From GPSG to Categorial Grammar (and partway back again)

Pauline Jacobson

Brown University

Appendices

Appendix 1: A brief comparison with Type Logical Grammar

A1.1. Languages (or, grammars) as logics

The approach here has some features in common with what is known as Type Logical Grammar (hereafter TLG), exemplified first in Lambek (1958) and by many subsequent works inspired by that. Especially striking are some similarities with the account(s) of extraction in much TLG work, but there are also other points of convergence and divergence. That said, a thorough comparison with TLG is obviously beyond the present scope; it would in any case be complicated by the fact that there are different types of TLGs and different accounts of empirical phenomena within the broad TLG tradition. Therefore, I will concentrate here primarily just some 'big picture' questions, leaving a comparison of the treatment of, e.g., extraction (and quantification) to brief remarks at the end (Section A1.6). And in order to make this concrete I focus primarily on the version of TLG in Kubota and Levine (2020) (hereafter K&L).

First, I would argue that part of the apparent difference between TLG and the theory here (as well as many other theories) boils down to preferred terminology and emphasis rather than deep differences in the conception of grammar. This might seem like a surprising claim in view of the fact that a good deal of work in the TLG tradition features as its fundamental and guiding principle the idea that grammar can be seen as a (kind of) logic. But this does not distinguish TLG from other theories. There are many different logics, and the general idea of a grammar as 'a logic' - although often not phrased that way - is not unique to TLG. Surely one can argue that any formal

grammar of a language is a 'logic' (or, put differently, that any language is a logic). The question is just what kind of logic.

Of course the above claim (that many theories take language as a logic) depends on how one defines 'a logic'. To that end, consider the definition of a logic from the *Stanford Encyclopedia of Philosophy* (Shapiro and Kissel, 2022): "Typically, a *logic* consists of a formal or informal language together with a deductive system and/or a model-theoretic semantics. ... The deductive system is to capture, codify, or simply record *arguments* that are *valid* for the given language, and the semantics is to capture, codify, or record the meanings, or truth-conditions for at least part of the language." Is there a sense in which TLG fits this definition in a way that the CG given here - or, for that matter, many other theories of grammar - don't? I don't think so. It is obviously true that most theories of grammar (including TLG) don't use the terminology of a system to capture 'arguments that are valid for the given language', because the term 'argument' is usually taken to be about reasoning, and not expressions of a language. Nonetheless, in the context of a grammar or a theory of grammar, one can take the notion of a 'valid argument' to mean a well-formed expression. As such, our grammars should be able to supply a proof of the well-formedness of any such expression. The goal of supplying grammars that do that holds, of course, for many different grammars and/or theories of grammar.

That some of the apparent differences between TLG and other theories are really matters of terminology is, I think, well illustrated by the following quote from K&L (2020: 19): "Instead, Lambek's system licensed sentences in precisely the way a logical proof licenses a conclusion based on one or more premises. In place of phrase structure rules, what has become known as the Lambek calculus provided an authentic logic - a set of connectives and rules of inference governing their use, which allowed one to deduce sentences as, quite literally, *theorems* of the

lexicon under the proof theory provided by the calculus." (The use of "instead" here is due to the fact that K&L are comparing this to a tradition of a theory with at least phrase structure rules and with a heavy emphasis on the trees. But the emphasis on trees - which are just the representations of the well-formedness of an expression according to a set of phrase structure rules - is both absent in other theories and is in any case orthogonal to the question of whether a system is a logic.) Leaving aside the question of what K&L mean by an 'authentic' logic, there is no obvious sense in which the Lambek calculus is a logic while, for example, a (context-free) phrase structure grammar is not. Any phrase structure grammar is a system which 'licenses' (i.e., proves well formed) sentences (and, for that matter, other expressions) via a logical proof based on premises. The premises are the phrase structure rules: a rule often written as A --> B C is just one notation for saying "if there is an expression of category B immediately to the left of an expression of category A then that whose string is a well-formed expression of category A". There are any number of other ways one can restate that same information. The lexicon gives the categories for the basic items, and so if one wants to call the sentences 'theorems' no harm is done. But the point here is simply that this too is a logic; once again the question is just what kind of logic is natural language(s).

I have used context-free phrase structure grammars simply for the sake illustration, but of course these are not the only systems which prove expressions well formed. In fact they are arguably overly impoverished (since we know that natural languages are not all context-free) as they allow only expressions to concatenate. And so indeed many theories of grammar really do not have trees as appropriate representations. Such an example would be a Categorial Grammar with infixation operations as in Bach (1979, 1980). Nor are Montague's 'analysis trees' actually trees in the relevant sense; they represent steps in a proof but some of those steps

go beyond context-free rules. And for that matter, transformational theories also do not represent a sentence as a tree (it is at best a sequence of trees). So the role of trees and/or constituent structure in actual grammars is a complex topic, but irrelevant to the question of what is a logic.

A1.2. Slash Elimination, Functional Application, and Modus Ponens.

Like many other Categorial Grammars, K&L begin with a definition of categories that includes basic (atomic) categories and 'function' categories - using the notation here $A/_{L}B$ and $A/_{R}B$. (We leave aside the non-directional or 'vertical' slash adopted both in K&L and here, returning briefly to that at the end of these remarks.). And like the generalized rule given above in (6), K&L (as with many others in the general Lambek-inspired TLG tradition) adopt two "slash elimination" rules. Right slash elimination allows a string of the form x-y to be proven to be a well-formed string of category A if x is of category $A/_{R}B$ and y is of category B, and the left elimination rule is just the mirror image. Note that under the view in this paper that function categories correspond to actual functions in <string, <string,string>> we were able to collapse the two corresponding rules into one (6). While K&L (and much other work within TLG) do not do this, there are undoubtedly ways that a similar move could be made (in fact the view of categories taken here could probably also be taken in TLG). So I would not press the point that this is a disadvantage of the basic architecture of TLG, although it is ultimately incumbent on TLG proponents to show how this collapsing is possible as it is obviously a defect to have two separate elimination (and slash introduction) rules.

That said, one interesting consequence of the insistence on thinking of grammar as a logic is that it reminds us of the commonality between 'functional application' and Modus Ponens. One can take any string x of category $A/_{R}B$ to be a premise saying 'if there is a string y of category

B to the right of x then x-y is a string of category A'. In other words, one can think of an $A/_RB$ combining with a B to its right to give an A to be an instance of (or correspond to an instance of) Modus Ponens. Putting this somewhat pedantically: Premise 1: If there is a y of category B to the right of x of category $A/_RB$ then x-y is a well-formed A. Premise 2: y is of category B and is to the right of x. Conclusion (by Modus Ponens): Therefore x-y is a well-formed A. But one can, of course, recast 'functional application' in similar terms. After all, take a function f which maps a to b; one can recast this as saying "if a is given as argument to f the result is b". As such, of course any instance of functional application also corresponds to an instance of a 'proof' using Modus Ponens.

A1.3. Hypothetical reasoning, and the status of function composition (or, Geach) and Lift

The one substantive and interesting difference that I focus on here between much work in TLG tradition and the present account concerns the status of the rules like Lift and Geach (or, instead of Geach, the binary combinatory rule of Function Composition). Here these are stipulated as additional unary or binary combinatory rules allowed in the grammar (beyond just functional application). TLG does not stipulate these as special operations, but instead uses two other pieces of apparatus: hypothetical reasoning and Slash Introduction. And from those, the effects of Lift and Composition (or, Geach) follow.

Thus in place of adding these combinatory rules, the first piece of relevance here - which plays a heavy role in the entire system - is the use of the analogue to hypothetical reasoning. We illustrate first with how this gives us the effect of Function Composition. We remind the reader that in the system here we are not actually directly adding in a function composition binary combinatory rule but rather the unary rule of Geach. However, presumably the effect of that could also be derived; we will use Function Composition instead for expository

simplicity. Hence, in a proof (or, derivation) of the well-formedness of some expression (and the derivation of its meaning), the TLG system makes use of hypothesizing some expression in a string without it necessarily actually being there, and drawing the appropriate consequences one would get if it were there. (We return to the semantics below.)

By way of illustration, imagine a string consisting of an expression x of category $A/_RB$ followed by an expression y of category $B/_RC$. (I continue to use the notation for directional slashes used in this paper rather than the Lambek notation adopted by K&L.) In order to do what positing a rule of Function Composition does, the goal is to prove that the string x-y has the category $A/_RC$ without stipulating a new combinatory rule. Note first that of course "Slash elimination" (aka functional application) does not license the combination of these two. But suppose we temporarily hypothesize that there is an expression z of category C to the right of y; let us notate that as [z]. (This is not their notation.) Then, keeping track of the fact that that is a hypothesis, we would have been able to prove that x-y-[z] is a well-formed A, because y-[z] is a B and that straightforwardly then allows a proof of the whole thing as an A. This is still just under the hypothesis that there is z there; as has been indicated by placing the material in brackets. (K&L use a different notation designed to account for more complex chains of hypothetical reasoning, but the bracket notation is convenient here).

In order to remove the hypothesis, there is one more step - which is the notion of Slash Introduction. Stepping away from the particular example here (which mimics function composition), the general idea of Slash Introduction is as follows. Take some string w and some string v of category E such that hypothesizing v to the right of w gives a well-formed string of category D. (That is, the hypothetical string w[v] is of category D.). Then this means that w has as one possible category the category $D/_RE$. This just follows from the general

notion of the categories: an expression of category $D/_RE$ is something that yields a well-formed D should it find a E to the right. Slash Introduction 'withdraws' the hypothesis (that there is a v to the right) and just gives a proof of the category of the string w. The left Slash Introduction is just the mirror image (again one would ultimately to collapse the two instances of Slash Introduction.)

Applying this to the case at hand ('proving' function composition), we have seen that given a string x of category $A/_RB$ and y of category $B/_RC$ we can determine that x-y-[z] is an expression of category A, provided that z is of category C. But then, by Slash Introduction, we know that the string x-y is of category $A/_RC$. This thus does what Function Composition does. Notice further that the order preserving properties of function composition follows, just as it does in the account in this paper whereby categories correspond to actual functions on strings and the relevant operation literally is function composition (once the first argument is supplied). And finally we can note that the fact that the TLG system gets function composition using hypothetical reasoning is in some sense not surprising. This is because the definition of function composition itself involves a kind of hypothetical reasoning: the composite of g and f is that function that maps any x (in the domain of f) to that which one would have gotten had f applied to x and that result was taken as the argument of g.

Thus whether we call this 'function composition' or hypothetical reasoning might look like a matter of terminology. But there is a difference: by positing hypothetical reasoning (plus Slash Introduction) we also derive other things. For example, Lift also follows 'as a theorem' in this approach; see Lambek (1958) though not phrased in these terms. Take an expression x of category A. Then one can hypothesize an expression [y] of category $B/_LA$ to the right of x, and thus the string x[y] is of category B. By Slash Introduction, that means that x is also a

$B/_R(B/_LA)$. The mirror image of this (the proof that x can have category $B/_L(B/_RA)$) is completely parallel. The 'order preserving' nature of "Lift" follows immediately (as it does here from the definition of Lift and the notion of categories corresponding to functions on strings).

A1.4. Semantics

There is certainly something appealing about deriving both function composition (and/or presumably also the Geach rule) and Lift from the same type of machinery (although it is not cost-free in that it involves the adoption of hypothetical reasoning and of Slash Introduction). But the particular semantics advocated in K&L is, in the opinion of this author, not a simple one. Most notably, they make use of variables and lambda abstraction as key steps in the computation of meaning. Crucially, in order to provide a semantics for the hypothetical reasoning step, they posit a variable of the appropriate type as part of the meaning, so the meaning of a hypothetical expression (along with a silent placeholder δ in the string) such as *see [δ]* is see'(x), for x an open variable. One might argue that the meaning of this is irrelevant in that this is a hypothetical expression and not an actual one. But it needs to be assigned a meaning in order to input the semantics associated with the next step - Slash Introduction - and that step is lambda abstraction over the variable whose corresponding hypothesis is withdrawn, a full blown linguistic expression.

This is not a completely innocent move. The use of variables as part of the semantic machinery is hardly simple and requires introducing a set of assignment functions (just 'assignments' for short) into the semantic apparatus where each such function assigns to every variable some object of the appropriate type. All expressions have values only with respect to

a set of assignment functions (while not usually phrased this way, one can think of their meanings as being a function to the set of assignments to something else). A 'closed' expression (one with no unbound variables within it) assigns the same value to each assignment, while any expression with an unbound variable within it has values which can vary by assignment functions. (For fuller discussion see, e.g., Jacobson (2014).). The next step in the semantic computation - the one associated with Slash Introduction - involves lambda abstraction. While we write this with a simple enough looking operation (just place the symbols λv (for v some variable) in front of the input expression), the actual semantics is not so simple, as any student who has struggled through a full semantic understanding of lambda abstraction knows. We will not write this out here as one can consult any textbook on the subject, but will note that it involves computing the value of the output expression on some assignment by consulting the value of the input on another. As suggested above, the ease with which we can write out the representations (and the fact that writing things with variables is quite familiar) can often mask the complexity that arises with the introduction of variables, assignments, and lambda abstraction.

Compare this to what happens if function composition is directly introduced as a binary combinatory principle. (Very similar remarks hold if we adopt the unary version Geach instead; but again we use function composition for exposition.) There is no need for variables or assignments as part of the semantic machinery in order to state this operation. Indeed, the Curry and Feys logic (Curry and Feys (1958)) on which Steedman based much of his work is variable-free. Of course we can write function composition with variables if we want, but that

is notation only; there is no step in the semantic computation crucially relying on assignments. Function Composition is just function composition.

Similar remarks hold for Lift. With hypothetical reasoning, one takes some string x of category A and hypothesizes a string [y] to the right (or left, as the case may be) whose category is $A/_L B$ (or $A/_R B$ if hypothesized to the left). But what is the meaning of the hypothetical string x[y]? In the system exposited in K&L it too is a variable of the right type to take x as argument. So since y is of category A/B, its meaning is a variable of type <a,b> - that a variable over functions from a (the type of meaning that expressions of category A have) to a (the type of meaning that Bs) have. That variable (which varies depending on the assignment function) takes the meaning of x (of type a) as argument, and returns something of type b. But what it returns depends on each assignment. The open variable is bound at the next step which is Slash Introduction, and which performs lambda-abstraction to yield something which has the same value on all assignments (that differ only on the value assigned to the relevant variable of type <a,b>). In the case of, proving that some individual denoting NP also has the 'lifted' category S/(S/NP), the hypothetical expression is a variable (call it P) of type <e,t> which takes an individual as argument. Then the step of Slash Introduction λ-abstracts over that variable. The meaning, then, of 'lifted' Porky, say, is λP[P(Porky')] exactly as is standard. (Note, though, that K&L treat generalized quantifiers as involving a vertical directionless slash but that is a separate matter; see below. The semantics is the same as what is exposited here.) Again the syntax seems reasonably simple but the semantics does not.

A1.5. A variable-free version?

But we leave the discussion of the semantics with a potentially rosier outlook for the TLG strategy of relying on hypothetical reasoning (and Slash Introduction). For it is not at all clear that the use of variables (and concomitantly, lambda abstraction) is in any way a necessary part of the of the semantic computation. We know that systems using variables can always be translated into a variable-free version, and such a move can also be done here. By variable-free we mean a theory with no use of assignment functions as part of the semantic machinery; see, e.g., Jacobson 1999 for full exposition. Variables can and often are still used in the notation for writing meanings but they are always bound in that notation - and that fact means that there is no use made of assignment functions. As noted above, it is possible to remove variables from the notation too if one wishes using for example the system of Curry and Feys (1958). We do not do this here (or in any of this author's work on variable-free semantics) for the simple reason is that it is not reader-friendly, at least not for most of us.

Consider, then, the natural semantics to associate with hypothetical reasoning in a variable-free setting. The syntax hypothesizes some string x in some position and considers the result when it combines with neighboring material. The semantics should be exactly parallel. Call the meaning of any such string x'. Then, for each such x the meaning of the resulting bigger expression (containing the relevant neighboring string and the hypothetical [x]) is just what one would get by the normal combining of the meaning of the neighboring string with x'. While our exposition has used a variable this again is just for convenience: the meaning of the result is a function mapping the meaning of each hypothesized string to the value one would have gotten were it present.

We can illustrate with the hypothetical reasoning that gives the effect of Lift; we will choose just one of the directional slashes, but the mirror image is the same. The syntax, as noted

above, takes a string x of category A and hypothesizes a string y of category $B/_{L}A$ to the right of x, giving the string x[y]. The semantics is as expected. Since any string of category B/A has as its meaning a function in <a,b>, the meaning of the resultant (hypothetical) string is a function from any f in <a,b> to f(x'). This of course is exactly the (variable-free) semantics of Lift. Once again it is useful to illustrate with the familiar case of 'lifting' NP meanings to meanings of type <<e,t>,t>. As illustrated above, take the NP *Porky* which denotes an individual call it Porky'. Then we posit an $S/_{L}NP$ to the right (or the mirror image of all of this). The meaning of the whole hypothetical expression then is a function from <e,t> meanings to t meanings; using P as a placeholder for any function of type <e,t>, the meaning is the familiar generalized quantifier λP[P(Porky')]. This is exactly as desired. There is one more step needed to complete the account: determining the semantics associated with Slash Introduction. That step has no semantic effect; the meaning of the expression of category S/(S/NP) is that which we got via the hypothetical reasoning step, so Slash Introduction has no semantic effect.

The analogue of Function Composition will work similarly. Going back to our earlier example, we took a string x of category $A/_{R}B$ followed by a y of category $B/_{R}C$ and - with the use of hypothetical reasoning and Slash Introduction - proved the string x-y to be of category $A/_{R}B$. Here too the semantics can be done variable-free. When hypothesizing a string z with category C to the right of x-y, the variable-free way of thinking says that the meaning is a function any object in the set of C-type meanings to whatever one would have gotten by applying the meaning of y to that object and taking the result as argument of the function denoted by x. That is exactly the semantics of function composition. Withdrawing the hypothesis by Slash Introduction keeps that meaning intact, so the meaning of the string x-y of category $A/_{R}C$ is the function λc[x'(y'(c))] (for c any object in the set of things denoted by C-expressions).

In the end then, let us compare my 'hypothetical' TLG (which uses a variable-free semantics and which, so far, has only directional slashes) with the fragment of a theory presented in this paper. The difference is no longer one of the machinery used in the semantic computation, but rather whether or not to stipulate combinatory rules like Lift and Function Composition (or, really Geach) or whether to instead adopt hypothetical reasoning plus Slash Introduction. The tradeoffs between these two approaches can only be determined by examining much bigger fragments, as well as ultimately by determining which strategy is more likely to be closer to actual human processing and production of language. Both of these projects are obviously beyond the current scope, so we leave this as territory to explore. We will mention that Jacobson (2014, 2025) capitalizes on having Lift as an operation in the grammar by using the definition of that operation as a way to simplify the categories for, e.g., generalized quantifiers, so having this has an additional payoff. That, however, is not really relevant to the fragment in K&L as they treat generalized quantifiers rather differently by use of their non-directional slash, and we turn now to that.

A1.6. Vertical Slashes

As noted at the outset of this Appendix, one of the main commonalities between the approach here and much work within TLG is agreement on the fact that 'wh-extraction' type cases do not involve the ordinary slash of CG but something else. I have notated this with | (K&L and other TLG works notate the 'extraction' slash as ↑.) There are a few differences worth noting between their use of this non-directional slash and the use here. But we can note at the outset that these differences are probably not deeply driven by fundamental theoretical considerations, and modifications could be made to either system to make the two look closer.

First, we have given a fully 'traceless' account of extraction sites in Sec. 5.1; there the vertical slash is sanctioned by a rule mapping any A/B to an A|B (this is akin to the account of the 'bottom' of an extraction chain in Gazdar, Klein, Pullum, and Sag (1982,1984)). But as noted there one could instead have a version with something akin to traces, by having | 'pass' in the way done here but - rather than having the above mapping - have a silent 'proform' X|X for any category X. (This is similar to what is actually done in GKPS (1985).). The semantics could be exactly the semantics Jacobson (1999) uses for proforms in general (and hence for lexical items of category X^X); each such item denotes the identity function over the appropriate type. Thus there is nothing deeply rooted either in the program of GPSG or of the program here that forces a decision for or against silent proforms (traces) in extraction sites; we have adopted for the 'traceless' version simply because it seems simpler, but this could all easily be revised.

K&L, on the other hand, crucially make use of conventions governing their vertical slash which involves an actual placeholder in the string of the relevant category. This is at least in part driven by their account of quantification. Thus, along with Montague (1973), they adopt something like a Quantifying In rule, which allows a generalized quantifier to be substituted into a position marked by such a place holder. If the treatment of, e..g, generalized quantifiers in object position or quantifier scopes requires a type of Quantifying In rule, then indeed there has to be some placeholder in a string to specify where the material goes. (Montague made use of an indexed pronoun for this; K&L have instead a placeholder in the string.)

The paper here (and the related paper in Jacobson (2025)) says nothing at all about the treatment of quantification. But Jacobson (2014) advocates instead the approach in Hendriks (1993) whereby verbs (and other expressions) taking e-type arguments can shift their meanings

to take <<e,t>,t> arguments, along with a corresponding syntax. (This strategy is simply a generalization of the strategy in Partee and Rooth (1985) which in turn is inspired by the lexical meaning assigned to transitive verbs by Montague (1973).) So in part the full treatment of the conventions involving the non-directional slash will depend on the best way to treat quantification in general which is obviously a topic far beyond the current scope.

I will mention, however, that should a Quantifying In type of approach prove to be empirically motivated over the Hendriks type of approach, that could also be accommodated in the general framework here. Recall that we have discussed at various points the possibility of an infixation slash, whereby it is incorrect to say that there is no internal structure to a linguistic string but that it has a dedicated infixation point. That infixation point is akin to any other type of placeholder within a string that one chooses. For starters, then, one could treat generalized quantifiers as infixes, where their category (or at least one of their categories) would be $S/_C (S/_I NP)$ (for C a 'circumfix slash, so that this category says give me as a circumfix something that wants an NP as an infix). (The idea spelled out in Jacobson (2025) that their category is Lift(NP) automatically gives this as one possibility.)

That alone, though, is not enough to accommodate all cases of Quantifying In as just using as single infixation point cannot extend to cases of multiple instances of Quantifying In. We would need, then, to extend the system in some way as to have multiple infixation points, and this might look quite different from a system with a simple infixation operation. Obviously this will not be done here. But the bigger point is that it is not clear that the treatment of vertical slashes for the case of quantification represents a deep difference between TLG (or at least the type of TLG in K&L) and the type of Categorial Grammar here as much as it does a difference between available analyses of empirical phenomena.

Appendix 2: "Ungrammatical but Acceptable"?

A referee expresses discomfort with the judgments on many of the examples here - including the Lakoff chain examples - and suggests that perhaps these are truly ungrammatical but nonetheless 'acceptable'. The intuition here (I believe) is that some of these seem good at first glance but - at least for the referee in question - once one thinks about them they appear to be a bit off. (As will be noted below, there are certainly many speakers - including this author - who find these quite fine, but I have found a few other speakers express uncertainty about some of the judgments.)

We are all familiar with examples of this general sort. In some cases their oddity is rather different from what I take to be the referee's discomfort here; a classic case going under the rubric of *syntactic illusions* (mentioned perhaps first in Montalbetti 1984) is the following:

(36) More people have been to Berlin than I have.

This sounds perfect on first hearing it, yet one cannot compute any meaning from this. But this kind of phenomenon is quite different from the examples here in that, as far as I can tell, the meaning of the Lakoff chain examples (and others involving apparent violations of the CSC) is totally clear. Moreover, speakers are quite uniform in their judgments that (36) makes no sense, while speakers seem to vary on the judgments of the CSC violating sentences of the types here. As mentioned above, many speakers find these fine (aside from judgments that I have collected in an admittedly informal way across a variety of speakers and settings) these types of cases have been well vetted in the literature. Such examples are found beginning, of course, with Lakoff 1986 but also discussed in Zwart 2005, Chaves, 2012, among others.

Given that there are some speakers who express uncertainty about some of the Lakoff chain types of examples, should we conclude - with the referee - that these are actually ungrammatical but acceptable? I will argue here that this would be explanatory only if accompanied by a hypothesis

as to why they sound acceptable. Equally critical is to provide an explanation for the robust contrasts exemplified by, e.g., (28) (the bad RNR cases) and (29) (the good parallel *wh* extraction cases). Quite crucially, these contrasts do fall out from the apparatus here which predicts that (29) is good. So if they are actually ungrammatical but acceptable, we need to understand why the the same is not true of (28).

Consider the reverse situation: cases which sound bad but where there is general agreement that this is for extragrammatical reasons. Take the classic case of multiple center embeddings famously introduced in Chomsky (1965):

(37) The mouse that the cat that the dog chased caught squeaked.

While my discussion here is not an exact rendering of the way Chomsky put it, we can use this to show the logic of Chomsky's ultimate conclusion (that the badness was not due to the grammar). First, it is uncontroversial that this sounds bad. But we do not *a priori* know why something sounds bad - we don't have primary intuitions about whether something is 'ungrammatical' or something else is going on. It could bad for reasons having to do with the grammar, or because of a variety of other facts such as processing difficulties, pragmatic problems such as violating the norms of cooperative conversation, etc. In this case we conclude that its explanation lies not in the formulation of the grammar for two reasons. First, the rules needed to allow the simplest cases of relative clauses (those without 3 levels of embedding) would just happen to predict that (37) is well-formed as those rules involve recursion. But there is also a second prong: here we can make a pretty good guess as to why this is bad based on processing difficulties, and there are processing models designed to explain just this.

Hence, if we wish to deem something ungrammatical but nonetheless acceptable, we should probably apply the same methodology. What this would mean first, is that we would need a good

reason to believe that the simplest hypothesis about how the grammar works (compatible with the clear cases) would predict these to be bad. And second, crucially, we would want some explanation - or at least the beginnings of one - as to some extra grammatical fact that is at play to explain why speakers nonetheless accept these. Consider how these desiderata would apply to the CSC violating examples of relevance here. As to the first, I (along with others who have rejected a syntactic CSC) have actually argued that the simplest account of extraction (and function composition) actually does not predict the CSC and allows extraction from (and function composition) into a conjunct. (Hence the line of reasoning taken in many works that reject the CSC is that the bad examples are due to facts about information structure/discourse coherence and not to the extraction processes themselves. See, e.g., Lakoff (1986), Kehler (1996) and many others.) But perhaps one could counter this by saying that there are so many bad instances involving so-called CSC violations that one needs to adjust the grammar. So let us leave open the question of what is the simplest grammar compatible with the clear facts (since what is at issue is the domain of the clear facts), and turn to the second prong. Were we to decide that it is simplest to have something which gives the effect of the CSC we would need at least a hint of an explanation as to why speakers nonetheless find some cases 'acceptable'.

Sometimes cases which seem good but some theory predicts are bad are explained by a sort of 'repair' mechanism. But one needs to say just how it is that the cases are repaired and why that mechanism is available in some cases and not in others. The referee gives us a suggestion in the particular case of (38) (from the Christmas carol *What Child is This*), and suggests that the repair is that we supply *about* after *sing* (to make it a case of ATB extraction):

(38) This, this is Christ, the King, whom shepherds guard and angels sing.

I question whether that is what this has to mean, and find it just as good (modulo a bad rhyme scheme) to have *whom shepherds guard and angels dance*. But this is a subjective judgment, and I assume the referee would not accept the *dance* example. In any case, how would that suggestion extend to a case like *How much beer can he drink and still stay sober?* (discussed in a lot of literature) or *How many banks can Clyde rob and Bonnie still not get upset?* Perhaps in the first we supply *after drinking* (again making it an ATB case) and in the second we supply *about his robbing* (again making this an ATB case). But why is this type of strategy not always available?

In particular, it is not clear how this can account for the crucial contrasts here - e.g., that RNR requires a gap in the rightmost conjunct in a Lakoff chain while *wh* extraction does not. Take the contrasts again between the good examples in (29a-b) and the bad ones in (28a-b). A repair strategy for (29) could posit that the listener supplies *after drinking* after *fall asleep* in (a) and (b), making these ordinary ATB extraction (i.e., conjunction of like categories). But then why is this exact repair strategy not available in (28)? It should ameliorate these in exactly the same way.

In conclusion, it seems that the idea that the CSC violating sentences (including Lakoff chains) are truly ungrammatical but somehow get repaired would need to be supplemented with an explanation of how the 'bad' cases are repaired, and why that mechanism happens only selectively. At the moment, then, I think the evidence is in favor of allowing the grammar to predict all such cases well-formed, and relying on constraints on information packaging and/or discourse structure to rule out the bad sounding cases.

Additional References

Chaves, Rui. 2012. On the grammar of Extraction and Coordination. *Natural Language and Linguistic Theory* 30(2), 465-512.

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax.* Cambridge: MIT Press.

Curry, Haskell B. and Robert Feys. 1958. *Combinatory Logic.* Amsterdam: North-Holland Publishing Co.

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

Jacobson, Pauline. 1990. Raising as Function Composition. *Linguistics and Philosophy* 13(4), 423-475.

Lambek, Joachim. 1958. The Mathematics of Sentence Structure. *The American Mathematical Monthly* 65(3), 154-170.

Shapiro, Stewart and Teresa Kouri Kissel (2022). Classical Logic. In Edward Zalta and Uri Nodelman (eds.), *Stanford Encylopedia of Philosophy.* https://plato.stanford.edu/entries/logic-classical/

Zwart, Jan-Wouter. 2005. The coordinate structure constraint: A minimalist perspective. Paper presented at the Groningen Workshop on Verb Clusters and Coordination. Leiden. https://let.rug.nl/~zwart/docs/ho05devos.pdf