L(B/_RA)$ or $B/_R(B/_LA)$, [[$lift_{<a,b>}$(α)]]>.

The only difference between the two rules in (11) is the syntax of the new category. Aside from the discomfort of really having two lift rules packed in here, this looks suspicious for another reason: why in both cases does the new category just happen to preserve word order? For example, when *Lee* lifts over VPs, why is the result of combining lifted *Lee* with *walks* exactly the same as *walks* combines with unlifted *Lee?* Is it just a stipulation that there is no way to apply lift to yield, for example, $B/_L(B/_LA)$ as the new category?

The reader may have anticipated the answer: if we take seriously the notion of syntactic categories as corresponding to functions, then the word order preservation property of Lift is no accident. By definition, a lifted object takes some function as argument to give exactly what would have resulted had that function taken the original object as argument. Hence we merely need to extend this notion to the syntactic categories. The translation into categories is, unfortunately, not immediate for two reasons. First the categories themselves are not functions but correspond to functions, Second, there is the extra layer that the function first needs to apply to the string (sound) part of the expression in order to be a function of the right type. But we can nonetheless extend the basic idea of lift to syntactic categories as follows. First we define F'-α as that function in <string,string> obtained by applying $F:CAT_\alpha([\alpha])$. In prose, F'-α is that function from strings to strings that results from applying the function corresponding to the category of α to the string [α]. Since the syntactic part of the lift rule maps a category A to something

that takes expressions of category B/A as argument, we notate that as $\text{Lift}_{A/B.}$ With this we can see the sense in which the syntactic lift is the same as the general lift operation:

(12) Given any category A, $\text{Lift}_{A/B}(A)$ is that category C such that for all expressions α such that $[\alpha]$ is in A and all expressions β such that $[\beta]$ is in (i.e. A/B),

$F'\text{-}\alpha([\beta]) = F'\text{-}\beta([\alpha])$. (Recall that the categories are also names of sets of strings.)

In other words, the new category C is such that for any string [x] in A and any string [y] in A/B, applying the function corresponding to C to [x] is a function mapping [y] to the string which is the string resulting from applying F:A/B to [y] and the to [x]. For example, since F-*walks* maps [Lee] to [Lee walks], then the lifted category (CAT) of *Lee* is such that F:CAT applied to [Lee] maps [walks] to [Lee walks]. From this, it follows that the lifted category can take an expression of category B/A to its right only if that expression had a category looking for an A to its left, and vice versa. It is thus not a 'bug' that lift happens to preserve order; it follows from the general definition of lift extended to categories. Note that if we have infixation slashes and corresponding circumfix slashes those would also immediately fold into the system: A could lift to $B/_C(B/_IA)$ or to $B/_I(B/_CA)$. Moreover if an infixation point is generally inherited from the function, the lifted category needs to specify that it is inherited from the argument. (This means that categories are more fully articulated in that they also encode infixation point inheritance; see Aus in preparation.)

We now return to the earlier puzzle of why a quantificational determiner has the surprising category $(S/_R(S/_LNP))/_RN$. Recall that most of those directional features are predictable, but the first (in the left-to-right sequence) R-slash is surprising. Since the last argument in in forming an S is usually taken to the left, why would the VP be taken to the

right here? Similarly, in the case of a lexical generalized quantifier like *nobody* why is its category $S/_R(S/_LNP)$? The answer for the case of *nobody* is that this is simply listed with category $Lift_{S/NP}(NP)$. It follows from that that it can be $S/_R(S/_LNP)$. It could also be $S/_L(S/_RNP)$. Similarly, a quantificational determiner such as *every* is listed in the lexicon with category $Lift_{S/NP}(NP)/N$. Incidentally it doesn't matter that there are items listed as being of category $Lift_{S/NP}(NP)$ which don't have identical strings of category NP (i.e., the generalized quantifiers), since the category $Lift_{S/NP}(NP)$ is still perfectly well defined.

2.5.2. Function Composition (or it's Curry'ed version - "Geach")

Steedman (1987, 2023 and many works in between) proposed adding function composition as a possible combinatory rule, and showed how this can account for leftward *'wh'*- extraction cases. As will be detailed in Sec. 2.6 I will actually treat leftward (*wh*-) extraction differently; one new rationale for is given in Sec. 4, and greater detail can be found in Au (in submission). Nonetheless, having function composition (or a variant thereof to be discussed below) has striking payoffs especially with respect to coordination. The general operation of function composition is defined as follows. Given a function f in <a,b> and a function g of type <b,c>, then g $\circ$ f (read "g compose f") is a function in <a,c> and is $\lambda x_a[g(f(x))]$. Initially then we add two general rules to the combinatory apparatus:

(13) a. Given an expression α of the form $<[\alpha], A/_RB, [[\alpha]]>$ and an expression β of the form $<[\beta], B/_RC, [[\beta]]>$, there is an expression of the form $<[\alpha\beta], A/_RC, [[\beta]] \circ [[\alpha]]>$. ("right composition")

b. Given an expression α of the form $\langle[\alpha], A/_L B, [[\alpha]]\rangle$ and an expression β of the form $\langle[\beta], B/_L C, [[\beta]]\rangle$, there is an expression of the form $\langle[\beta\alpha], A/_L C, [[\beta]] \circ [[\alpha]]\rangle$. ("left composition")

With the addition of these two rule schemas, a sentence like *every candidate tells lies* has two possible derivations giving the same word order and the same meaning. The familiar derivation is the one in which *tell* (of category $(S/_L NP)/_R NP$) first combines with the NP *lies*. This yields *tells lies* of category $S/_L NP$ which is taken as argument of *every candidate* of category $S/_R(S/_L NP)$ with the familiar semantics. Alternatively, *every candidate* can function compose with *tell* as the reader can verify to give the $S/_R NP$ *every candidate tells.* When this takes *lies* as argument we get the same meaning as the 'traditional' derivation. Indeed this follows by the very definition of function composition, hence $[[\text{every candidate}]]_o[[\text{tells}]]([[\text{lies}]]) = [[\text{every candidate}]]([[\text{tells}]]([[\text{lies}]]))$.

The same possibilities are there with an ordinary subject like *Lee*. Here the minimal derivation is the ordinary one where *tells* combines with *lies,* and *tells lies* takes as argument the NP *Lee.* But *Lee* can also lift to the generalized quantifier category, in which case *Lee tells lies* has the two derivations exactly analogous to the case of *every candidate tells lies.*[5] In particular there is an analysis by which *Lee tells* is a well-formed expression. (See Jacobson 2014 for an additional way of allowing *Lee tells* to be a well-formed expression which allows *Lee* to directly combine with *tells* - 'swallowing up' the subject slot. This simplifies some of the derivations, but has no effect on the content here.)

One benefit of these 'extra' and unusual constituent structure analyses is that the existence of a variety of cases of so-called non-constituent coordination and/or Right Node

Raising follow effortlessly. Thus a typical RNR example like (14) is simply the coordination of two expressions of category $S/_RNP$ which then ultimately take the NP *model theoretic semantics* as argument:[6]

(14) Lee loves and Sandy hates model theoretic semantics.

In addition, Dowty 1988 showed how lift and function composition together allow for a constituent like *lobster on Tuesdays* in (15), which in turn predicts the existence of surprising so-called non-constituent conjunction cases like (15):

(15) Captain Jack serves scallops on Mondays and lobster on Tuesday.

on Monday is a VP modifier; i.e. $VP/_LVP$. *scallops* is an NP, but can lift over transitive verbs to category $VP/_L(VP/_RNP)$ or - for short $VP/_LTV$. These two combine by the left composition rule in (13b) yielding *scallops on Monday* of category $VP/_LTV$. Thus this (like lifted *scallops* itself) wants a transitive verb to its left to give a VP. *lobster on Tuesday* composes in the same way, the two conjoin, and take the transitive verb *serves* to the left.[7]

The claim that *Lee loves* can be a constituent within *Lee tells lies* is of course often met with shock, and a suspicion that anyone making such a claim must have never taken a beginning syntax course. After all, don't we learn in baby syntax that there is a VP category (the particular name is not relevant) and hence *tells lies* is a constituent in *Lee tells lies?* (Recall that here we are taking 'structure' just as a way to represent how the rules prove something well formed, but no harm will be done by using the standard terminology of 'constituent structure'). Moreover, both *Lee tells* and Dowty type cases like *lobsters on Tuesday* in (11) pass none of the constituent structure tests except coordination. But in fact, this common reaction is unwarranted. First, the usual conclusion that *Lee tells* cannot be

a constituent is based on a hidden assumption: that unambiguous sentences can have only one structure. The arguments showing that *tells lies* can be a constituent in *Lee tells lies* (and that therefore there is a VP category) are just that - they show that the standard view of the 'structure' of this sentence is one possible structure. In no way does it follow from that this this is the only structure; this would follow only from the hidden assumption given above. But not only does that assumption follow from nothing, but actually under any theory it is almost certainly incorrect: it would take a lot of extra work to preclude two different structures for, e.g., *Roses are red and violets are blue and irises ae white.* The two structures correspond to a single meaning (since [[and]] is associative). Second, as to the fact that the 'strange' extra structures pass none of the textbook constituent structure tests except coordination, that is also true for many other well accepted constituents. The italicized expressions in (16) also pass none of the standard tests except coordination:

(16) a. Lee liked the meal *which that restaurant served.*

b. Lee believes that *the earth is round.*

c. Lee believes that Sally *walked to the bus stop.*

Indeed, is well known that the familiar battery of tests are all one-way tests: to pass is to be a constituent, to fail shows nothing. Much more than constituency is needed for each test. For detailed discussion see Jacobson 2014, Jacobson 2023, and Steedman 2023. The only phenomenon which seems to require only constituency is coordination, and this follows from the category of *and* as (X/X)/X, for X a variable over categories, where X can be instantiated by, e.g., $S/_{R}NP$.[8]

Let us return now to the two function composition schemas in (13). Restricting ourselves to categories making use only of R and L slashes (thus leaving aside the possibility of infixation) the main claims of this paper are that (a) these are the only two types of function composition and (b) that this follows from the view of syntactic categories advocated here. The bulk of Secs. 4 and 5 will justify the claim in (a) where we show that some striking empirical predictions follow from this. Elaborating on this, we claim that when an expression of category $A/_RB$ composes with one of $B/_RC$ the result is $A/_RC$ and not $A/_LC$ and that this is no accident. Moreover - unlike the view in Steedman 2023 and other works - we also claim that there is no composition such that an expression of category $A/_RB$ can take to the right an expression of category $B/_LC$ to give $A/_LB$ (and where the semantics is function composition). We refer to this as S-crossing composition. Steedman also allows the mirror imagine, whereby an $A/_LB$ can take to its left a $B/_RC$ to give an $A/_RC$ (also here called S-crossing composition). A major claim in this paper is that there is no S-crossing composition. Before we return to the question of why that should be so, we can see an immediate benefit to disallowing this under the assumptions here. Thus allowing for S-crossing composition would allow *believe* (of category $VP/_RS$) to take *lost* (category $S/_LNP$) to its right to give *believe lost* of category $VP/_LNP$. That then would give a well-formed VP *Joe believed lost* and thus ultimately sentences like **Lee Joe believed lost* (meaning "Lee believed that Joe lost"). Incidentally, this is not to say that it is inconceivable that *believed lost* can combine - as is well know, this is found, for example, in leftward *wh*- type extraction cases (we return in Sec. 2.6). But while this may be a well-formed expression, we don't (under the system here) want this to have the

category VP/$_L$NP which is what would be allowed if S-crossing composition was available.[9] If we are correct that this type of "composition" is precluded why is that? Put differently, why are the only instances of syntactic function composition (as demonstrated by coordination possibilities) just those that preserve word order?

Once again, the answer lies in the view of syntactic categories assumed here. If categories correspond to actual functions in <string,<string,string>> and 'function composition' as literally the composition of two functions (resulting after the function-category is applied to the string in question), then the schemas in (13) are the only two possibilities using only R and L slashes. Note that if there are I and C slashes then indeed some forms of crossing composition might be possible (see Aus in preparation). After all, there is a function F'-*believe*; which is a function from strings of category S to strings of category VP. And there is a function which is the composite of that with F'-*lost.*. Whether that composition is allowed in the grammar depends on how infixation point inheritance is encoded in the categories (see Aus in preparation). But in any case that composition would not yield VP/$_L$NP; it would have to be VP/$_I$NP (where *believes lost* would take *Joe* as infix to give *believes Joe lost.* Anything else is, by definition, not function composition.

With this, we reformulate (13). Recall that for any α of category X, F'-(α) is that function which results F:X is applied to [α], hence (13a and b) can be recast as (17):

(17) Given an expression α of the form <[α], A/B, [[α]]> and an expression β of the form <[β], B/C, [[β]]>, there is an expression γ of the form <F'-α ∘ F'- β ([α])([β]), CAT of γ is such that F'-(γ) = F'-(α) ∘ F'(β), [[α]] ∘ [[β]]>.

Since S-crossing composition has been made crucial use of in Steedman 2023 and other works, we should of course show how the phenomena he uses this for have other accounts. Space obviously precludes exhaustive discussion, but we mention one intriguing example: Steedman shows that Dutch and Swiss German cross-serial dependencies can be handled by his 'second order' S-Crossing composition (e.g.. Steedman 2023). But Pollard 1984 showed that these can also be handled using a Wrap operation, so cross serial dependencies can be accommodated without S-crossing composition[10].

2.5.3. Recasting Function Composition as the "Geach" rule

Function Composition is a binary operator: it takes two functions as input and yields a third as output. But any binary operator can be mapped by "Curry'ing" to a function taking one argument at a time. The Curry'ed version of function composition is often referred to in the CG literature as the "Geach" rule (or in some literature "Division"), which we will notate as **g**. This rule thus takes a function in <a,b> and maps it to a function in <<c,a>,<c,b>> such that for any function h in <a,b>, **g**(h) = $\lambda X_{<c,a>}[\lambda C_c[h(X(C)]]$. The reader can verify that for any h of type <a,b> and any f of type <b,c>, **g**(h)(f) = h $\circ$ f. (Note that **g** is not a single operation on functions because it depends what is the type of the newly introduced argument slot, so technically the case above should be notated as $\mathbf{g}_c$, see Jacobson 2014 for details.) Jacobson 2014 provides evidence for using this rather than function composition; the main difference is that using the **g** introduces an additional step in the composition where the result of the **g** might input other things. Another potential advantage of **g** plus application is found in Sec. 5.1 with respect to (43).

Such a modification, however, does not change any of the remarks in this paper. If we recast function composition as **g** (plus application), then we see that the only two ways to apply this to syntactic categories using only R and L slashes are to map an $A/_RB$ to an $(A/_RC)/_R(B/_RC)$ and an $A/_LB$ to an $(A/_LC)/_L(B/_LC)$ - these are the only applications of a rule which we call **g** such that for any two functions h and f, **g**(h)(f) = g $\circ$ f. We will call the mapping of $A/_RB$ to $(A/_RC)/_R(B/_RC)$ the "Geach equivalent" of right composition, and similarly for the left version. Of particular relevance, we do not allow, e.g., an $A/_RB$ to map by **g** to $(A/_LC)/_R(B/_LC)$, for that would be the Geach equivalent of one type of S-crossing composition. Again the view here claims that this is ruled out not by stipulation, but by the definition of **g** combined with the notion that a category corresponds to a function which - once applied to a string of that category - is a function in <string,string>. Similarly, there is no mapping of $A/_LB$ to $(A/_RC)/_L(B/_RC)$. Of course if we have I and C slashes there will be additional instances of **g**, but none of relevance here.

While I believe there is evidence for breaking function composition down into the two steps of **g** plus application most of the rest of this paper continues to use function composition to save a step in the exposition. Exactly the same points hold under the view this a shorthand for two steps, so the reader should keep in mind a when we say 'there is no S-crossing composition' that also means there is no **g** equivalent to this.

2.6. "Wh" (leftward) Extraction

In the CG and related literature there has been a certain amount of debate as to whether the ordinary 'slash' of CG (indicating an argument slot) is to also be used for the kinds of "leftward extractions" that go under the rubric of *wh* extraction (by which I include

Topicalization, even though it contains no overt *wh*). Gazdar et al 1985 and earlier Gazdar 1979 clearly hinted at the similarity by naming the extraction feature that was used in GPSG as 'slash'. But those proposals were not embedded in a Categorial Grammar, and so the two were treated differently, and there was no claim that an argument slot 'missing' in a *wh*-type extraction construction had the same status as an expected argument in CG.

However, Steedman 1987 famously united them within CG by simply having many items select (to their right) something with a right slash. For example, a question word like *who* can have as its category $Q/_R(S/_RNP)$ - it takes to its right an $S/_RNP$ to yield a question, where the expression of category $S/_RNP$ is composed in exactly the same way that we find in, e.g., RNR cases. This idea is certainly appealing: all other things being equal, the Steedman approach is obviously simpler than having the argument structure slash and the 'extraction slash' be different. That said, all other things are not equal. It has often been noted that there are differences between RNR (which in the framework here is a matter of conjoining two things with a right slash) and *wh* extraction (see, e.g., McCawley 1982, McCloskey 1986). One well known example is that extraction is possible in cases like *believe __ was guilty,* but neither Heavy NP Shift nor RNR are allowed:[11]

(18) a. Who do you believe __ was guilty of cheating?

b. *I believe was guilty of cheating the student who sat in the front row.

c. *I believe was guilty of cheating and know should be reported to the dean the student who sat in the front row.

Note that we have already shown that *believe was guilty of cheating* cannot be put together under the view of function composition here to give a $VP/_{L}NP$; by the same logic it cannot be put together to give $VP/_{R}NP$, so the badness of (b) and (c) are expected.

Hence I follow Oehrle 1990 in separating these within a CG framework. I use the notation | to indicate an 'extraction' gap; one can think of this as akin to the superscript notation given in Jacobson 1999 which contains the information that an expression contains within it a pronoun (unbound within it). For the purposes of this paper we will not think of either | or Jacobson's pronoun superscript as actual features. If thought of as that way, they would have a slightly different behavior as they would always be encoded on result categories. I actually see no harm with doing that, but it is simply more convenient here to not make that move. (That the extraction 'gap' | and the CG slash are separate was also suggested in Jacobson 1999 fn. 19, and she actually also proposed | was always encoded on result categories for reasons quite independent of those given here.) Thus we simply expand the recursive definition of categories: for A and B both categories, there are categories of the form A/B (with R, L and possibly I and C on the 'slashes'), a category of the form A|B and a category of the form A^{B}. The semantic type of all of these is <b,a> but the syntax is different. An expression of category A/B expects to find an actual B to combine with (but of course it might not; it might instead function compose (or map into something more complex by **g**) and it might itself be argument of some other function. (In CG in general there is no analog to anything like a Projection Principle or a Theta criterion - nothing requires functions - either in the syntax or the semantics - to be 'saturated'.) An expression of category A^{B} contains (usually) within it an unbound proform of category B

(and again denotes some function in <b,a>) and A|B simply indicates an A-like expression but one 'missing' a B within it.

In terms of how the | is 'percolated', we adopt here a fairly simple set of conventions. First, let any A/B to map to A|B; this is what Gazdar et al. 1985 called the 'bottom' of an unbounded dependency construction, and allows an expected argument to instead be encoded as a gap. Assume additionally that for every A/B there is also a variant (A|C)/(B|C). This is how the information that a daughter has an 'extraction' gap is passed to the mother. The mapping of A/B to A|B has no effect on the semantics (since both are of semantic type <b,a>), while the semantics of the second mapping above is the "Geach" rule (see Sec. 2.5.3). All this is undoubtedly too general: it allows *wh*-extraction gaps virtually anywhere. It provides no account of typical island effects (but see fn. 11), and it does not build in a Left Branch condition. (That would be easy enough to do, restrict the | 'passing' rule above to $(A|C)/_{R}(B|C$. But we do not do that here since the examples below regarding "Lakoff chains" show that gaps can be on the left branch only, suggesting that Left Branch condition effects cannot be attributed to a blanket constraint against gaps on a left branch.) It also provides no account of *that-trace* effects, etc. But we leave those matters here as this is not the main focus of the account, and for now we content ourselves with what is surely an overly permissive account of *wh* extraction One other point to note is that ATB extractions are - just as was discussed in Gazdar et al 1985 and in various CG works - unsurprising as they just involve the coordination of likes: the lexical specification of *and* is (X/X)/X, and X throughout can be instantiated by a category of the form A|B.

3. Coordinate Structure Constraint, Comma chains, and Lakoff chains

3.1. The coordinate structure constraint

Since Ross 1967 it is well known that that extraction from just one conjunct (rather than both) often gives rise to bad outputs. The debate on how to account for this has a long history, but one can identify two major strategies. One posits something extra in the syntax to account for these; the other assumes that these are not about the syntax at all but rather about information structure (Lakoff 1986, Kuno 1987, Kehler 1996 etc.). We will not attempt to propose or endorse any particular full information structure account here, but will be providing new evidence that CSC effects should not come from a principle hardwired into the syntax given how easy it is (with the right construction) to come up with counterexamples. The strategy for constructing the kinds of counterexamples in this paper was first taken in Lakoff 1986 and involve violations of the CSC in comma chains. These provide a fertile ground for many extensions of his observations as will be shown below.

Before turning to these, it is worth pointing out why it is that CSC effects are not automatic in theories like G/HPSG and varieties of CG. One might think that they are - given the presumed "fact" that only like categories conjoin, combined with the idea that a category like A|B or A/B is not the same as the category A. Indeed, that was an early hope in, e.g., Gazdar 1981, but had been abandoned already by Gazdar et al. 1985. The reason that these effects do not follow from the idea that 'only like categories conjoin' is that it is a mistake to think that 'only like categories conjoin' is itself something hardwired into the grammar. In fact, without an extra principle, it is just (often) a consequence of the lexical specification of conjunctions (i.e., in CG (X/X)/X and similar ideas in other theories),

and/or the phrase structure rule schemas (or modern equivalent) allowing any rules like X --> X Conj X or two binary rules to do the same thing). But the lexical categories in CG and/or the phrase structure rules (or their modern equivalent) in, e.g., GPSG, HPSG, and movement theories are exactly what *in general* can be overridden by 'slash passing' conventions, function composition, and/or movement. Indeed, that is the very rationale for these devices! We find verbs like *think* which expect to combine with a full S but where we find that their complement (on the surface) is missing something, so we posit "slash', function composition, or movement. Why should *and* be any different? Given that, a principle to the effect 'only likes can conjoin' would have to be something extra - some sort of output filter. This was already recognized by Gazdar et al 1985, who instead accounted for CSC in a different way. In Steedman 2023 the CSC effect is built in by via a new kind of slash which resists composition, and using that for both the right and left argument slashes of conjunctions. But that too is extra.[12] So if we were to discover that the syntax does not have to be responsible for CSC effects where they occur, that should be a welcome result.

Thus there is a large body of literature spanning decades speculating that CSC effects are about information structure rather than an actual syntactic violation, along with a large body of counterexamples to claim that the CSC is hardwired into the syntax (regardless of the particular theory). A typical and often cited one is (19), (20) is along the lines of examples from Goldsmith 1985 (who defends the CSC):

(19) What did Lee go to the store and buy __?

(20) How much beer can Lee drink __ and still stay sober?

Though less discussed, one can also construct counterexamples with *or*:

(21) This is the car that Lee will (either) buy __ or else he'll just keep taking the bus.

One reaction to these counterexamples is there is a different *and* in these examples (and hence presumably also a different *or* in (21)) (see Goldsmith 1985 for a restructuring analysis). For a case like (19) it is sometimes thought that this a 'subordinating' *and* rather than a coordinating *and*; the hope would be to extend that to (20) and (21). But it's not clear what it would mean for there to be a 'subordinating' *and* distinct from the normal *and* since under many modern views of the structure of *and* - including that here - it is subordinating (any theory with only binary rules has that as a consequence). And it's not clear what would be the meaning of the new *and*. Sometimes it is compatible with an *and then* reading (so-called narrative' *and)*, but that can't be right for (20) where the two 'events' are contemporaneous. Note that these need not involve just VP conjunction; (22) - (24) are S conjunction even with different subjects, nor does there need to be a coreferential pronoun in the second conjunct as long as the connection between the two is clear:

(22) How many banks can Clyde rob __ and the police still not arrest him?

(23) How many banks can Clyde rob __ and Bonnie still pretend that all is fine?

(24) This is the castle that Charles will (either) decide to keep __ or (else) William and Kate will just have to give up their dreams of living in a castle.

And what about the new *or* needed in (21) and (24)? It seems to have the same meaning as ordinary *or.* Again there is a lengthy literature on this (with respect to *and)* and I have nothing new to offer on just what is the needed information structure; see e.g. Lakoff 1986, Kehler 1996, Altshuler and Truswell, 2022 among many others. Incidentally, none of the

CSC violating examples here (and others in this paper) reduce to well known phenonemon of 'coordination of unlikes' as in *She is a linguist and proud of it.* In those standard cases, two expressions of different categories can be conjoined only in environments where each category itself is licensed. This is not the case for these CSC violations. (See fn. 2.)

3.2. Lakoff Chains

What I take to be especially suggestive evidence against the claim that there are different items *and* and *or* (crucially with a different structure) is that we find the same CSC-violating possibilities in comma chains, which involve null *and* and null *or.* This was pointed out in Lakoff 1986 (for the *and* case; he did not look at *or*). His own argument using these examples against a syntactic CSC was different than the point to be made here; I nonetheless refer to these as 'Lakoff chains'. Hence I define a *Lakoff chain* as any comma chain in which some of the conjuncts contain 'missing material' and some don't (including the case in Sec. 5 where some are missing material on the left which others are not). Thus for the case of *and,* Lakoff gives examples along the lines of those in (21) (I indicate which expressions have gaps and which don't by use of __, with no intended theoretical significance). (22) shows the same point with *or*:

(25) a. How much/What kind of beer did Lee go to the store, buy __, pack up __, and then bicycle home?

b. How much beer will Lee buy __, drink__ and still not fall asleep?

c. How much/What kind of beer did Lee buy __, bicycle home, and then instantly proceed to drink __?

(26) This is the car that Lee will (either) go ahead and buy __, decide he doesn't actually have enough money to spend on silly things, or convince his father to give him __.

I put *either* in (26) to make it more natural and easy to process, as it signals right away that the chain is an *or* chain, but that is not strictly speaking necessary.

Consider the implications of the claim that the above all involve a different *and* or *or* whose syntax is such that they escape CSC violations. Presumably they would also have different meanings from ordinary *and* and *or*, but it is difficult to imagine what each would mean (ordinary [[and]] and [[or]] are sufficient to get the right semantics for (25) and (26)). Even more problematic is the observations earlier about comma chains: the last member must be overt and it determines the interpretation of all the rest of the chain (all [[and]] or all [[or]]). This holds equally well in the Lakoff variants of these; (26) for example is interpreted as three disjuncts. (For (26) one might argue that this comes from the pragmatics, but the same holds for *This is the car that Lee will rent __for a month, buy __, or decide to just learn to bicycle.* The first two alone can either have an 'and' interpretation or an 'or' interpretation, but the last disjunct forces 'or' throughout.). In the next section we develop a set of conventions to ensure this in comma chains. But under almost analysis of comma chains, if Lakoff chains involved different conjunctions with different syntax we would need a new set of conventions for their behavior in comma chains.

Interestingly there seems to be one additional item pronounced *and* which is the 'sum' *and* (Link 1983). It takes two individuals (atoms or plurals) as input and returns a Linkian plural; since those are individuals further sum conjunction is possible. (Recall that the ordinary Boolean *and* never conjoins NPs because meet is not defined for individuals;

cases of apparent NP conjunction are conjunction of GQ; sum *and* however conjoins individuals.) Sum *and* also has a null variant, as we find comma chains like *Groucho, Harpo, Chico and Zeppo made a great comedy team*. And, not surprisingly, if the rightmost conjunct has *and,* the chain has to have either the Boolean *and* or the sum reading throughout. Thus consider the following scenario (inspired by a referee's comment). Take a ride sharing coop where one accumulates hours by driving seniors to medical appointments, etc. Once one has accumulated 100 hours, one is entitled to the same services when needed. Suppose that the hours can be accumulated both by individuals or by teams, with the number of benefit hours being doled out to the individual or to the team (for the team to decide how to split them). Let Lee and Sandy be a team who together have accumulated 100 hours, and let Jessie be an individual who also has. I can report (with the right prosody) *Lee and Sandy and Jessie have reached the goal.* But I cannot report this as *Lee, Sandy, and Jessie have reached the goal* where the overt *and* is Boolean *and* (both the team and the individual met the goal) but where *Lee, Sandy* is sum *and.* The truth conditions here require each of Lee and Sandy to have put in 100 hours (alternatively this has a reading which is a sum throughout if all three had constituted a team).[13]

Strikingly, sum *and* also participates in Lakoff chains:

(27) This is the student who the mother of __, the father of __, and the principal of North High became an ad hoc committee to investigate the students' violent drawings.

None of this is surprising. Other than the fact that sum *and* conjoins NPs and Boolean *and* conjoins other categories, their basic syntax appears to be the same. But if the CSC violations found in Lakoff chains were because of some other syntax then there would be

three new conjunctions looking just like Boolean *and, or,* and sum *and* all three of which would have a new syntax allowing for CSC violations, and all of which would necessitate a new account of comma chains with these new items. Henceforth, then, we assume that the CSC effects are not syntactic, and turn now to an account of comma chains in order to return to their interaction with 'extraction' and - more centrally - with function composition.

3.3. Comma chains, Lakoff chains and *wh* extraction

The simplest account of comma chains would simply posit that there are silent lexical items *and* and *or* (and the sum *and)* which have all the same properties as the overt ones. Indeed, this is the account mentioned at the outset of this paper. These items would exactly the same meaning and word order possibility as overt *and* and *or* - just a different phonology, and so I write them as ~~*and*~~ and ~~*or*~~ (henceforth I ignore sum ~~*and*~~ as all the same remarks hold). But this alone does not account for the constraints detailed in the introduction: in a comma chain, the last conjunct must be overt and there is consistent semantics throughout the chain which is determined by the overt one.

Given that there cannot simply be silent conjunctions with the exact same behavior as overt ones, one account of comma chains assigns them a flat structure. But this necessitates a new type of rule making use of the Kleene star. For example, one might propose a phrase structure rule (or its equivalent in modern terms) of the form *X* --> *X* and/or X*, paired with an appropriate semantics; a related proposal is in Gazdar et al. 1985) I set this aside for now but return in Appendix 1, where I argue against this solution. (Some accounts - including Gazdar et al. 1985 - also allow for this with overt conjunctions.)

I will, then, assume that there are silent items ~~*and*~~ and ~~*or*~~ (and sum ~~*and*~~) with a similar syntax to their overt counterparts (and the same semantics), but I adopt a feature passing solution to account for their basic distribution. To this end we adopt two features [&] and [v] (and presumably a third which is [⊕] for sum *and)* and refine the categories for overt *and* such that the category for *and* is (X[&]/X)/X.[14] This allows the overt items to 'launch' the chain (it will be the lowest conjunct) and the relevant information can be passed up. *Or* is the same but with the [v] feature instead. As to the role of these features on the case of null conjunction, we simply posit that the category of ~~*and*~~ is not exactly that of *and.* The full lexical entry for ~~*and*~~ (once directional slashes have been added) is: <[∅], $(X[\&]/_L X)_R/X[\&]>$. This ensures that ~~*and*~~ combines only with something as its right argument that has an overt *and* 'at the bottom', and it also passes up that feature just as does ordinary *and.* All of the same holds for ~~*or*~~. Note that these conventions assure a right branching structure (in an application only derivation) throughout for comma chains. (Henceforth reference to a right branching derivation means the application derivation.)

Moreover, recall the earlier discussion about features: if X itself is complex the relevant feature will be on the result; see below for two striking benefits. So, for example, take the case of two conjoined VPs like *danced and sang.* The whole VP needs to have the [&] feature because it could conjoin on its left with ~~*and*~~ (giving, ultimately, *chewed gum, danced and sang).* But the [&] feature doesn't just span "VP", rather it is on the result; the category of *danced and sang* is actually S[&]/NP.[15] And this in turn means that the [&] feature is propagated up, so that an ordinary sentence like *Lee danced and sang* is S[&]. (Note that the feature goes no higher; it stops propagating up once we have an expression

of the category of the result). While it may initially seem counterintuitive to have the feature above where it appears to be needed, we document below two benefits. I see no harm in this provided it is understood that any item of the form A/B can take B[&] as argument. Thus if *that* is CP/$_R$S it can take an S[&] - which might be an S with, for example, VP conjunction within it. (Note that [&] will not be on the CP.)

The Lakoff chain examples above - all involving *wh* extraction cases - are unsurprising given the conventions above for the passing of the | feature. The category for *and* is (X/X)/X. Suppose X is instantiated as, for example, VP. Then the above conventions allow a version of *and* which is ((VP/$_L$VP)|NP/$_R$(VP|NP). The lexical entry for *drink* is VP/NP but it can shift (with no meaning change) to VP|NP. Thus *and* can take *drink* (here, to its right, to give give *and drink* of category ((VP/$_L$VP)|NP. *bicycle home* is an ordinary VP, but can lift to VP/$_R$(VP/$_L$VP). That can shift by "|- passing" to give (VP|NP)/$_R$((VP/$_L$VP)|NP). Then *bicycle home and drink* is a VP|NP. (There actually is a much simpler derivation using the recursive definitions in Jacobson 2014; the interested reader can consult that.) Incidentally, these chains provide evidence not only against a syntactic Coordinate Structure Constraint,[16] but also against at least a completely general Left Branch Condition. Obviously some principle is needed for some such effects, but if it is too general it will incorrectly rule out some of the examples above. Even simple cases like (25a,b) involve a gap in a left conjunct only.

Especially interesting for our purposes is to note that the Lakoff chains in these *wh*-extraction cases allow for what I call 'mix and match'. In any two coordinated constituents, the gap can be on the left and not on the right as in (25a), (*pack up __ and then bicycle*

home), on the right and not on the left (*bicycle home and then proceed to drink*__ as in (25c)), or on both (as in ordinary ATB cases). This is exactly to be expected given the | passing conventions adopted above. Moreover, the lowest one can contain a gap or not, as can the highest one. When we turn to right composition (the RNR examples) and then to left composition (the "MM" examples) we see that these do not show the same freedom.

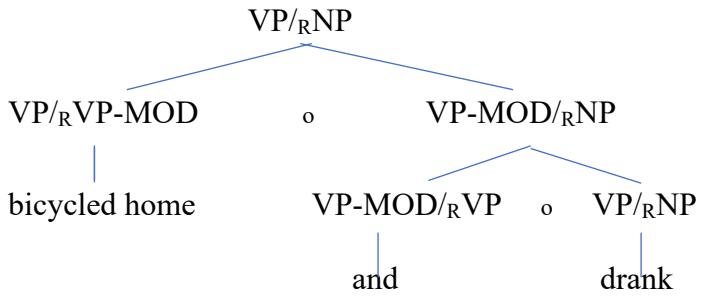
4. Rightward extraction and Lakoff chains: the lesson for function composition

Consider now Lakoff chains in the case of rightward 'extraction' – i.e., cases thought of as RNR and Heavy N Shift. Here we also find that 'mix and match' is generally allowed. But unlike *wh*-extraction an interesting constraint emerges: the rightmost member of the chain has to contain a gap. Before showing why, consider first a good case like (28):

(28) Lee went to the store, bought __, bicycled home, and drank __ 2 sixpacks of beer.

The bottom of the chain in (28) is *bicycled home and drank* __, where we have a full VP on the right conjoining with one with a 'gap' - or, put differently, with the ordinary transitive verb *drank* which is of category $VP/_{R}NP$. The expression *bicycled home and drank* can combine by function composition. For expository convenience, we use the abbreviation VP-MOD for $VP/_{L}VP$ (note that this is the category of, e.g., *and chews gum*). Note moreover that an ordinary VP like *bicycled home* can lift to $VP/_{R}(VP/_{L}VP)$ which we also abbreviate as $VP/_{R}$VP-MOD. In the illustration below of the derivation we begin with the lifted category of *bicycled home* to save space, and so the lowest VP here is analyzed as follows (we suppress the & feature):

(29)



Given the derivation above of *bicycled home and drank,* two things are possible as we compose bigger expressions. This expression can combine with a VP on its left in exactly the same way - using either overt *and* or, in the cases of interest here, ~~*and*~~. This for example is what happens in (30), where *bicycled home and drank* __ combines with *found his bicycle* in the same way as happens at the bottom of the chain as shown in (30).

(30) Lee bought __, found his bicycle, bicycled home, and drank __ 2 sixpacks of beer.

Alternatively, *bicycled home and drank* __ can combine with another VP/RNP by ordinary coordination of likes; this is exemplified in (28) where *bicycled home and drank* __ conjoins (by ~~*and*~~) with *bought* __. (This also happens in (30) at the top of the chain.). So, once the chain is launched, we can keep mixing expressions with a gap with ones without.

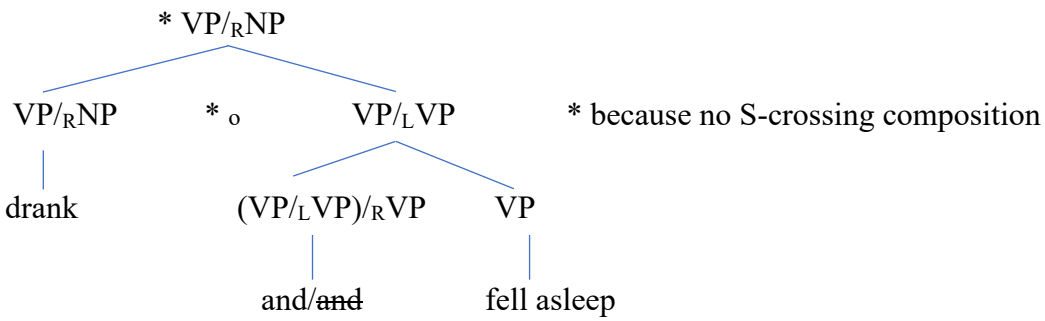
We have dealt here only with combinations of VP/RNP (transitive verbs, whether simple or complex) and VP, but we should also find this with S/RNP and S. And we do:

(31) George loved __, Paul hated ___, Ringo was just like 'whatever', and John insisted on performing __ that silly song about a yellow submarine.

While these look so far like the situation with *wh* extraction, the account here makes a striking prediction. We see above that a VP on the left can combine with a VP/RNP on the right (with *and* or ~~*and*~~ as intermediary) and the result is VP/RNP. But there is no way for an 'ordinary' VP on the right to combine with a VP/RNP to the left so as to give

$VP/_RNP$ - the category needed in RNR (and "Heavy NP Shift"). For this would not be function composition - it would have to be S-crossing composition. For example, take *drank* which is of category $VP/_RNP$ and *fell asleep* of category VP. *and/~~and~~* can of course take the VP on its right to give $VP/_LVP$. But there is no way for *and/~~and~~ fell asleep* to take a $VP/_RNP$ to its left to give $VP/_RNP$); my use of * indicates an impossible composition:

(32)



Notice that S-Crossing composition gives exactly the above. The $VP/_LVP$ *~~and~~/and fell asleep* would take to its left the $VP/_RNP$ *drank* yielding a $VP/_RNP$. But we stress that it is not true function composition and so is precluded under the view of categories here.[17]

Is this prediction correct? Indeed it is; compare (33a,b) to (33c) and (31) to (34):

(33) a. *Lee drank __ and fell asleep 2 sixpacks of beer.

b. *Lee went to the store, bought __, drank __ and fell asleep 2 sixpacks of beer.

c. Lee went to the store, bought __, bicycled home and drank __ 2 sixpacks of beer.

(34) *Paul loved __, George hated ___, John insisted on performing __ and Ringo was just like 'whatever' that silly song about a yellow submarine.

Thus while RNR cases do allow 'mix and match' in a Lakoff chain once it is launched, it has to be launched by having the 'smaller' item (the one of category $A/_RB$)

being the rightmost one. This contrasts with the case of leftward *wh* type extraction. By way of a complete minimal pair, compare (25b) to (35):

(35) *Lee will buy__, drink__, and still not fall asleep 2 sixpacks of beer.

Hence this phenomenon also provides new evidence that the two require different conventions (beyond those already noted in past literature, see e.g., McCloskey 1986).

5. Leftward function composition chains

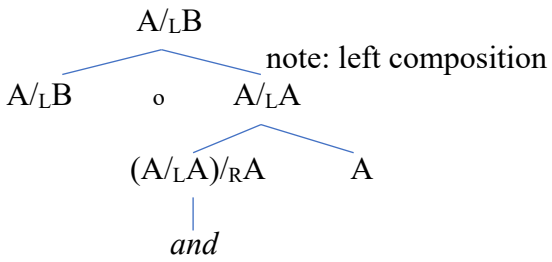
5.1. The predicted and actual existence of left looking chains

As noted above, a chain with a rightmost 'filler' is launched by something of the form *A and A/$_R$B,* thus we find (among many others) chains of the general form:

[A ~~and~~ [A/$_R$B and A/$_R$B]] B. The mirror image - in terms simply of the string order - would be B [A/$_L$B [A/$_L$B and A]] (The bracketing is not mirror image due to the structure of *and* and ~~*and*~~). This raises two questions. Under the view here, would we expect to find such cases? And do we find them? The answer to both is yes.

Turning first to the question of whether we should expect these, we can show that a left-looking Lakoff chain can be launched as shown below:

(36)



This can then conjoin with another A/$_L$B by the ordinary coordination of likes, where that conjunction is either via an overt *and* or an instance of ~~*and*~~; the latter gives rise to a Lakoff chain. Once that is done the string order will be exactly what is noted above: B [A/$_L$B

[~~and~~ A/$_L$B and A]] - or, to instatiate this with actual categories: NP [S/$_L$NP [~~and~~ S/$_L$NP and S]]. Another instantiation would be TV [VP/$_L$TV [~~and~~ [VP/$_L$TV and VP]] (recall that VP/$_L$TV is the category of an NP lifted over transitive verbs; its full category is (S/$_L$NP)/$_L$((S/$_L$NP)/$_R$NP) where this is a valid lift of an NP). And recall that (leaving aside sum *and)* the NPs would have to lift as Boolean [[and]] is not defined for individuals.

The next question is: do we get these? The answer is yes - these are exactly the Maxwell/Manning cases. Simplified versions are (37) and (38);

(37) Barry [$_{S/LNP}$ plays the dulcimer ~~and~~ [$_{S/LNP}$ does magic tricks [and [$_S$ Patty plays the flute].

(38) Barry [$_{TV}$ plays [$_{VP/LTV}$ the dulcimer ~~and~~ [$_{VP/LTV}$ the standup bass [and $_{VP}$ does magic tricks.].

Because the syntax and the semantics track each other and involve parallel combinatorics, the semantics of these and all subsequent examples work out as expected. By way of illustration take (37). The composition of *does magic tricks and Patty plays the crumhorn* involves function composing the first VP with the final S. The *and* here is propositional *and* (here we need to give the semantics intensionally as $\prod$ is defined for two arguments of type <s,t>). Hence *and Patty plays the flute* is $\lambda p_{<s,t>}$[p $\prod$ patty-play-flute']. This composes with *does magic tricks* - of type <e,t>, to give λx_e[does-magic-tricks'(x) $\prod$ patty-play-flute']. This conjoins with *plays the hammered dulcimer* (in two steps since ~~*and*~~ is Curry'ed) by the ordinary conjunction of likes. Just as would be the case for any conjunction of two expressions of type <e,t>, the resulting semantics for *plays the hammered dulcimer, does magic tricks, and Patty plays the flute* is λx_e[plays-the-

dulcimer'(x) ∏ does-magic-tricks'(x) ∏ patty-play-flute']. This function is ultimately applied to [[Barry]]. (One might be tempted to think that these are instead involve ellipsis; see Appendix 2 for challenges to an ellipsis approach.)

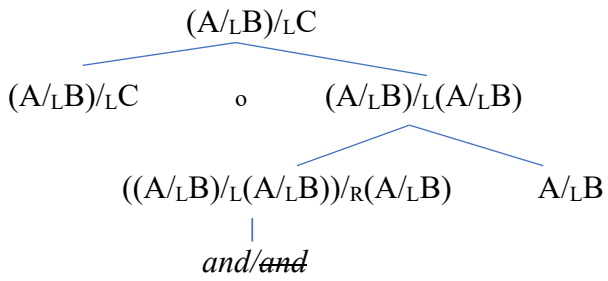
The semantics of (38) is also straightforward. *The standup bass* lifts from its initial e-type meaning to a meaning of type <<eet>,<et>>; in particular its meaning is $\lambda R_{<e,<e,t>>}$[R(the-standup-bass')]. Given the generalized semantics for *and* in Partee and Rooth 1983, [[and does magic tricks]] is $\lambda P_{<e,t>}$[P ∏ does magic tricks], which is of type <et,et>. But this combines not with an <et> meaning but with something of type <eet,et> via function composition, yielding $\lambda R_{<eet>}$[R(the hammered dulcimer) ∏ does magic tricks]. The same thing happens further up such that the meaning of *the hammered dulcimer, the standup bass, and does magic tricks* is (39):

(39) $\lambda R_{<eet>}$ [R(the standup bass) ∏ R(the hammered dulcimer)] ∏ does magic tricks

[[plays]] eventually fills in the R slot, and we get the intersection of playing stand up bass, playing dulcimer and doing magic tricks, exactly as we want. We will henceforth omit the semantic composition for each case as the semantics is predictable from the syntax.

We also predict that there are more complex Lakoff chains. Note that an expression of any category $A/_LB$ itself could combine with something smaller to its left such as $(A/_LB)/_LC$ in exactly the same way that was illustrated in (36):

(40):



To put this all together, take the tree in (36) and imagine that it is the expansion of the rightmost node in (40) (the node $A/_LB$). This yields chains like (41), shown first in the abstract and then instantiated with actual categories, and finally actual words:

(41) $[(A/_LB)/_LC$ ~~and~~ $[A/_LB$ and A]]

$[(S/_LNP)/_LTV$ ~~and~~ $[S/_LNP$ and S]]

the dulcimer ~~and~~ does magic tricks and Patty plays the flute

We cannot show that this exists simply by placing the TV *plays* to the left, as that has a perfectly good different parse - the very one in (37) where *plays the dulcimer* can be analyzed as a VP conjoining with *does magic tricks.* Nonetheless we can show that cases like (41) exist by conjoining *the dulcimer* to the left with another raised object by the ordinary conjunction of likes. And indeed this is good:

(42) Barry plays the standup bass, the dulcimer, does magic tricks and Patty plays the flute.

The question was raised earlier as to whether the 'growth' needs to be incremental: do we, e.g., have chains of the form *(lifted) NP, (lifted) NP, and S*? The answer is yes:

(43) Barry plays the hammered dulcimer, the standup bass, and Patty plays the flute.

NP TV $[(S/_LNP)/_LTV$ [~~and~~ [$(S/_LNP)/_LTV$ and S]]].

One simple analysis involves using Geach + application rather than function composition.. Recall that *and S* can compose up to be $S/_LS$. We need then only two applications of **g**. The first maps this to $(S/_LNP)/_L(S/_LNP)$ (a VP modifier). The second maps that to $((S/_LNP)/_LTV)/_L((S/_LNP)/_LTV)$. Hence *and Patty plays the flute* takes *the standup base* as argument. From there we have coordination of likes. There is also an analysis using just

function composition. *the hammered dulcimer* lifts to category $S/_L(S/_RNP)$. Since *and Patty plays the flute* is $S/_LS$, the two can compose to give *the hammered dulcimer and Patty plays the flute* as a well-formed expression of category $S/_L(S/_RNP)$. *the standup bass* lifts in the same way and we have coordination of likes in the expression *the standup bass, the hammered dulcimer and Patty plays the flute.* Recall that *Barry plays* can compose up (with lifted *Barry*) to give an expression of category $S/_RNP$ and that thus is the argument of the complex expression above.

The actual example in Maxwell and Manning 1996 involved not just a simple object but a "fancy" one of the type analyzed in Dowty 1988:

(44) John wanted to study medicine when 11, law when 13, and to study nothing at all when 18.

(Incidentally, their analysis of these has virtually nothing in common with the analysis here nor did they discuss this in any generality.) These examples are fun but really add nothing new. *medicine when 11* can compose in the way shown earlier for (15) to be something wanting a TV to its left to give a VP, so the rest is the same as the case of an ordinary object which lifts over TVs. And unsurprisingly - parallel to the "Barry" sentence in (42) except with 'fancy' objects - we find more complex chains like (45):

(45) Laura teaches syntax in the fall, compositional semantics in the spring, goes to music workshops in the summer, and she spends each sabbatical writing a book.

5.2. *A note on the [v] feature and <u>either</u>*

We turn briefly to some well known but at first glance surprising facts about the distribution of *either*. There is a large literature on this (especially on 'too high' *either*) so

the discussion here cannot do full justice. Nonetheless, it is striking that much of the puzzling distribution of *either* follows immediately here. But since the literature contains a wealth of rich facts which we do not deal with here, we offer the following as preliminary.

First, it has been noted that *either* can appear in a position which seems to be 'too low' (Hendriks, 2004, den Dikken, 2006) as in (46):

(46) Barry will play either the dulcimer or (will) do magic tricks.

We would expect it to scope over the two VPs (as in *will either play the dulcimer or do magic tricks* or *either will play the dulcimer or will do magic tricks)* since the scope of *or* appears to be about VP disjunction. But given function composition this phenomenon is unsurprising. Since *either* needs to find an overt *or* somewhere, assume that it takes an expression with the [v] feature and simply gives back the same expression without that feature, hence its category is X/X[v]. That it swallows' up this feature can be shown as follows. First we should note that some people do not like *either* with more than one disjunct, but many do, and so we should get *either* with comma chains:

(47) He'll either dance, sing, or jump.

But ~~*or*~~ here (between *dance* and *sing or jump)* needs its right sister have the [v] feature. We thus predict that if *either* were in the way it would have swallowed up that feature and not licensed ~~*or*~~ higher up. This is indeed correct as shown by (48):

(48) a. *He'll either dance, either sing or jump.

b. *He'll dance, either sing or jump.

We assume that the semantics of *either* is the identity function though it does strengthen the exclusive implicature of *or;* for reasons left open here.

Hence given the feature passing conventions adopted earlier for [v], nothing more needs to be said to license 'too low *either';* its existence follows directly from function composition. Consider (46). Here *the dulcimer or will do magic tricks* is put together just as we saw for the last two conjuncts in (40) where they function compose, and the *or* feature is inherited on the result by the way function composition works. *or does magic tricks* is of category VP[v]/$_L$VP (we show the [v] feature on the full VP result here but recall it will actually be on the S of the category S[v]/NP). *the hammered dulcimer* can lift over transitive verbs to be of category VP/$_L$TV. When they compose the result is VP[v]/TV. Recall that the category of *either* is X/X[v], but recall further that when we write X[v] that is shorthand for any category where [v] is on the ultimate result. In more detail, *either* takes as argument any expression whose category has [v] on the result, as is the case for *the dulcimer or will do magic tricks.* (Spelling out the full category, it is (S[v]/NP)/((S/NP)/((S/NP)/NP)). The semantics is just like the case of (39) except this uses the 'join' rather than 'meet operator. So in (46) , it is no surprise that *the hammered dulcimer or does magic tricks* can be argument of *either* (which would remove the feature). Or of this could just instead combine (again by function composition) with (lifted) *the standup bass* to give a Lakoff chain, and *either* can be introduced there (or never):

(49) Barry will play either the standup bass, the dulcimer or will do magic tricks.

It has also been noted that *either* can apparently go 'too high', as in

(50) a. Barry will either play the dulcimer or the bass.

b. Either Barry will play the dulcimer or the bass.

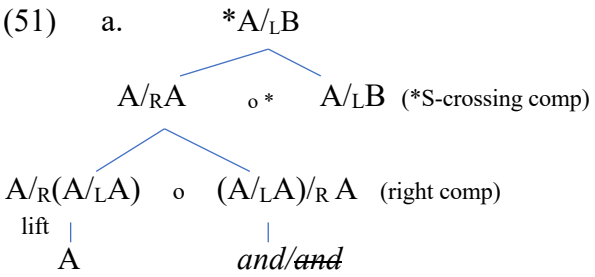
Here the situation is more complex as there are various constraints on 'too high' *either* and the generalizations are not entirely agreed upon in the literature. But the basic picture also follows here. Thus in (50a) *either* is at the VP level but *or* combines two (lifted) NPs. That is also the scope of *or* in (50b), yet *either* shows up above the whole S. This too follows from the analysis of feature passing here. Recall that the most natural way to conceive of features in CG is to have them the result category for any function category. From that it follows that *Barry play the dulcimer or the bass* is of category S[v]. *Either* can then combine with this and 'swallow up' the [v] feature. Similarly it can combine in (50a) with the S[v]/NP (returning S/NP).[18] The conclusion here is that 'too low' *either* is licensed in a fashion exactly analogous to what gives left looking Lakoff chains, and 'too high' *either* is licensed in a fashion exactly to the material discussed below in Sec. 5.4.

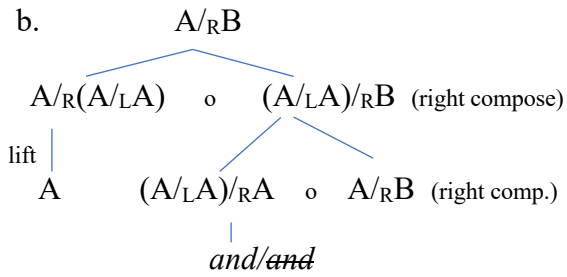
5.3. The "Grow Only Rightward" phenomenon: No S-Crossing Composition

Recall that in the case of a Lakoff chain looking for something to its right, the rightmost one had to be the smallest, but otherwise any kind of mix and match is possible. The left looking case is different. We have seen how larger expressions on the right (lower down) can combine with smaller ones to their left - such cases are all 'launched' by the bottom of a chain being an expression of the form $A/_L B$ *and* A where that expression itself is $A/_L B$. (As discussed above - a chain can also be launched by, e.g., as $(A/_L B)/_L C$ *and* A).) But there is an interesting prediction: such an expression should not be able to combine with something larger to the left ("larger" means a category like A as opposed to A/B, and we take VP to be 'larger' than lifted NPs).

To exposit, let us try to compose an expression of the form *A and/~~and~~ A/$_L$B* - either at the beginning (rightmost, lowest) part of the chain (which would involve overt *and*) or anywhere higher up (using ~~*and*~~) to give an expression looking for something on its left (i.e., an A/$_L$B). (If this can't be done, it follows that we also can't get *A and/~~and~~ (A/$_L$B)$_L$C).*) Consider what it would take for *and* (or ~~*and)*~~ to combine on the right with an expression of category A/$_L$B. First, we could use the version of *and/~~and~~* of category ((A$_L$B)/$_L$(A/$_L$B)/$_R$(A/$_L$B). This is fine but just involves coordination of likes. Let us try instead to use the version of *and/~~and~~* of the simpler category (A/$_L$A)/$_R$A. Can this combine with an A/$_L$B to its right to give (A/$_L$A)/$_L$B? This, for example, would allow strings of the form *S and VP (i.e., S/$_L$NP)* of category S/$_L$NP at the bottom - or as part of - a left looking Lakoff chain. It would also allow a string *VP and VP/$_L$NP* (the latter being the category of lifted objects) where that string has the category VP/$_L$NP? If there were S-Crossing composition the answer is yes. But under the view of categories here the answer is no.

We might still try one more tack, which is as close as possible to the mirror image of what happens in the right looking case. This would be for *and* to combine with its arguments in a left branching fashion. After all that should be allowed by function composition just the same way that we have a left branching derivation of, e.g., *Lee tells lies* (as discussed in Sec. 2.5.2). The problem here is that the next step once again would require S-crossing composition. We can illustrate this, comparing the left looking derivation in (51a) with the right looking one in (51b). The categories shown in bold are the ones we are ultimately trying to conjoin to get result being aimed for, but the category at the top requires S-crossing composition, so (51a) does not exist:

(51) a. 

b. 

There are other possible things one might try given combinations of Lift and other operations, but since Lift is order preserving it won't matter: there is no way to get a larger expression to the left of a smaller one, not only at the beginning of the chain, but anywhere higher up. It is thus predicted that the only way to get leftward Lakoff chains is as shown in Sec.5.1. Things can grow rightward, but they cannot shrink.

Strikingly, this prediction is correct. Compare (52a) to (52b):

(52) a. Sarah teaches Lexical Semantics in the fall, Compositional Semantics in the spring, and plays the violin in her spare time.

b. *Sarah teaches Lexical Semantics in the fall, plays the violin in her spare time, and Compositional Semantics in the spring.

(b) is good only on the pragmatically bizarre reading in which Lee plays Compositional Semantics. Similarly (53b) has only the strange reading where Barry plays treehouses.

(53) a. Barry builds dulcimers, treehouses, and plays the standup bass.

b. *Barry builds dulcimers, plays the standup bass, and treehouses.

5.4. The surprising reading: features on result categories and 'too high' ~~conj~~

The same point should hold in the case of chains of Ss followed by VPs. At first glance, it appears that this generalization does not hold in that case; (54a) is the expected order, but (54b) - where we flip the VP and S - is surprisingly good:

(54) a. Sally teaches semantics, studies math, and she plays the crumhorn.

b. Sally teaches semantics, she plays the crumhorn, and studies math.

But on closer look, this turns out not to be a counterexample. In fact, it involves a different and perhaps surprising analysis - the very same analysis which allows for the pragmatically bizarre readings of (52b) and (53b). This is that *plays the crumhorn* and *studies astronomy* are conjoined VPs which take *she* as subject to give an S. Thus the structure is (55b) rather than (55a) which would require S-crossing composition:

(55) a. Sally teaches semantics ~~and~~ [[$_S$ she plays crumhorn] and [$_{VP}$ studies math]].

b. Sally teaches semantics ~~and~~ [$_S$ she [$_{VP}$ plays the crumhorn and studies math]].

It is easy to show that (55b) is the right structure simply by using a different subject:

(56) Sally teaches semantics, Lee plays the crumhorn and studies math.

We cannot understand this as saying that Sally studies astronomy; it is Lee who studies astronomy. That is what we would expect from the parse in (55b), whereas under (55a) *Sally* is the subject of the lower VP, and the separate sentence *Lee plays the crumhorn* would have nothing to do with the interpretation of the astronomy-VP. Thus the Grow Only Rightward prediction is born out, which in turn supports the hypothesis of no S-crossing composition, as we would expect under the view of syntactic categories here.

But (55b) at first glance brings up a new mystery: shouldn't it also be bad? Isn't the overt *and* too low to support ~~*and*~~ higher up? Actually not. The hypothesis that the [&] feature is on the result category makes the right prediction; the full structure is:

(57) Sally teaches semantics ~~and~~ [$_{S[\&]}$she [[$_{S[\&]/NP}$ plays the crumhorn] and [$_{S/NP}$ studies math]].

This is completely analogous to the situation with "too high" *either*. The [&] feature at first (and hence the silent ~~*and*~~ which looks for this feature) seems to be too high, but since it is encoded on the result it is not. And so ~~*and*~~ is happy here; it requires as its right argument something with that feature, and it finds it.

We should also find this when if the overt *and* conjoins lifted objects. While perhaps awkward, (58) also seems alright with the right prosody (i.e., a large pause before *and*), and we correctly predict that these can also have an *or* reading as in (59):[20]

(58) Barry will play the crumhorn, Patty will play the flute, and the Irish whistle.

(59) In order to have enough courses, either Lee should teach Pragmatics in the fall, Sandy teach Intro in the spring or (else) develop a new course for January term.

And we should find the same phenomenon with conjoined lifted objects and VPs - and we do. Recall that (52b) and (53b) had pragmatically strange readings: these are exactly the readings here. The following removes the pragmatic anomalies:

(60) a. Sarah conducts an orchestra, plays the violin and occasionally the viola.

b. Sarah deserves tenure. She wrote three papers while on sabbatical, taught syntax the following fall, and semantics that spring.

Because the [&] and [v] features which seem 'too high' must be on a result category, we also predict that they will stop at an S node (see fn. 18). Thus *and* will not license silent comma chains higher up if it is an embedded S and the silent conjuncts are higher. This appears correct; compare (60a) and to (61):

(61) *Sarah conducts an orchestra, claims that she plays the violin and occasionally the viola.

6. Conclusions

I have made three main suggestions embedded within a CG framework. The main one is that 'function' categories correspond to functions taking two strings as argument to give a third. Once we add in Lift and Function Composition, this view predicts that only certain types of lifts and compositions are available - just those which literally correspond to those operations. Of interest here is that this precludes the possibility of what we are calling S-crossing composition. Second, I have proposed a structure for comma chains which gives them a right branching structure and accounts for their basic distribution and interpretation. The third is that features on function categories are on the result. I have also argued against a syntactic CSC (hardly a new position). Taken together, these account for a number of new and not new facts: the existence of RNR Lakoff chains and - more strikingly - the fact that these must have a gap in the rightmost conjunct; the existence of left looking Lakoff chains (i.e., the surprising MM cases) and the Grow Only Rightward phenomenon; (tentatively) the existence of both 'too low' and 'too high' *either;* and the surprising (at first glance) possibility of things like (56)-(59) in left looking Lakoff chains.

Appendix One: Comma chains as flat conjunctions?

There have in the literature been proposals to the effect that chains of silent conjuncts/disjuncts involve a 'flat' structure (e.g, Gazdar et al 1985). For example, one might propose something like the following phrase structure rule schema for these:

(62) X --> X* and X (or *or)*

Note that * here is the Kleene start - the interpretation of this means any number of expressions of category X. (Hence this abbreviates an infinite number of 'ordinary' rules.)

This can have an associated semantics such that the relevant operation is that which is dictated by overt conjunction. In the case of *and,* for example*,* the meaning of the mother is the generalized meet operation of the daughters. (Because all connectives that allow for comma chains have an associative semantics this type of semantics for (62) generally works - but see the remarks on *but* below.)

This admittedly has one striking advantage: there is no need for any kind of feature passing to keep track of the semantics. Whether the last part of the chain is *and X* or *or X,* the chain will be interpreted consistently throughout. That said, there are both theoretical and empirical reasons for rejecting this or any similar schema. As to the theoretical reasons, note first that rules of this general form are not compatible with any theory countenancing only binary rules as in Minimalism or the version of CG here. Of course the hypothesis that all rules are binary could be wrong. But aside from the question of whether all rules should be binary, this requires the incorporation of a new device into the system- the Kleene star - a device which does not seem to have any other motivation.

There are also empirical problems. The first centers what has gone under the rubric of Weak Crossover effects (WCO). By way of background, consider one of the arguments that has been given for the right branching structure for ordinary conjunctions like *walk and chew gum.* It is fairly standard to assume that the 'binder' of a pronoun - in standard terms - must c-command it at some relevant syntactic level; this is used account for the familiar WCO asymmetries like *Every third grade boy* boy_i *called* his_i *mother.* vs. *His_i *mother called every third grade boy.* I have phrased this here in the standard term of 'c-command' but the framework assumed here could not state a principle using c-command,

as trees are only representations of how the syntactic and semantic combinatorics work. Nonetheless, the analogous prediction is made in the variable-free theory of 'binding' put forth in, e.g., Jacobson 1999. The basic prediction is that the 'binding' effect will hold between a higher (or later argument in) slot (e.g., the subject slot) and a lower slot which can ultimately contain a pronoun. (Thus then notion of one expression X c-commanding another expression Y translates into the fact that X is introduced later in the syntactic/sematic combinatorics.) And, as noted in Munn 1993, the fact that we find asymmetries like (63) is one reason to conclude that *and* takes its right argument first and then its left (see also Wagner 2005):[21]

(63) a. Every third grade boy and his mother should report to the principal.

b. *His mother and every third grade boy should report to the principal.

Returning to the analysis of comma chains, the same sorts of asymmetries argue against a flat structure, as we get the same types of contrasts here:

(64) a. Every third grade boy_i, his_i mother, and the principal will attend the ceremony.

b.*His_i mother, every third grade boy_i, and the principal will attend the ceremony.

c. Every third grade boy_i, his_i mother, and his_i dog should all immediately leave.

The flat structure would allow for all the good sentences (under a c-command account) but it should also allow for the bad ones. (In the variable free analysis of Jacobson they should all be bad as there is no actual item to undergo the shift that accomplishes binding.)

A second empirical problem with the flat structure is that there is no obvious way for it to allow for the cases in Sec. 5.4 (examples (56) and (58)-(60)). Under the flat analysis, the *and* in these cases really is too low and cannot be part of a chain licensed by

any kind of rule of the general form *X --> X* and X* (or any modern equivalent of this). Recall that these examples follow from the right branching analysis combined with the hypothesis that the feature is on the result category. And actually the problem with the flat analysis is actually more serious: it has no (ready) account of MM cases in general. To this end a referee suggests that those all involve ellipsis (see Appendix 2). But even if we did posit ellipsis for the basic MM facts, it would not extend to the cases in 5.4, as the interested reader can verify. Along these same lines, it provides no ready account of the distribution *either* in comma chains such as (49), unlike the right branching analysis.

Finally, there is an interesting behavior of *but* in comma chains (both ordinary ones and the Lakoff variety of these):

(65) a. Barry plays the dulcimer, the bass, but not the flute.

b. Barry will play the dulcimer, the bass, but won't do magic tricks.

It is widely agreed that *but* has the same truth conditions as *and* but with an extra implicature. The chain as a whole has *and* semantics, with the *but*--implicature is there only on the last conjunct. This is easy to account for in the right branching analysis, which allows a compositional analysis conjunct by conjunct. To show this, it must be that there is no silent ~~*but*~~ since only the last part the *but*-implicature. Let the category of *but* - like that of *and* - be $(X[\&]/_L X)/_R X$. Then this launches a comma chain with *and* semantics throughout (licensing ~~*and*~~ higher up). But the overt instance will have the *but* implicature. It is unclear to me how to formulate the relevant semantics under the flat analysis. One needs a generalized 'meet' or 'join' (or 'sum') operator for the semantics of the whole chain under a phrase structure rule schema like that in (62), where meet, join, or sum will be

determined by the overt member. But there is no obvious way to fold in the extra implicature that that last member contributes as it has no compositional contribution of its own; it merely determines the semantics of the whole chain.

Appendix Two: Ellipsis for the MM examples?

A referee suggests that the left looking Lakoff chains are not leftward function composition, but instead involve ellipsis. This was paired with the idea that the comma chains use a flat structure which would necessitate some such analysis for reasons mentioned above.. But even if one accepts the right branching analysis of Lakoff chains, there remains the possibility that these involve ellipsis, so it is worth considering. While I cannot rule this out, I believe there are reasons to be suspicious that ellipsis is at work here.

First, we need to say something about how ellipsis is allowed in this configuration. Note that as long as one grants that the right branching structure is motivated (and that there is function composition), then nothing more is needed for the left-looking Lakoff chains. More seriously, ellipsis by itself does not account for the Grow Only Rightward phenomenon. To that end, the referee suggests that ellipsis is subject to a constraint that the antecedent must be in the immediately preceding clause. This is quite clever and delivers the desired result. But - while there is often a preference for closer antecedents - that is certainly not a general property of ellipsis. Even the rather fussy case of Sluicing is not subject to such a constraint as in (66), and we can also construct an 'at a distance' case in a Lakoff chain as in (67):

(66) I've decided to buy one of those hats. I'll see how much money I have and then I'll figure out which.

(67) At the graduation ceremony, let's announce that someone in the class has just won a huge prize, have a big drumroll, and then we can announce who.

While (66) may be a bit awkward, (52b) and (53b) are different. These simply cannot pick up their antecedent from two conjuncts earlier and so force a pragmatically deviant reading. Take (53b): *Barry builds dulcimers, plays the standup bass, and treehouses*. This has only the bizarre reading where Barry plays treehouses. Yet if this were only a matter of a processing preference, that should be overridden by the pragmatics. Take (68):

(68) Lee hid my ring somewhere. Sandy will talk to him and will then tell me where.

Grammatically there is no reason why *will then tell me where* could not be picking up the location of where Sally talks to Lee (on a Sprouting reading). If there were just a strong processing principle forcing the immediately preceding clause to be the antecedent, that should be the reading for (68). But indeed here the pragmatics clearly wins out, and the 'at a distance' reading of (68) is quite smooth. Thus having to pick up the antecedent from the immediately preceding clause is neither a general property of ellipsis nor a general processing strategy, and so would need to be stated idiosyncratically for ellipsis in comma chains. (Indeed, it is also well that in ellipsis in general the antecedent can follow, as in *'While I don't know who, I heard that she nominated someone'* But if this were allowed in comma chains we would predict that *Barry played the dulcimer, the flute, and Patty played the crumhorn* could mean that Patty played the flute. So comma chains just do not behave like other types of ellipsis.)

Third, binding facts again indicate that this cannot be right, consider:

(69) The judge sent a warning to every defendant, his lawyer, his bail bondsman, told them to all be in court the next day, and made it clear that she was not very happy.

While there are issues about how binding works in these coordination cases (see fn. 21), it is clear that (69) is far better than (70) with full Ss suggesting that (69) cannot be accounted for by simply scoping out *every defendant* to have scope over all the conjuncts:

(70) *The judge sent a warning to every defendant, the judge/she sent a warning to his lawyer, she sent a warning to his bail bondsman, she told them to all be in court ...

All in all the ellipsis hypothesis faces challenges.. And if we cannot maintain ellipsis we also cannot (easily) maintain a flat structure for Lakoff (and comma) chains in general.

Endnotes

[1] In accordance with reasonably standard terminology in CG (and many other places) I use the term NP; the reader can substitute DP if preferred. It should also be noted that I call these "NP" but in fact, following Partee and Rooth's 1983 semantics for coordination these cannot have meanings of type e but have to be lifted to the generalized quantifier type <<e,t>,t>. Given the other assumptions in this paper (see Sec.2.5.1) this means their category is not NP; for now call it GQ.

[2] Thus the bulk of this paper centers around surprising coordination of unlikes in comma chains. Of course there are many other well known cases of coordination of unlikes such as *Lee is a linguist and proud of it* or cases of CP and NP coordination as in *Lee believes that the earth is flat and every other silly thing on the internet.* Those, however, are different from the type of coordination of unlikes in this paper: those are all instances where an expression of category X may conjoin with an expression of category Y just in case both

are allowed (separately) in the same environment. Cases of this form can always be handled by positing an underspecified category, or a union category (see, e.g. Bayer 1996). This is not the situation with the M&M cases, since while we can get *Barry played the flute* we cannot get ***Barry played does magic tricks.*

[3] I am grateful to Dick Oerhle for pointing this out to me, and for pointing out that with this there is no mystery as to why, e.g., lift rules preserve word order (see Sec. 2.5.1).

[4] The view that *and* is polymorphic in the lexicon is quite common, but one might be unhappy at the idea of a lexical abbreviating an infinite number of items (since X can range over any category). Jacobson 2014 (Ch. 11) suggests a possible alternative whereby there is a single item in the lexicon (or perhaps two) and the others derived from that.

[5]Actually in all cases there are an infinite number of derivations yielding the same meaning, since the subject and VP can keep lifting over each other.

[6] It has often been argued RNR is more general and not just about coordination - a thorough discussion is in Chaves 2007. For example, we find cases like *The student who loves __ was assigned to mentor the student who hates __ OT phonology.* We have nothing to say about these cases here and hope that they can be handled in the same way that *wh*-type extraction parasitic gaps are. For example, Steedman 1987 provides an account of parasitic gaps using an additional combinatory principle within the general CG apparatus, and even though I am treating (leftwards) *wh*-type extraction differently from rightward cases the same idea could be extended to the right cases. That said, I do not see at this point how to fold that operation into the general view of categories taken here, so I leave that open.

[7] The reader may wonder about cases like *Lee cooked and Sandy ate scallops on Monday and lobster on Tuesday.* This can be put together by having *cook* and *eat* first lift from TVs to category $VP/_R(VP/_LTV)$. (Recall that TV is an abbreviation for $VP/_RNP$). As seen earlier, *Lee* and *Sandy* can both lift to the ordinary category for generalized quantifier subjects: $S/_RVP$. Since those can compose with *cook* (and *eat)* by right composition we get *Lee cooked* of category $S/_R(VP/_LTV)$, and similarly for *Sandy ate.* These two can cojoin, and *Lee cooked and Sandy ate* is of course also of category $S/_R(VP/_LTV)$. Note that $VP/_LTV$ is the category of both a lifted object and also of a 'fancy object' like *scallops in Monday,* as well as being the category of the conjoined fancy object *scallops on Monday and lobster on Tuesday.* It is therefore the right sort of thing to be argument of *Lee cooked and Sandy ate.* The semantics comes out as expected as an interested reader can verify.

[8] Actually even this is an oversimplification. Not every category can instantiate X. As discussed in Sec. 2.4, many theories including the one here treat *and Lee* as a constituent in *I recommended Sally and Lee.* Yet we do not get ***I recommended Sally and Lee or and Sandy* (meaning "I recommended Sally and Lee or Sally and Sandy".

[9] Steedman 2023 does allow for free S-Crossing composition giving a $VP/_LNP$ *believes left* but his system precludes this from taking an NP to its left because in his system plain NPs are never introduced into a syntactic derivation: all "NPs" are type raised via a case marker (crucially also there is no free type raising). There is a marker (ACC) which maps an NP to $VP/_L(VP/_RNP)$ (for object NPs) but there happens to be no case marker yielding a category $VP/_R(VP/_LNP)$. As Steedman notes, he needs a second ACC marker of category

$S/_L(S/_RNP)$ to account for RNR cases like *Lee loves and Sandy hates Phonology,* somewhat undermining the beauty of having RNR follow immediately from function composition.

[10] Pollard's analysis was not embedded within a CG but is easily translated into CG. Because his dissertation is not well known and not readily available on line, we briefly summarize the key ideas (translating into the system here). Thus take a Dutch case (i):

(i) ...dat Jan Piet Cecilia zag helpen zwemmen

... that Jan Piet Cecilia saw help swim

"...that Jan saw Piet help Cecilia swim "

The crucial verbs (here *zien* and *helpen)* take an object and a VP or S complement (depending on one's theory). The analysis is neutral with regards to the theory of control; here we follow Pollard and treat the complements as just VPs. As Dutch is SOV, verbs take NP objects to the left. But rather than taking the VP complement either to the left or to the right, the complement is taken as a circumfix, meaning that the control verbs are infixes. Thus (presumably after word order rules apply) a verb like *helpen* is $((S/_LNP)/_LNP)/_CVP$. Pollard takes all expressions to be headed and states infixation points in terms of heads: in Dutch infixes go to the left of the head (in English - to the right). Moreover, heads are inherited in the expected way. This gives the necessary ingredients; we leave it to the interested reader to work through the example and see how this generalizes to the full range of cross-serial dependencies.

[11] Another well-known difference noted as early as McCawley (1982) is that RNR does not obey island constraints while of course *wh* extraction does. But since it has often been suggested that the prohibition from extracting out of relative clauses and *wh* questions is

due to processing principles, it is conceivable that island effects show up when the 'filler' is on the left but not on the right. This is of course merely suggestive, but I think it would be premature to use the island differences as conclusive evidence for a difference between right and left (*wh* type) extraction.

[12] Steedman 2023 (fn. 37) claims that this follows from the semantics of *and* and *or.* Like the view here, he adopts the Partee and Rooth semantics such that *and,* for example, is λp[λq[p ∏ q]] for p and q variables of any type whose ultimate result is t. He then claims that because ∏ is only defined for two expressions of the same semantic type, it follows that p and q (the expected arguments) must be of the same type. This much is correct: [[and]] expects two arguments of the same type. But why should it not be able to participate in, e.g., function composition just like any other function can? (Or, map to a more complex function by **g?**). *Before,* for example, also expects two arguments of the same type (in this case propositions) but allows extraction. Thus one instantiation of (X/X)/X is (VP/VP)/VP in which case it denotes a function taking two VP meanings (extensionally, sets) and giving their intersection. Nothing in the semantics stops such a function from composing with a TV type meaning (as is shown by many examples in this paper whose semantics are perfectly coherent). The only way I see to have the lexical specification of *and* preclude composition would be by an assumption about how categories are instantiated in the case of polymorphic items like *and;* perhaps one can assume that the variable X over categories can only be instantiated during application. But even so, that does not block 'extraction' (*wh* type or RNR) from the left argument. Once X is instantiated as VP the category of, e.g. *and chew gum*, is VP/$_L$VP; and the left argument slot should allow composition.

[13] I thank a referee (hereafter R) for a comment which led me to look at sum *and* in Lakoff chains. R's particular example was slightly different than the one discussed here, and was used to make a different point. Thus R points out the following. Suppose I am in charge of pairing kids in need of adoption with couples, and you ask which kid I will pair with which couple. Let Lee be a kid, and Sandy and Kim a couple. I can answer (with the right prosody) *I'll pair Lee and Sandy and Kim* but not **I'll pair Lee, Sandy, and Kim.* R took this to be evidence for a flat structure for comma chains (see Appendix 1) rather than the right branching structure here. Presumably the intuition is that under a flat structure there is no way to collect up two of the disjuncts (here *Sandy and Kim)* to form a couple and then coordinate that (by a pairing interpretation of *and*) with the other disjunct (*Lee).* R assumes - as do I - that the version with overt *and*'s can have a right branching structure, which can then be put together semantically so as to give the desired reading.

The example is intriguing and the judgment robust, but I don't think it leads to the conclusion that comma chains cannot have a right branching structure. For even when all *and*s are overt, a right branching structure is not enough to give the relevant interpretation; we need to understand the meaning of each *and* so as to get that interpretation. The most obvious possibility to try is that this is the sum *and* throughout. But that will not yield the child + couple reading, because sum *and* (like Boolean *and)* is associative, so a string of sum *and*s will just give a group. Perhaps one of the *and*s is sum and the other Boolean. But if so, then this is exactly like the case discussed above; we do not have the same conjunction throughout the chain. Or perhaps there is yet another *and* - call that pair *and* - which can only pair two things (in this case an individual and a plural individual). And

suppose that a verb like *pair* selects for this: i.e., it selects for an NP with the feature [p&] on it. As to the *and* for the couple, it is conceivable that that is just sum *and*; perhaps our new 'pair *and'* is selected only by verbs like *pair*. If so, these too are different *and*s throughout (in the overt case) and so the conditions are not met for a comma chain. But there is more to say: it is conceivable that the new pair *and* can also be the one for the couple. In that case, we do get the right reading when overt throughout, and we would possibly expect this to support a comma chain. But even if so, it is also perfectly possible that our new 'pair *and'* (if that is the correct way to look at this) does not have a silent variant. As support, note that this *and* also does not have the reduced *'n* variant: thus **I'll pair Lee 'n Sandy 'n Kim* also lacks the relevant reading. (There is sum *'n* as we can get chains like *Lee 'n Sandy 'n Kim are a set of triplets.)* Note *I'll pair Lee and Sandy 'n Kim* is good on child + couple reading expected; *'n* here can simply be the ordinary sum *and* (while *and* can be the pair *and.)* In any case, I think the composition of the relevant reading under the overt-throughout right branching structure is sufficiently unclear that the fact that we don't have this with a comma chain tells us nothing definitive about the comma chain.

[14] There is one wrinkle. For overt *and*, it was ensured that the innermost slash had a left feature on it because that was a modifier category - in other words *and chews gum* is of category VP/VP and so it follows that it takes its argument to the left.. Similarly, we need to ensure that ~~*and*~~ *walks and chews gum* also takes its next argument to the left (as in *sings,* ~~*and*~~ *walks and chews gum)*. But technically the category of *and chews gum* as well as of ~~*and*~~ *walks and chews gum* is not a modifier since it these can combine with any ordinary X (without the [&] feature) to give something with that feature. In other words, the

argument of this expression and the result are not precisely the same category. Hopefully this depends on an exact definition of 'modifier' (i.e., an exact specification of the directional slash rules) which allows for a bit of featural mismatch. We leave it open here as to how to do this, but it does not seem insurmountable.

[15] There is a second technical problem. Consider *After college, Lee will go to graduate school, join a fancy company, or [inherit a batch of money and travel the world].* (The bracketing is to show the intended reading.). The lowest conjoined VP has the category [S&/NP]. Indeed it needs that [&] feature since it could combine to its left with *and* in which case it would carry along the [[and]] semantics. But here it happens to find an overt *or* to its left, whose category is (X[v]/X)/X. And we certainly want *join a fancy company or [inherit a batch of money and travel the world]* to have the [v] feature on the final result to give the semantics for the whole S. The issue, then, is that there are competing principles dictating the feature on the result category in *join ... or [inherit ... and travel]].* If *or* is the boss it should be the feature [v] (and this is what we want given the intended reading), but by inheritance from the lowest *and* it should be &. We will assume conventions by which the two features are mutually exclusive, and when the feature passing conventions conflict with the lexical specification of, e.g., *or,* the latter takes priority.

[16] As is well known, there is a strong principle which has the effect ruling out extraction *of* a conjunct (rather than just *from* a conjunct) as in (i). This does not follow immediately from most accounts of extraction including the one here, so we leave this as unsolved:

(i) a. *Which table did Lee build a matching chair and __?

b. *Which table did Lee build __ and a matching chair?

[17] One can get structures here *and/~~and~~* combines first with the left argument by function composition and combine with the right argument. Unsurprisingly, the generalization discussed in Sec. 5.2. is not changed by this as the interested reader can verify.

[18] There is more to say as the literature has documented a number of interesting constraints on 'too high' *either,* which we do not account for here. But at least some do fall out from the idea that features are encoded on the result. For example, Larson 1985, among others, claims that *either* cannot go higher than a tensed S, as seen in contrasts like (i)

(i) a. I said that either she should be nominated or there should be an open convention.

b.*I said either that she should be nominated or there should be an open convention.

This would follow here since the [v] feature would stop at S and not go onto the CP. The empirical generalization, however, is controversial as Hofmeister 2010 conducted a large scale judgment study which found that *either* can in fact go above embedded Ss. I have no account of these findings, but will note that the one example he gives (taken from actual corpus data) is extremely confusing. I think we would need to look more closely at the examples and also see if the results can be replicated before drawing any firm conclusion.

That said, there are other constraints on 'too high' *either* which I leave unaccounted for. In particular, Larson 1985 showed that (ii) has no narrow scope de dicto reading:

(ii) Either they are looking for a semanticist or a phonologist.

But while I have no full account I offer a preliminary speculation. This is that for some reason the relevant feature launched by *or* stops going up when an intensional object like this finds itself as the object of an intensional verb (the inability to occur with *either,* then, is a fact about the feature stopping here). Some support for this hypothesis is in fn 20.

[19] In an earlier version of this paper I reported these judgments as surprisingly good but not perfect. I now think they simply require a certain prosody; others have confirmed this. To be honest, I do not know whether my earlier hesitation was being driven by then having no account of them, or whether my current acceptance of them is driven by the fact that I have since realized they are predicted here. More systematic informant work is needed.

[20] If the speculation in fn. 18 is correct, we would expect to find intensional objects conjoined with *or* (with a de dicto reading) to not support a higher silent *or*. This seems correct. Imagine a fledgling Linguistics Department which has many needs but knows they will be given minimal resources. Suppose they hope that at least they can move into a new building, or they can have a dedicated search for a phonologist or a dedicated search for a syntactician. Now take the same scenario but where their hope is to move or to be allowed to conduct a disjunctive search (just looking for the best candidate). We can report the first, (with the right prosody) but not the second scenario as follows:

(i) The hope is to be allowed to move into a new building, search for a syntactician or a phonologist.

It is striking that the distribution of 'too high' *either is* exactly the same as the distribution of 'too high' silent conjunctions; neither support the de dicto reading of the lowest disjunct.

[21] It is, however, not completely clear how to accomplish the binding in (63) either under the standard account of binding or Jacobson's variable free account. For the standard account, one can put the semantics together by having the GQ scope out of the conjunct so that the interpretation of the rest is λx[lift(x) ∏ lift(the-mother-of(x)(report-to-the-principal)]. The problem is that usually GQs cannot scope out of conjuncts. This is an

empirical fact independent of any theoretical commitment to a CSC: *?* Every third grade boy and Sally will call his mother.* The variable free account has further problems as the binding is generally by e-type argument slots, yet Boolean [[and]] is not defined for individuals. While one can generalize to allow binding by other argument slots, getting the right semantics requires further additions to the system.

References

Altshuler, D. and R. Truswell 2022. *Coordination and the Syntax-Discourse Interface.* Oxford: Oxford University Press.

Au (in submission). Title and venue suppressed here as it would compromise anonymity.

Aus (in preparation). "Discontinuous Constituents in Categorial Grammar" (tentative title).

Bach, E. 1979. "Control in Montague Grammar", *Linguistic Inquiry* 10.4, 515-31.

Bach, E. 1980. "In Defense of Passive", *Linguistics and Philosophy* 3, 297-341.

Bayer, S. 1996. "The Coordination of Unlike Categories". *Language* 2.3. 579-616.

Chaves, R. 2007. *Coordinate Structures - Constraint-Based Syntax-Semantics Processing.* Ph.D. Dissertation, University of Lisbon.

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

Chomsky, N. 1970. "Remarks on Nominalization", in R. Jacobs and P. Rosenbaum (eds.), *Readings in English Transformational Grammar.* Waltham: Ginn, 184-221.

Den Dikken, M. 2006. *Either*-float and the syntax of co-ordination. *Natural Language and Linguistic Theory* 24(3):689–749.

Dowty, D. 1982. "Grammatical Relations and Montague Grammar", in P. Jacobson and G.K. Pullum (eds.), *The Nature of Syntactic Representation.* Dordrecht: Kluwer, 79-130.

Dowty, D. 1985. "On Recent Analyses of the Semantics of Control", *Linguistics and Philosophy* 8.3. 291-331.

Dowty, D. 1988. "Type Raising, Functional Composition, and Non Constituent Conjunction", in R. Oehrle, E. Bach and D. Wheeler (eds.), *Categorial Grammars and Natural Language Structures.* Dordrecht: Kluwer, 153-197.

Gazdar, G. 1979. "English as a Context Free Language". ms., Sussex: University of Sussex.

Gazdar, G., E. Klein, G.K. Pullum and I. Sag, 1985. *Generalized Phrase Structure Grammar.* Oxford: Blackwell and Cambridge, MA: Harvard University Press.

Goldsmith, J. 1985. "A Principled Exception to the Coordinate Structure Constraint", *Proceedings of the 21st Meeting of the Chicago Linguistic Society.* Chicago: Chicago Linguistics Society, 133-143.

Jacobson, P. 1987. "Phrase Structure, Grammatical Relations, and Discontinuous Constituency", in G. Huck and A. Ojeda (eds.), *Discontinuous Constituency* (Syntax and Semantics 20). New York: Academic Press, 27-69.

Jacobson, P, 1999. "Toward a Variable Free Semantics", *Linguistics and Philosophy* 22, 117-184.

Jacobson, P. 2014. *Compositional Semantics: An Introduction to the Syntax/Semantics Interface.* Oxford: Oxford University Press.

Hendriks, H. 1993. *Studied Flexibility: Categories and Types in Syntax and Semantics.* Ph.D. Dissertation, University of Amsterdam Institute for Logic, Language and Computation.

Hendriks, P. 2004. 'Either, Both and Neither in Coordinate Structures', ms., University of Groningen.

Hofmeister, P. 2010. "A Linearization account of *either...or* constructions". *Natural Language and Linguistic Theory* 28, 275-314.

Kehler, A. 1996. "Coherence and the Coordinate Structure Constraint" *Proceedings of the Twenty-Second Annual Meeting of the Berkeley Linguistics Society: General Session and Parasession on The Role of Learnability in Grammatical Theory*. Linguistic Society of America *eLanguage,* 220-231.

Kuno, S. 1987. *Functional Syntax - Anaphora, Discourse and Empathy.* Chicago: University of Chicago Press.

Lakoff, G. 1986. "Frame Semantic Control of the Coordinate Structure Constraint". Proceedings of the 22nd Annual Meeting of the Chicago Linguistics Society. Chicago: Chicago Linguistics Society. 152-167.

Larson, R. K. 1985. "On the syntax of disjunction-scope". *Natural Language & Linguistic Theory*, 3: 217–264

Larson, R. 1988. "On the Double Object Construction", *Linguistic Inquiry* 19:3, 335-391.

Link, G. 1983. "The Logical Analysis of Plurals and Mass Terms: A Lattice-theoretical Approach", in R Bäuerle, C. Schwarze, and A. von Stechow (eds.), *Meaning, Use, and Interpretation of Language.* De Gruyter.

Maxwell, John T. and Manning, Christopher D. 1996. A Theory of Non-constituent Coordination based on Finite State Rules. In Proceedings of the First LFG Conference. Grenoble: CSLI Publications.

McCawley, J. 1982. "Parentheticals and Discontinuous Constituent Structure", *Linguistic Inquiry* 13. 91-106.

McCloskey, J. 1986. "Right Node Raising and Preposition Stranding", *Linguistic Inquiry* 17.1, 183-186.

Miller, P., 1990. "Pseudo-gapping and do so substition". *Proceedings of the Twenty-Sixth Chicago Linguistics Society* 1. Chicago: Chicago Linguistic Society, 293-305.

Montague, R. 1973. "The Proper Treatment of Quantification in Ordinary English", in P. Suppes, J. Moravcsik, and J. Hintikka (eds.), *Approaches to Natural Language.* Dordrecht 221-242.

Munn, A. 1993. *Topics in the Syntax and Semantics of Coordinate Structures.* Ph.D. Dissertation, University of Maryland.

Oehrle, Richard, 1990. "Categorial Frameworks, Coordination, and Extraction" in *Proceedings of the Ninth West Coast Conference on Formal Linguistics.* Stanford: Stanford Linguistics Department. 411-426.

Partee, B. and M. Rooth 1983. "Generalized Conjunction and Type Ambiguity", in R. Bauerle, C. Schwarze, and A. von Stechow (eds.), *Meaning, Use and Interpretation of Language,* Berlin: De Gruyter, 361-383.

Pollard, C. 1985. *Generalized Phrase Structure Grammar, Head Grammars, and Natural Language.* Ph.D. Dissertation, Stanford University.

Schwarz, B. 1999. On the syntax of *either ... or*. *Natural Language & Linguistic Theory*, 17:339–370.

Steedman, M. 1987. "Combinatory Grammars and Parasitic Gaps", *Natural Language and Linguistic Theory* 5, 403-439.

Steedman, M. 2023. On Internal Merge. Linguistic Inquiry, 1-83. Advance online publication. https://doi.org/10.1162/ling_a_00521