Glossa
a journal of general linguistics

Cyclic scope and processing difficulty in a Minimalist parser

RESEARCH

ROBERT PASTERNAK
THOMAS GRAF
Author affiliations can be found in the back matter of this article

ubiquity press

ABSTRACT

A common view in the theoretical literature is that quantifier raising (QR) is a clause-bounded operation. But in a paper published in *Glossa*, Wurmbrand (2018) argues that (I) QR is not clause-bounded, and the apparent clause-boundedness of QR is due to the human parser's difficulty in processing extraclausal QR; and (II) the relative difficulty of extraclausal QR depends on the size of the embedded clause from which QR takes place. She then proposes a theory of scope processing in which parsing Logical Form (LF) movement is costly for the human parser, which in conjunction with independently motivated assumptions about A′-movement generates the desired results. In this paper, we accept Wurmbrand's descriptive observations and proposed syntax but offer an alternative, rigorously defined metric of scope processing difficulty that makes precise quantitative predictions. Our proposal is formalized with Minimalist Grammars (Stabler 1997) and expands recent work by Kobele et al. (2013), among others, that uses this formalism to account for numerous processing phenomena. Our metric correctly handles Wurmbrand's observations as well as cases that are problematic for her account, and points the way toward an explanatory theory of scope processing.

CORRESPONDING AUTHOR:
Robert Pasternak
Leibniz-Center for General Linguistics (ZAS), Schützenstraße 18 10117 Berlin, Germany
mail@robertpasternak.com

KEYWORDS:
quantification; scope; scope processing; formal parsing; Minimalist Grammars

TO CITE THIS ARTICLE:
Pasternak, Robert and Thomas Graf. 2021. Cyclic scope and processing difficulty in a Minimalist parser. *Glossa: a journal of general linguistics* 6(1): 8. 1–34. DOI: *https://doi.org/10.5334/gjgl.1209*

DOI: 10.5334/gjgl.1209

1 INTRODUCTION

When presented with a sentence that seems ungrammatical or lacks an otherwise expected reading, linguists face the question of whether this is due to competence or performance (Chomsky 1965). Perhaps the most famous case where the answer seems to be the latter is center-embedding (Chomsky & Miller 1963). Compared to the right-embedding structure in (1a), the semantically identical center-embedding structure in (1b) is commonly considered unacceptable.

(1) a. The cheese attracted the mouse that was eaten by the cat that was bitten by the dog that was owned by the barber.
b. The cheese attracted the mouse that the cat that the dog that the barber owned bit ate.

Yet for various reasons the broad consensus is that the grammar of English does not litigate against center-embedding; rather, the human parser is overwhelmed by the difficulty of the processing task.

A growing body of work asks the same question about quantifier raising (QR): is the purported clause-boundedness of QR due to competence or performance? Many quantifiers seem unable to take scope outside of the clause in which they are merged. Consider (2), due to Fox (2000: p. 62):

(2) a. Someone said [$_{CP}$ that every man is married to Sue].
b. Someone said [$_{CP}$ that Sue is married to every man].

Each sentence in (2) only allows for a reading in which some person claims that Sue is polygamous (∃ > ∀), and not one in which different people have made opposing claims about who Sue is married to (∀ > ∃). The absence of this second reading suggests that *every man* cannot undergo QR to a position outside its embedded CP, where it can scope over *someone* in the matrix clause. This inability of QR to cross clause boundaries is surprising because other instances of A′-movement, e.g. *wh*-movement, are not so constrained (*Who$_i$ did someone say that Sue is married to t_i?*).

But in a paper published in *Glossa*, Wurmbrand (2018) argues that (I) QR is not clause-bounded after all; (II) extraclausal QR is, however, more difficult to process than within-clause QR (cf. Syrett & Lidz 2011; Syrett 2015a; b); and (III) the relative difficulty of processing extraclausal QR correlates with the size of the embedded clause from which QR takes place. To account for (III), Wurmbrand proposes that each movement step that impacts Logical Form (LF) comes with a processing cost, and scoping out of larger embedded clauses is more costly because it involves more iterations of LF-movement.

While Wurmbrand's proposal is notable for making concrete, formally defined, empirically testable claims that build on independently motivated principles of syntactic theory, it struggles to account for some data points. Due to the exclusive focus on the number of movement steps that affect LF, Wurmbrand's notion of scope processing difficulty does not consider how Phonetic Form (PF) is affected by movement. But there is evidence that the latter matters, too. As Wurmbrand herself notes, overt cyclic *wh*-movement is much more easily processed than cyclic QR even though in both cases operators are semantically interpreted at great distances from their merge positions. This contrast between *wh*-movement and QR indicates that movement that impacts LF is noticeably easier to parse when it also impacts PF: overt scopal movement is more readily processed than covert movement. In addition, experimental results from Lee (2009) suggest that when a quantificational DP undergoes overt movement, it is more easily interpreted in its post-movement position (surface scope) than in its pre-movement position (inverse scope), even though the latter would involve fewer movement steps that affect LF. It thus appears that the comparative difficulty of computing a given scope configuration does not depend exclusively on the relative complexity of the LF side of the derivation or representation, as in Wurmbrand's analysis. Instead, processing difficulty increases due to mismatches between PF and LF, and consequently movement is processed more easily when it affects both PF and LF than when it affects only LF (QR) or only PF (reconstruction).

Wurmbrand herself makes this observation at the end of her paper, but she does not formalize this idea to the degree that is needed for strong empirical predictions—our paper closes this

gap. We offer a novel theory of scope processing difficulty that is very much in the spirit of Wurmbrand's insight about LF-PF mismatches. This theory builds directly on Minimalist Grammars (MGs, Stabler 1997), a computational formalization of Minimalism (Chomsky 1995). More precisely, we adopt a framework pioneered by Kobele et al. (2013) that derives processing predictions from the memory usage of the top-down MG parser introduced by Stabler (2013). This rigorous computational foundation not only allows us to precisely quantify processing difficulty and derive strong processing predictions, but also illuminates why LF-PF mismatches are costly for the human parser and situates this insight within a broader theory of how the human parser's memory usage affects sentence processing.

The framework of Kobele et al. (2013) has previously been used to account for a variety of syntactic processing phenomena attested in the psycholinguistic literature, including: center- vs. right-embedding (Kobele et al. 2013; Gerth 2015); crossing dependencies vs. nested dependencies (Kobele et al. 2013); subject vs. object relative clauses in English and East Asian languages (Graf et al. 2015; 2017); stacked relative clauses in English and Chinese (Zhang 2017); Heavy NP Shift (Liu 2018); dative DP attachment ambiguities in Korean (Lee 2019); a variety of word order and relative clause processing facts in Italian (De Santo 2019); and the emergence of gradience in syntax and the role of syntactic priming (De Santo 2020). As can be gleaned from this list, all previous work has focused on cases where the sentences being compared differ in their overt structure. Our paper marks the first attempt to extend this productive research program to cases where the detectable differences lie only in the comparative (un)availability of certain semantic interpretations.

The paper is laid out as follows. We start with a cursory overview of Wurmbrand 2018 and the data that motivate her proposal (Section 2.1). We also discuss two phenomena that are problematic for her approach (Section 2.2) but could be accounted for if a sentence's processing difficulty depends on how much its PF and LF differ from each other. In order to hash out this intuitive notion, we introduce the MG processing model in Section 3. We subsequently extend it with LF-movement (Section 4.1) and define **Location Differential (LD)** as a means of quantifying PF-LF mismatches (Section 4.2). Once the machinery is in place, it is straightforward to show that it accounts for all the scope phenomena discussed in Section 2 (Sections 4.3–4.6).

2 CYCLIC QR AND WURMBRAND'S (2018) ANALYSIS

Our paper is a direct follow-up to Wurmbrand's (2018), and we adopt many of her assumptions and data points. We first briefly discuss data suggesting that QR is less limited than commonly believed, as well as how Wurmbrand relates the acceptability differences between various types of QR to the complexity of the involved syntactic structures (Section 2.1). We then point out two particular problems with her account that suggest that the relevant factor is actually how much LF and PF differ from each other (Section 2.2), an intuitive idea that will be fully developed in Sections 3 and 4.

2.1 WURMBRAND'S PROPOSAL

Wurmbrand (2018) cites a wealth of evidence from the theoretical and experimental literature in favor of her claims about the unboundedness and relative processing difficulty of QR. We will not go through the details of this evidence here, and will only discuss the highlights; readers interested in more in-depth discussion of this evidence are referred to Wurmbrand 2018 and sources therein.

As for the claim that QR is in principle unbounded, a variety of evidence has been offered suggesting that quantifiers can scope out of both non-finite and finite embedded clauses, including over matrix-clause subjects. Larson & May (1990) and Kennedy (1997) provide examples with non-finite clauses, such as (3) from Kennedy (1997: p. 674):

(3) a. At least two American tour groups expect [to visit every European country this year].
b. Some agency intends [to send aid to every Bosnian city this year].
c. At least four recreational vehicles tried [to stop at most AAA approved campsites this year].
d. Some congressional aide asked [to see every report].

DOI: 10.5334/gjgl.1209

Each of these examples has a reading in which the embedded quantifier scopes over the matrix subject: in (3a), for example, tour groups can covary with countries. The ready availability of extra-clausal QR from non-finite clauses has also been supported by experimental work from Moulton (2007), Syrett & Lidz (2011), and Hackl et al. (2012), among others.

Similarly, while examples like (2) have led many researchers to conclude that QR from finite clauses is ungrammatical, a growing body of work suggests that this is not the case. An example from the theoretical literature, due to Farkas & Giannakidou (1996: p. 36), can be seen in (4):

(4) A student made sure [that every invited speaker had a ride].

(4) permits a reading in which different students find rides for different speakers (∀ > ∃). While Farkas & Giannakidou argue that scoping out of finite embedded clauses is constrained by the choice of embedding predicate, experimental evidence from Syrett & Lidz (2011), Syrett (2015a; b), and Tanaka (2015a; b) suggests that inverse scope readings are available with a wide variety of finite-clause-embedding verbs, and are not as restricted as Farkas & Giannakidou suggest.

However, scoping out of embedded clauses seems to impose an extra cost, as many of the same experiments showing the possibility of embedded quantifiers scoping over matrix subjects also show that this is more difficult to process than objects scoping over subjects within the same clause (e.g. in *A technician inspected every plane*). Wurmbrand (2018) attempts to explain this observation by positing that a sentence's processing difficulty increases with each movement step that affects LF, i.e. each movement step that alters the position from which a phrase takes scope. More precisely, she adopts the copy theory of movement (Chomsky 1995) in conjunction with Fox's (2002, 2003) operation of *trace conversion*, in which lower copies of quantifiers are altered at LF in order to generate trace-like bound variable interpretations. Syntactic trace conversion is not universally adopted by semanticists—see e.g. Ruys 2015, Pasternak 2020 for alternatives—but Wurmbrand's basic idea could be rehashed in many different ways and isn't inextricably tied to this mechanism. But trace conversion provides an intuitively pleasing picture: if trace conversion is a costly operation, then every movement step that impacts LF increases processing cost by adding another instance of trace conversion. In essence, there is a "tax" on LF traces.

If each LF trace imposes a processing cost, then in conjunction with common assumptions about A′-movement, the additional cost of extraclausal QR falls out immediately. Suppose that in a monoclausal example like *A technician inspected every plane*, some *n*-many QR steps are required in order for *every plane* to scope over *a technician*. In a sentence like (4), meanwhile, it is generally thought that any mover that escapes the embedded clause must first stop at its edge—CP is a *movement domain*—before undergoing the *n* (or more) instances of movement required to generate inverse scope. Thus, generating inverse scope for (4) will require more scope-taking movement steps—and thus, more costly LF traces—than in the monoclausal case.

In fact, experimental evidence tentatively suggests the existence of finer-grained distinctions in inverse scope processing difficulty. In previous work, Wurmbrand (2001; 2014a; b; 2015) argues that verbs taking non-finite clausal complements can be divided into three classes that take increasingly large functional complexes as their arguments. First, *try*-type infinitives have a very reduced structure that stops at the *v*P-level.

(5) Becca tried [$_{vP}$ to go to Boston].

The next larger class consists of "future-shifting" verbs like *decide*, whose complements must be headed by some functional head related to tense, aspect, or modality (TAM). Wurmbrand argues that this involves at least some future-shifting modal head WOLL.

(6) Becca decided [$_{\text{WOLL}P}$ WOLL [$_{vP}$ to go to Boston]].

Finally, *claim*-type infinitives display the full CP structure:

(7) Becca claimed [$_{CP}$ C … [$_{TP}$ T … [$_{vP}$ to be in Boston]]].

This distinction is relevant because evidence from Moulton (2007) suggests that inverse scope is harder with *decide*-type infinitives than *try*-type infinitives. In other words, inverse scope is easier for monoclausal (8a) than for *try*-type (8b), which is easier than for *decide*-type (8c).

DOI: 10.5334/gjgl.1209

(8) a. A technician inspected every plane.
b. A technician tried to inspect every plane.
c. A technician decided to inspect every plane.

In addition to these between-sentence observations, there are the more obvious within-sentence observations: for each sentence in (8), inverse scope is harder to process than surface scope. This is shown for the embedded clause cases in many of the studies discussed previously, and for monoclausal cases by Kurtzman & MacDonald (1993), Tunstall (1998), Anderson (2004), and others. It is worth noting that the between-sentence observations about embedded quantifiers are limited, both in the strength of their evidence—Moulton's experimental evidence is compelling but does not quite reach statistical significance—and in their breadth: where finite clauses and *claim*-type infinitives fit into the picture in relation to each other and to *decide*-type infinitives is not established. With this latter point in mind, we will follow Wurmbrand in mostly sticking to (8), though it is worth noting that both Wurmbrand's account and ours predict QR out of finite clauses to be at least as hard as QR out of both *claim*-type and *decide*-type infinitives—an intuitively plausible prediction, but one that to our knowledge remains to be verified experimentally.

Wurmbrand notes that the between-sentence observations in (8) can be accounted for on her theory of scope processing difficulty, given certain reasonable assumptions about movement domains. More specifically, suppose that the set of movement domains includes not only *v*P—a commonly held view since the development of phase theory by Chomsky (2000; 2001)—but also WOLLP, at least when it is the complement of the embedding verb. In this case, if in (8a) *n*-many steps of QR are required to generate inverse scope, then in (8b) at least *n* + 1-many steps are required, since *every plane* must also move to the edge of the embedded *v*P. In (8c), meanwhile, this increases to *n* + 2: the embedded object must move not only to the edge of the embedded *v*P, but also to the edge of embedded WOLLP before moving to scope over the subject.

In summary, Wurmbrand (2018) concludes that (I) QR is in principle unbounded, (II) extraclausal QR is costlier than within-clause QR, and (III) extraclausal QR is costlier for *decide*-type than for *try*-type infinitives; we follow her in adopting these conclusions. She then proposes that there is a "tax" on LF traces: the more LF-impacting movements are required, the costlier an interpretation is. This conjoined with independently motivated syntactic assumptions makes the right predictions for (8). Next we turn to data that are problematic for Wurmbrand's analysis.

2.2 THE PROBLEMATIC IMPACT OF OVERT MOVEMENT

Since Wurmbrand's analysis connects processing difficulty to the number of LF traces, the best counterevidence comes from cases where LF-relevant movement does not appear to be a particularly costly operation, or even better, cases where a parse involving more LF traces is *easier* than a parse involving fewer. We present two cases here: *wh*-movement (already mentioned by Wurmbrand), and the scope of subjects and objects with respect to negation in English and Korean (Lee 2009, not previously discussed in connection with Wurmbrand's proposal).

Wh-movement poses an obvious challenge, one that is acknowledged by Wurmbrand herself. Consider, for example, the sentences in (9):

(9) Wurmbrand 2018: p. 25, based on her (29)
a. What did a technician say that John inspected?
b. A technician said that John inspected every plane.

In (9a), *what* moves cyclically from the complement of *inspect* to some specifier in the left periphery of the matrix clause. Assuming that *wh*-phrases that overtly move to the left periphery also take scope there (Karttunen 1977 and much work since), we can infer that each of these movement operations also leaves a trace at LF. (9a) therefore requires at least as many iterations of scope-taking movement as an inverse scope interpretation of (9b)—possibly more, since the latter may not require *every plane* to move all the way to the left periphery. Thus, Wurmbrand predicts (9a) to be at least as difficult to process, if not more so, than an inverse scope interpretation of (9b). While we are not aware of any experimental studies

DOI: 10.5334/gjgl.1209

directly addressing this question, at least on an intuitive level things appear to be quite the opposite: (9a) is noticeably *easier* to process than the inverse scope of (9b). Of course, relying on intuitions to make judgments of processing difficulty is questionable to say the least, but pending much-needed experimental investigation comparing and contrasting such cases, we maintain that a model that predicts (9a) to be easier than the inverse scope of (9b) is *prima facie* more likely to be correct than one that predicts the opposite.

Another issue for Wurmbrand's theory—one that has gone undiscussed so far, and that has much stronger experimental evidence to back it up—pertains to the relative scope of sentential negation and universal quantifiers, as explored in detail by Lee (2009). Lee tests such scope preferences in both English and Korean for universal quantifiers in both subject and object positions.[1] Some relevant examples in both languages can be seen in (10) and (11).

(10) ***Every* in subject position**

a. According to the story, every kid didn't feed the doves in the park. (Lee 2009: p. 93)

b. *Korean* (Lee 2009: p. 79)
hwacangsil-eyse motun haksayng-i son-ul an ssis-ess-ta-ko
in the restroom every student-NOM hand-ACC NEG wash-PST-DECL-COMP
iyaki-un malhaycwunta
story-TOP tell
'The story tells that every student did not wash her hands in the restroom.'

(11) ***Every* in object position**

a. According to the story, Cindy didn't light every candle last night. (Lee 2009: p. 124)

b. *Korean* (Lee 2009: p. 112)
ecey pam-ey Sehee-ka motun chospwul-ul an khye-(e)ss-ta-ko iyaki-nun
last night Sehee-NOM every candle-ACC NEG light-PST-DECL-COMP story-TOP
malhaycwunta
tell
'The story tells that Sehee did not light every candle last night.'

The Korean examples above utilize what is known as the *short-form* negation. Korean also has a *long-form* negation, and Lee's findings for this form mirror those for short form: subjects and objects both tend to scope over negation. In English, subjects also tend to scope over negation, but the scope of objects remains under negation. This curiously lines up with major word order differences between the two languages: whereas the default order for English is subject-negation-object, it is subject-object-negation in Korean. Hence Lee's basic finding is a preference in both English and Korean for surface scope over inverse scope in these constructions.

Lee's data are unexpected under Wurmbrand's account. Let us consider the case of subjects first. If subjects are base-merged in Spec,*v*P, which presumably is structurally lower than negation, then there should be a preference for subjects to scope under negation. It is true that the subject moves to a position above negation later on (Spec,TP), but this step is costly because it requires trace conversion at LF. A less costly option would be to treat the subject movement as only affecting PF, keeping the subject in Spec,*v*P at LF and thereby avoiding the need for trace conversion. The very same argument applies to Korean objects: even though the object moves to a position above negation, it would be less costly if this movement only affected PF. Only English objects behave the way that is predicted by Wurmbrand, but this is because they presumably do not overtly move at all. Once movement occurs, pure PF-movement should be preferred, assuming that this is an available option. In any system where it is possible for movement to affect PF but not LF, Wurmbrand's account incorrectly predicts a preference for inverse scope in Lee's examples because pure PF-movement avoids the need for trace conversion at LF.

One could resolve Lee's contradictory findings by stipulating that pure PF-movement is simply unavailable in these constructions, but this would still leave the issue of *wh*-movement.

1 Lee also tests native Korean-speaking L2 speakers of English. We will not account for these results in this paper as it is not sufficiently clear what a Korean L2 grammar (let alone parser) of English looks like. However, the approach we advocate for could certainly be extended in this direction.

A unified solution is actually available in the form of a different scope processing metric that is briefly mentioned by Wurmbrand (2018: fns. 20, 25): processing difficulty doesn't depend (only) on the number of trace conversion steps, but on how much LF and PF differ from each other. Under this metric, *wh*-movement is easy to process because it affects both LF and PF in the same way. The scope data in Lee (2009) is expected because it is cheaper for movement to affect both LF and PF rather than just the latter, so that subjects (and Korean objects) also end up in positions above negation at LF, not just at PF. The data discussed earlier in Sec. 2.1 would still be accounted for as it involves pure LF-movement, which alters LF without affecting PF and thus also increases their dissimilarity. Hence it apparently isn't just any kind of movement that affects LF, but only movement that affects LF without also affecting PF.

This, in a nutshell, is the approach we will adopt in the rest of this paper. Such an approach requires answering two questions. First, how do we formally define a scalar notion of PF-LF mismatch with a sufficient degree of predictive power? And second, what is it about the nature of the human parser that entails that this particular property of syntactic representations (or derivations) should lead to processing difficulty? In order to bring us closer to an explanatory theory of scope processing difficulty, we will approach these questions from a formal parsing perspective: we will provide a formal grammar that generates both PF and LF representations, along with a formal parser for that grammar, and we will show that scope processing difficulty correlates with a particular type of information storage arising during the course of a successful parse. The formal grammar utilized will be a version of Stabler's (1997) Minimalist Grammars (MGs); as discussed in the introduction, MG parsing has been used to account for a variety of syntactic processing phenomena, and our theory serves as a significant extension of this already productive research program.

3 MINIMALIST GRAMMARS AND TOP-DOWN PARSING

MGs were developed by Stabler (1997) in order to study Minimalist syntax from a computational perspective. Over the years, numerous aspects of the formalism have been explored in great depth (see Stabler 2011a), including parsing. The work on MG parsing will allow us to put the "LF-PF mismatch" proposal on a more rigorous foundation, and in doing so we will also be able to account for the problematic cases of *wh*-movement and the surface scope preference discussed in Section 2.2. Specifically, we adopt a program initiated by Kobele et al. (2013), which directly relates the processing difficulty of any given sentence to how much memory is consumed by an MG top-down parser (Stabler 2013) that has to build the correct syntactic structure for said sentence. Our main innovation, presented later on in Section 4, is the addition of LF-movement to this approach. With the addition of LF-movement, the MG parsing model of sentence processing naturally extends to all the constructions that were discussed in the preceding section.

We will start with the basics of MGs, focusing in particular on the role of derivation trees (Section 3.1). Awareness of the differences between derivation trees and phrase structures is indispensable for understanding how the MG parser in Stabler 2013 can build structures in a top-down fashion (Section 3.2). Once the top-down parser is in place, it is fairly easy to measure how memory is used during the construction of a parse and to relate these measurements to processing difficulty (Section 3.3). Following earlier work on MG processing, our presentation is deliberately framed at a high level of abstraction that omits many of the formal details that have no effect on the specific processing predictions. For the sake of completeness, though, we include a fully worked out formal system in the appendices.

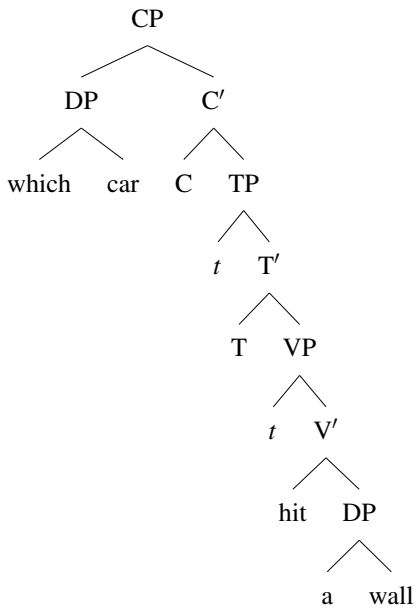
3.1 STANDARD MGs: FEATURES, OPERATIONS, AND DERIVATION TREES

MGs build on two central tenets of Minimalist syntax: syntactic structure is built via operations (Merge/Move), and operations are triggered by features on lexical items.

Suppose we want to build the (simplified) phrase structure tree below for *which car hit a wall*, where *which car* first undergoes subject movement to Spec,TP and then *wh*-movement to Spec,CP:

DOI: 10.5334/gjgl.1209

(12)



This will involve a number of Merge and Move steps, all of which must be triggered by features.

Let us first consider a single Merge step. Just like in Minimalist syntax, the DP *a wall* is built by merging the lexical items *a* and *wall*. In MGs, this specific operation must be triggered by matching features on the two lexical items: *wall* carries a *category feature* N, and *a* carries the corresponding *selector feature* =N. Since *a* carried the selector feature, it acts as the head of the resulting phrase *a wall*. It also carries the category feature D, indicating that this phrase is a DP. Once N and =N have triggered Merge, they are considered checked and no longer participate in any syntactic operations.

The MG literature uses a specialized notation to express the preceding paragraph more succinctly. The phonetic exponent of a lexical item like *wall* is separated from its features by a double colon.

(13) a. wall:: N
b. a:: =N D

Note that the selector feature =N on *a* occurs before the category feature D because *a* cannot act as the head of a DP until it has selected an NP argument via Merge. Rather than unordered feature bundles, then, MGs use feature strings. The features of every lexical item are linearly ordered, and a feature cannot trigger any syntactic operations until all features before it have been checked.

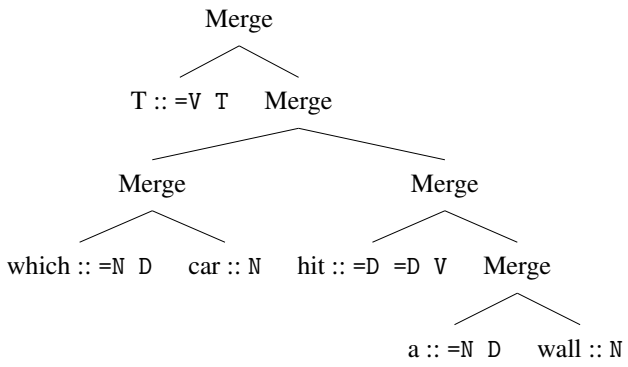
Once *a wall* has been built, the same system of feature-driven Merge allows us to assemble the VP and part of the TP. First, the verb *hit* carries a selector feature =D which is matched by the category feature D on the determiner *a*. Since *a* is the head of *a wall*, the whole DP *a wall* is merged with the verb *hit*. We build the DP *which car* in exactly the same way *a wall* was assembled, and then merge it with *hit*. Note that this requires *hit* to carry a second selector feature =D as the first one was already checked when *hit* merged with *a wall*—in MGs, each feature is a resource that is used up once it has triggered an operation. Finally, the category feature V on *hit* allows it (or rather, the phrase it heads) to be merged with the T-head, which carries the matching selector feature =V. At this point, then, the following syntactic operations have taken place:

(14) a. Merge *a* and *wall*, checking =N on *a* and N on *wall*
b. Merge *hit* and *a wall*, checking =D on *hit* and D on *a*
c. Merge *which* and *car*, checking =N on *which* and N on *car*
d. Merge *hit a wall* and *which car*, checking =D on *hit* and D on *which*
e. Merge T and *which car hit a wall*, checking =V on T and V on *hit*

This can be represented in the more readable format of a *derivation tree*.

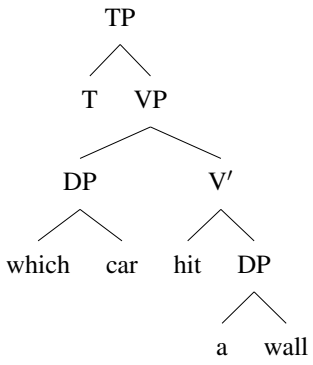
DOI: 10.5334/gjgl.1209

(15)



In a derivation tree, all leaf nodes are lexical items, and all interior nodes are labeled with syntactic operations. In this case, all interior nodes are labeled Merge, and each Merge operation applies to the two subtrees immediately underneath it. Note that if we remove all features and relabel each interior node with an X′-label matching the category feature of the selector, we get the phrase structure tree that would have been built up to this point:

(16)



This shows that derivation trees encode all the information of phrase structure trees—the latter can be fully recovered from the former. For this reason, derivation trees have played a central role in MG research ever since Kobele et al. 2007, effectively replacing phrase structure trees as the primary syntactic representation format. As we will soon see in Section 3.2, derivation trees are also an integral component of the MG parsing approach we adopt.

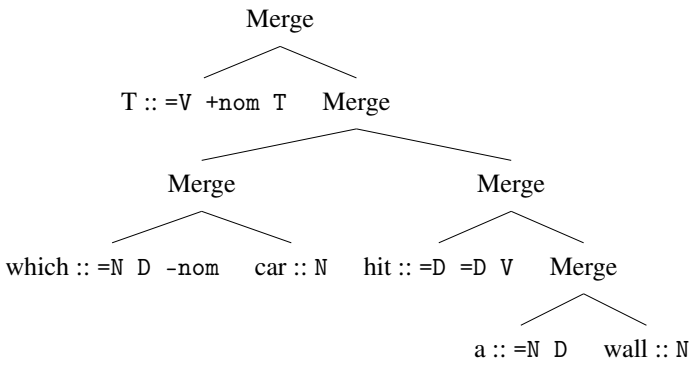
Now that the VP has been selected by the T-head, the next step to take place is movement of the subject *which car* from Spec,VP to Spec,TP. This requires some modifications to what we have done so far. Displacement of subtrees is handled by the operation Move, and just like Merge, Move must be triggered by two matching features. Whereas Merge requires matching category and selector features, Move requires matching *licensee* and *licensor* features. In the case at hand, the T-head must carry a licensor feature, which we may call `+nom`; the feature name is arbitrary, and we do not claim that subject movement is in any way related to checking of nominative case. The head of the mover must carry a matching licensee feature `-nom`. In the case at hand, this head is *which*, the feature string of which we assumed to be `=N D`. This assumption is now revealed to be wrong: *which* must have a feature string that also includes the licensee feature `-nom`. The placement of the licensor and licensee features is relatively fixed: licensor features occur between the selector features and the category feature, and licensee features occur after the category feature.[2]

Based on these observations, we have to make minor adjustments to the feature strings for T and *which*. The T-head now has the feature string `=V +nom T`, while the feature string of *which* is `=N D -nom`. Rather than the pre-movement derivation tree in (15), then, we actually

2 The reasoning behind this feature order is as follows. Licensee features must occur after the category feature because a phrase cannot move anywhere before it has been selected and thus made part of a larger structure that furnishes potential landing sites. The licensor features on a head H must occur after the first selector feature of H as it is impossible for anything to move to H if H hasn't selected a single argument yet. At the same time, all licensor features must occur before the category feature of H—otherwise, a phrase would move to Spec,HP after HP has already been selected by some other head. This would be an instance of countercyclic movement, which is usually considered illicit. Usually this ban against countercyclic movement is generalized so that a head must not select any arguments once it has been targeted by movement. Under this assumption, licensor features must always appear after all selector features.

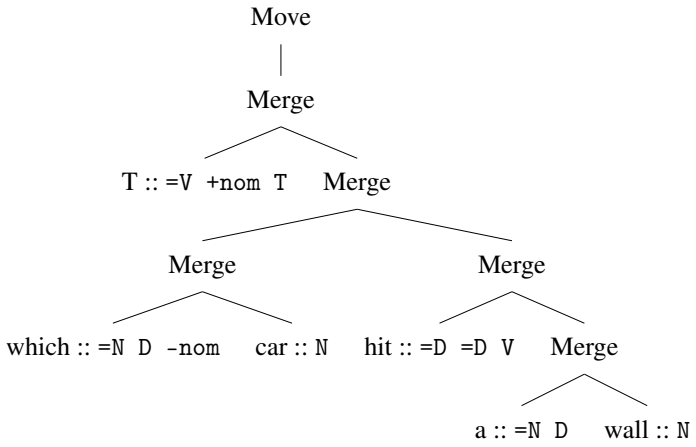
have the one in (17), where appropriate licensor and licensee features have been added for subject movement.

(17)



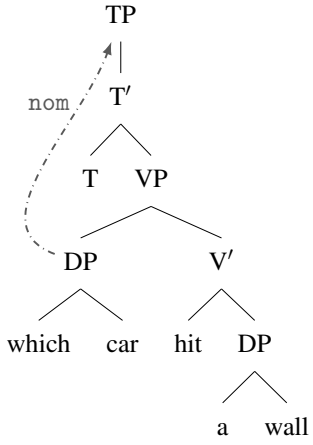
At this point, the head of the whole assembled structure is T, with feature string `=V +nom T`. But the selector feature `=V` has already been checked when the T-head merged with the whole VP, leaving the licensor feature `+nom` as the first unchecked feature on the T-head. This feature can only be checked by Move, so the whole structure is searched for a lexical item whose first unchecked feature is the matching licensee feature `-nom`. As we already know, this is the case for *which*, whose remaining unchecked feature is `-nom`. The two matching features thus trigger Move. In the phrase structure tree, this results in displacement of the whole phrase headed by *which* to Spec,TP. In the derivation tree, we merely record that the operation Move was triggered at this point.

(18)



Due to derivation trees only recording that Move is taking place, the geometry of the derivation tree no longer mirrors exactly the structure of the phrase structure tree. However, it is still the case that all the information of the phrase structure tree is encoded in the derivation tree. To see this, we may once again remove all features from all lexical items and relabel interior nodes in an X′-fashion. In addition, we also add feature-annotated movement arrows that connect the moving phrase to the corresponding Move nodes. The result is just a notational variant of standard phrase structure trees where the mover is depicted in its starting position rather than its final landing site.

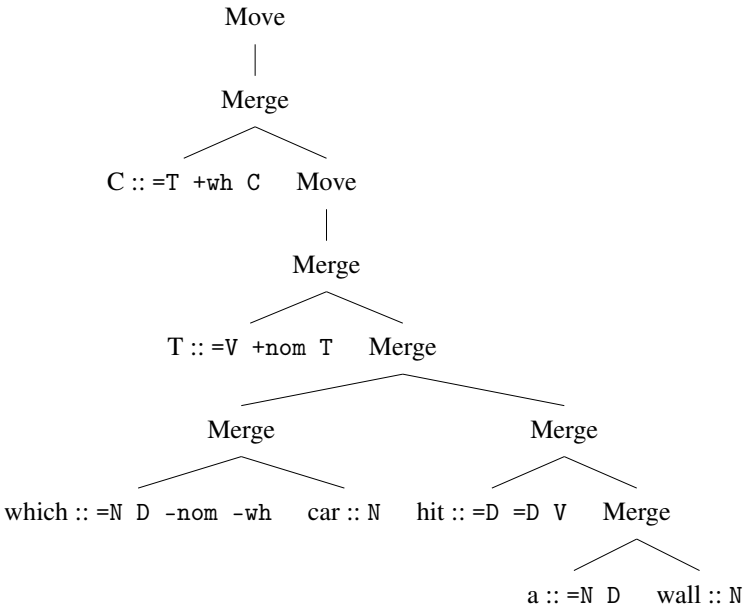
(19)



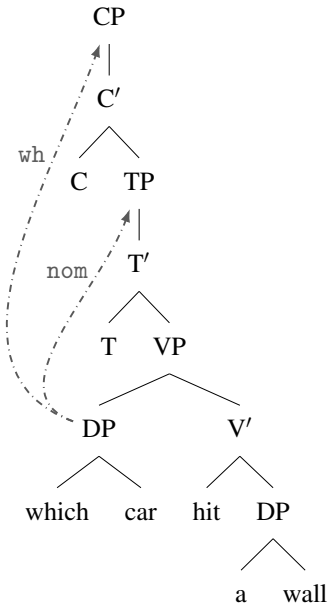
Even with the presence of Move, then, derivation trees are a viable syntactic encoding that furnishes all the information provided by phrase structure trees. The tree in (19) is still a derivation tree, albeit enriched with notational bells and whistles that should make it easier to read for most linguists. For the rest of the paper, we will use this format of derivation trees with movement arrows and X′-labels as notational aids.

We still have to finish our example derivation, though. Once movement has taken place, the T-head's only unchecked feature is the category feature `T`. This allows it to be merged with the C-head, which carries the matching selector feature `=T`. It also carries a licensor feature `+wh`, indicating that it needs to act as a landing site for *wh*-movement. In order to satisfy this requirement, we once again have to revise our analysis and posit that *which* actually carries two licensee features, first `-nom`, then `-wh`. With this change, the phrase headed by *which* will undergo *wh*-movement after it has undergone subject movement. The final derivation tree is depicted in (20) below in the conventional derivation tree format, whereas (21) shows it in the enriched format inspired by phrase structure trees.

(20)



(21)



This concludes our simple example, which has established all the required properties of MGs. Structures are built via the operations Merge and Move in a fashion that closely mirrors Minimalist syntax. The major change is that the feature calculus is more explicit, requiring each lexical item to be annotated with a finite list of features (category and selector features for Merge, licensee and licensor features for Move). The order of features on a lexical item determines the sequence of Merge and Move operations that this item participates in. Crucially,

a single word may correspond to several lexical entries that differ only in their feature make-up. For example, the determiner *a* corresponds to at least two lexical entries, depending on whether or not it heads the subject (and thus moves to Spec,TP via `-nom`).

(22) a. `a:: =N D`
b. `a:: =N D -nom`

Thus, in the spirit of the Borer-Chomsky conjecture, MGs are a fully lexicalized formalism where all syntactic variation is expressed purely in terms of feature annotations on lexical items.

We have only focused on the most basic version of MGs here as it is sufficient for this paper. There is a vast body of literature that extends MGs in various ways, including head movement (Stabler 2003), sidewards movement (Stabler 2006; Graf 2012), across-the-board movement (Kobele 2008; Torr & Stabler 2016), clustering movement (Gärtner & Michaelis 2010), phases (Stabler 2011a), adjunction (Frey & Gärtner 2002; Fowlie 2013; Graf 2014; Hunter 2015), island constraints (Gärtner & Michaelis 2007), transderivational constraints (Graf 2013), and much more. These extensions make MGs more similar to current strands of Minimalist syntax, but computationally they do not stray far from the basic version presented here. The remainder of this paper applies equally to this simple version of MGs and those with numerous additional operations and constraints.

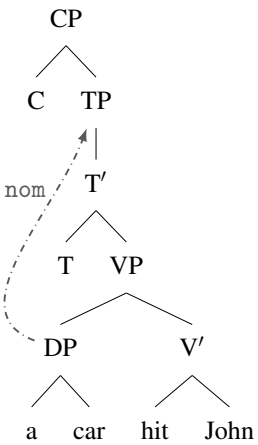
3.2 TOP-DOWN PARSING OF MGS

MGs provide us with a computational foundation for syntax, but in order to develop a rigorous account of the link between scope and sentence processing, we need to combine MGs with an equally rigorous theory of parsing. Fortunately, there has been plenty of work on MG parsing (Harkema 2001; Stabler 2011b; Stanojević & Stabler 2018; Torr et al. 2019). In the wake of Kobele et al. 2013 there has been a flurry of work using MG parsing as a model of human sentence processing, covering a wide range of constructions and phenomena (see the list in the introduction). They all share the common insight that processing predictions can be derived directly from the structure of MG derivation trees. Our proposal in Section 4 builds on this tradition and requires only minor modifications that are compatible with all prior work.

While there are many different parsing algorithms for MGs, work on sentence processing has largely focused on the MG top-down parser of Stabler (2013). Like every parser, this one takes as its input a string of lexical items, and it returns one or more trees that can be assigned to the input according to some predefined grammar. Stabler's top-down parser is actually a *recursive descent parser*. This means that (I) the parser starts at the root of the tree and works toward the leaves, (II) the parser always builds a full branch of the tree before working on a new branch, and (III) when given a choice between exploring branch A or branch B, the parser first works on whichever branch contains pronounced material that occurs farther left in the input string. The main innovation of Stabler's parser is that it builds MG derivation trees instead of phrase structure trees, and that it takes movement into account when choosing which branch to work on.

We omit a formal definition of the parser and instead provide an illustrative example. Suppose that we are given the input string *A car hit John*, which should receive the derivation tree below (depicted once again in the X′-inspired notation).

(23)



Let us go through each step that the MG top-down parser takes in building this structure. We recommend that the reader reference ***Figure 1*** at the same time.

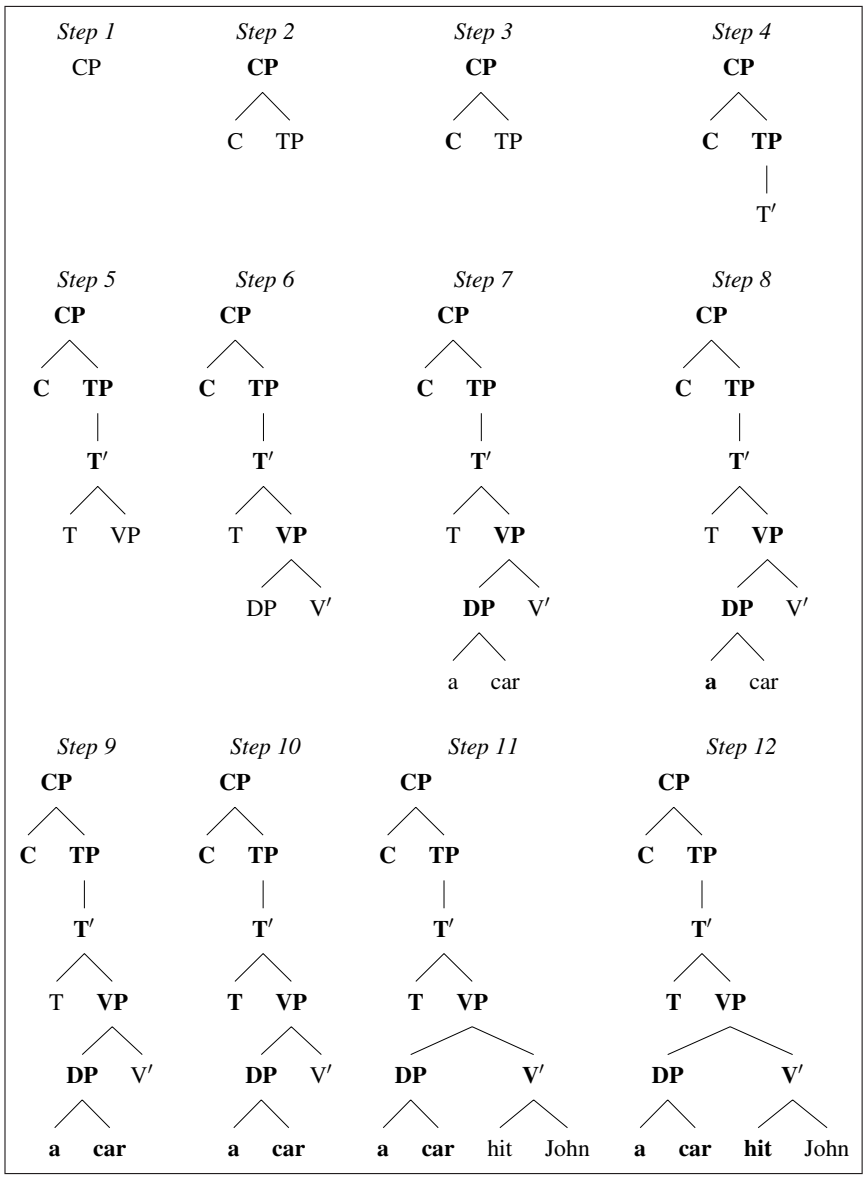


Figure 1 Stepwise construction of the derivation tree for *A car hit John*, with the final step of scanning *John* omitted; boldface indicates that the parser has completed all work on the node.

1. Since the parser operates in a top-down fashion, it has to start at the root node. By assumption, the input is a sentence, so the parser starts by positing a finite CP.
2. The parser now has to consider what kind of finite CP this may be. This depends on the grammar, but for English there are at least three options: a CP without any movement, a CP targeted by *wh*-movement, or a CP targeted by topicalization. Suppose that the parser somehow makes the right choice and posits a CP without movement (we will return to this issue at the end of this subsection). Then the parser may assume that this CP is built from a C-head and a TP, so it adds these two nodes to the tree, as daughters of CP.
3. Now that we have the partial structure $[_{CP}$ C TP], the parser has to decide which one of the two branches it has to work on first. Since the C-head will be linearized to the left of TP, it picks this one first. C is a lexical item, so there is nothing further to expand. Since C is an unpronounced head, we cannot verify its presence in the input string either, and the parser simply marks it as done.
4. The parser now starts working on TP. Again there are many ways a TP could be built, but the parser will assume that it is a TP that is targeted by subject movement. In this case, the derivation tree will only have a single T′-daughter, which is added below TP. In addition, the parser records that this is a T′ that must contain a subject mover in some lower position (i.e. a phrase carrying `-nom`).

DOI: 10.5334/gjgl.1209

5. Since there is only one branch to work on, the parser considers TP done and instead starts to expand T′. It assumes that T′ can be split into a T-head and a VP. It also has to guess which branch will lead to the subject mover, which is still missing from the structure. Since the T-head cannot be a subject mover, the choice is easy, and the parser records VP as the subtree containing the subject mover.

   Once again the parser has to decide which branch it should work on first. Earlier on, it preferred the C-head over the TP because C is linearized to the left of the TP. This time, the choice is different. While it is true that the T-head must be linearized to the left of its VP-complement, the VP is assumed to contain a subject mover that will end up to the left of the T-head. So the VP-branch contains material that will eventually be to the left of T. For this reason, the parser postpones further work on the T-head and instead keeps expanding the VP. Essentially, the parser took movement into account when calculating which branch is "leftmost" in the sense that it contains a piece that is to the left of all material in the other branch. This additional reasoning step is what separates the MG parser from standard recursive descent parsers.

6. With work on the T-head currently on hold, the parser instead expands the VP. As before, a reasonable MG for English will furnish many possible expansions, but we will assume that the parser picks the correct expansion: a DP subject and V′. Since the DP is the subject, it should undergo subject movement, and therefore the parser assumes that this DP will provide the required licensee feature `-nom`. This takes care of the movement dependency that was started earlier on when TP was expanded into T′. The parser also has to decide whether it wants to work on DP or V′ first. Based on the parser's guesses so far, no lexical material in V′ can appear to the left of the subject DP. Hence the parser decides to prioritize the subject DP and leave V′ for later.

7. The parser hypothesizes that the DP is built from the two lexical items *a* and *car*—to reiterate, there are thousands of alternative expansions, we are only focusing on the one here that will yield the correct tree structure.

8. Since the D-head *a* must be linearized to the left of its complement *car*, the parser chooses to work on *a* before *car*. This is the first pronounced lexical item the parser has encountered, so if the structure built so far is correct, the first symbol in the input string should be *a*. The parser performs a *scan* step to match the leaf node against the first input symbol. The two match, and the parser marks *a* in the input as successfully scanned. It also marks the D-head in the tree as completed and moves on to the complement.

9. Since the complement is the lexical item *car*, the parser performs another scan step. The first unscanned symbol in the input string is now *car*. Again we have a match, so the parser marks the input symbol *car* as scanned and also marks the leaf node *car* as completed.

10. Now that the whole subject DP is completed, the parser has to decide what to work on next. There are two unfinished pieces of structure: the T-head, and V′. Since by the parser's previous assumptions V′ does not contain any material that moves to the left of the T-head, the parser works on the T-head first. But as this is an unpronounced node that cannot be checked against the input, the parser simply marks it as done.

11. This only leaves us with V′. One final time the parser has to make a guess as to how this structure should be expanded, and it picks *hit* and *John*.

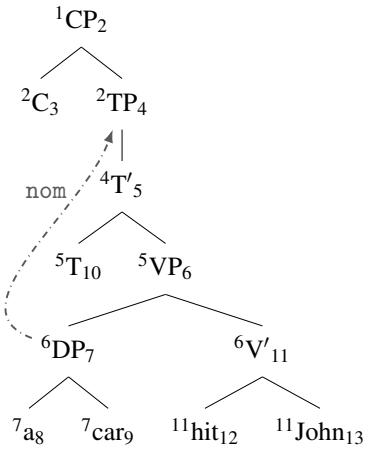
12. The parser prioritizes *hit* over *John* because the V-head must be to the left of the complement unless *John* were to move to some higher position, which it doesn't. It performs a successful scan step, matching the leaf node against the input.

13. This only leaves *John*. The parser performs another successful scan step. Since the derivation tree has been fully built, and there are no remaining input symbols, the parse is considered successful.

The parse steps above can be represented more compactly using the *index/outdex* notation of Kobele et al. (2013) (the term "outdex" is first used by Graf et al. 2015). Each node is annotated

with a superscript (its index) and a subscript (its outdex). A node's index denotes the step at which the parser first adds the node to the structure. Its outdex denotes the step at which the parser finishes all work on the node (although the parser may still be working on material underneath the node). This is shown in (24).

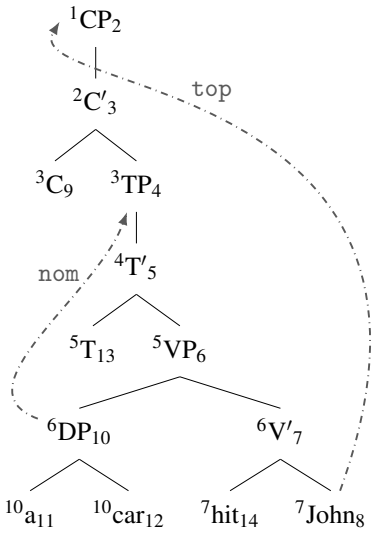
(24)



We encourage the reader to verify that each node's index and outdex in the example above does indeed match the steps where it is first introduced and finally completed, respectively. For the rest of the paper, we will make heavy use of this notation to represent the parser's behavior for specific sentences.

One more example may help solidify the reader's intuitions. Suppose that our input sentence involves topicalization of *John*, yielding *John, a car hit*. In this case the derivation tree is almost exactly the same, except for the addition of a Move node at the very top. But the index/outdex annotation is very different, reflecting a major change in how the parser constructs this derivation tree.

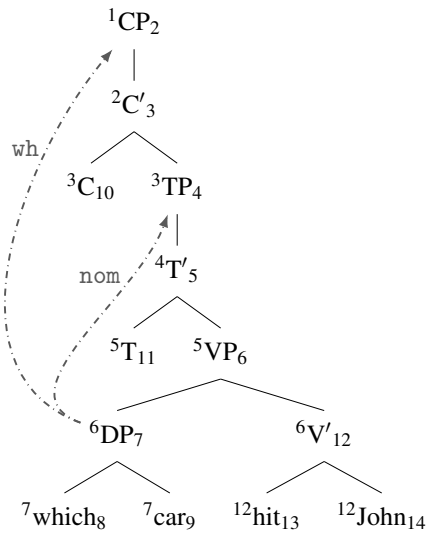
(25)



The topicalization of *John* completely changes the left-to-right order between elements. For instance, the parser has to delay working on C because TP contains an element that will appear to the left of C, namely *John*. For the same reason, the T-head, the subject, and the verb *hit* all have to be held in memory and cannot be completed right away. Instead, the parser has to make a beeline directly to *John*, and only once the parser has found *John* may it work on the rest of the tree. Quite generally, the parser always expands the tree in such a way that it can take the shortest route towards whatever lexical item it believes to appear next in the input string.

Finally, it is instructive to consider what happens in the case of intermediate movement, as in *Which car hit John*. Again the index/outdex-annotation differs quite a bit from (24) because the subject now moves to a position to the left of C, so that the latter cannot be worked on until the subject has been fully built.

(26)



But the intermediate subject movement step has no effect on anything else. In particular, note how the T-head can be worked on as soon as the C-head is completed. This is because the parser has already made the assumption that the subject mover is also the *wh*-mover, which must be to the left of the C-head. Hence there is no material in the input string between C and T — the intermediate landing site does not affect the linear order of lexical items and thus has no major effect on how the parser builds the tree.

In sum, the MG top-down parser of Stabler (2013) builds derivation trees in a top-down fashion, making educated guesses about how interior nodes should be expanded into one or two daughters, and which subtrees contain specific movers. If there are multiple places where the current tree could be expanded, the parser always prioritizes the node that it believes will ultimately yield a structure that contains the next symbol in the input string. When the parser reaches a lexical item, it uses a scan step to verify that this lexical item matches the current input symbol. This simple strategy allows the parser to be incremental and predictive, two essential properties of human sentence processing.

The discussion above abstracts away from many technical aspects of the parser, in particular its formal definition in terms of a parsing schema (Pereira & Warren 1983; Sikkel 1997), the representation of MGs in terms of prefix trees for more efficient search, and how incomplete parts of the tree are organized with a priority queue. While those are key aspects of the parser, they have no bearing on the subsequent discussion. Similarly, we say nothing about how the parser chooses between different ways to expand a given node, nor how it recovers from incorrect guesses. There are many different solutions to these issues, from backtracking or exhaustive parallel search to probabilistic search beams. In the parsing literature, these issues are considered part of the control structure, and Stabler's top-down parser is compatible with many different control structures. The recent work on MG models of human sentence processing completely sidesteps the issue of control structures and focuses just on how the shape of the derivation tree itself affects processing.

3.3 MGS FOR SENTENCE PROCESSING

There are many factors that go into human sentence processing and affect how hard a given sentence may be to process: structural ambiguity, the size of the search space, lexical frequency, context, memory retrieval, attention, and more. Kobele et al. (2013) argue that even if one puts aside all these factors and considers only how a specific derivation is assembled in a top-down manner, it is still possible to account for well-known processing contrasts. Even if one assumes that the parser always makes the right guesses, it can still be a demanding task to build the derivation tree because incomplete nodes must be held in memory. This reductionist approach has proved surprisingly fertile, providing novel, syntactically grounded explanations for processing phenomena. As was mentioned in the introduction, this includes center- vs. right-embedding, crossing vs. nested dependencies, subject vs. object relative clauses, stacked relative clauses, Heavy NP shift, dative DP attachment ambiguities, gradience in syntax, and priming effects. To the best of our knowledge, no other approach has been able to account

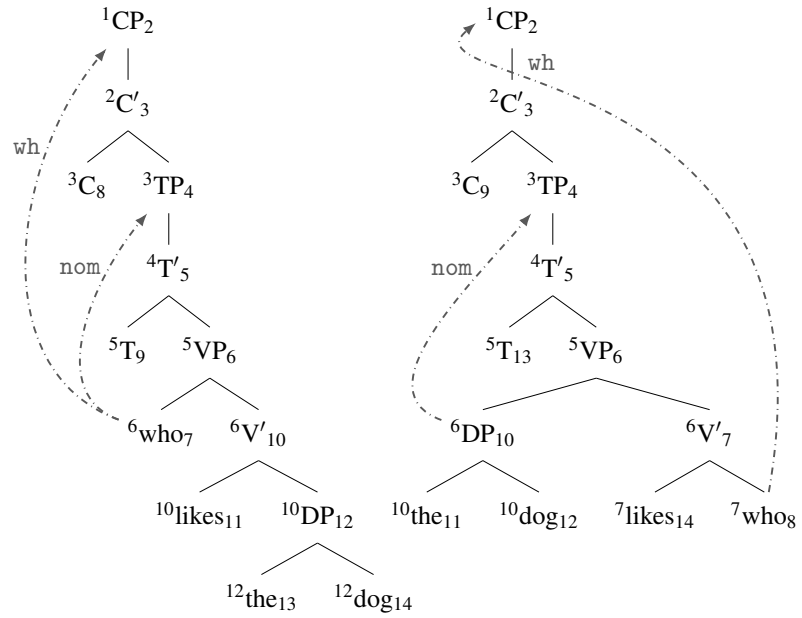
for such a diverse range of phenomena purely in terms of syntactic structure, and no other account is as sensitive to seemingly minor differences in syntactic structure.

For a simple illustration of this approach, consider once more the example of how the MG top-down parser builds the derivation tree for *A car hit John*. The index/outdex-annotated derivation tree in (24) shows how each node is introduced at a particular step and then completed at a later step. The difference between the index and the outdex is the amount of time that the node is stored in working memory, also known as its *tenure* (see also Joshi 1990 and Rambow & Joshi 1994 for an earlier application of this idea). High tenure is costly because it means that some parts of the structure have to be held in working memory for a long time. However, while the cost of tenure is a centerpiece of the MG processing work mentioned above, tenure will not play a role in this paper, and instead we focus on metrics that directly measure movement dependencies.

Movement dependencies are costly because the parser has to keep track of all the movers it still needs to find. The longer the movement dependency, the longer this information needs to be held in working memory. This kind of memory load is called **Size** by Graf et al. (2015) (a misleading term as it has nothing to do with the size of the mover). While we do not directly invoke **Size** in this paper, the metric we propose is closely modeled after it.

Size-based metrics can be used to explain, among other things, why object relative clauses are harder to process than subject relative clauses. Example (27) shows the derivation for a subject relative clause on the left and an object relative clause on the right; in order to keep the examples small, we omit all structure outside the relative clauses.

(27)



The object relative clause contains a longer movement path because the object starts from a lower position compared to the subject. Following Graf et al. (2015), we can quantify this as the difference between when a mover's existence is first conjectured and when that mover is finally encountered, as in (28). This formulation is not identical to Graf et al.'s (2015), but is equivalent to it; the different notation will help illustrate the eventual similarity between **Size** and the to-be-introduced **Location Differential** (**LD**).

(28) a. **Size**
Given a node n of derivation tree t, we write $o(n)$ for the outdex of n. We write m_n for the *mother* of n (the node immediately dominating n). We write f_n for the final landing site of n in t, where m_n qualifies as a landing site. Then the **Size** of n is $size(n) = o(m_n) - o(f_n)$; for the root node r (which has no mother), $size(r) = 0$.

b. **SumSize**
Given a derivation tree t, **SumSize** is the sum of *size(n)* for all nodes n of t.

Let us apply these definitions to the trees in (27). The subject relative clause contains only one mover, *who*, whose mother's outdex is 6: *who* is encountered in step 6. The final landing site of *who* is CP, whose outdex is 2: the mover's existence was conjectured when we expanded the

CP in step 2. Hence the **Size** of *who* is 6–2 = 4, and that's also the **SumSize** value of the whole tree because all other nodes have a **Size** of 0. The object relative clause, on the other hand, contains two movers, *who* and the subject DP. The outdex of *who*'s mother is now 7, and the outdex of its final landing site CP is 2, already giving us a slightly increased **Size** of 7–2 = 5. In addition, the subject DP's mother has an outdex of 6 and TP's (its final landing site) is 4, yielding a **Size** of 6–4 = 2. Hence the object relative clause has a **SumSize** value of 5 + 2 = 7, which is higher than the subject relative's value of 4. Since **SumSize** measures memory load and hence should be kept low, we correctly predict that subject relative clauses are easier to process than similar object relative clauses.

It must be pointed out that while the MG processing approach provides concrete numerical values for any given derivation, these numbers are not directly comparable. A derivation with a score of, say, 30 may still be easier to process than an entirely different derivation with a score of 5. The numbers only yield useful results when comparing minimal pairs. In addition, the scores only provide an *ordinal* scale—when comparing three minimally distinct trees *a*, *b*, and *c* with scores 5, 6, and 20, respectively, it need not be the case that *c* is much harder to process than *b* whereas *b* is only slightly harder than *a*. The claim is only that *a* is easier than *b*, which in turn is easier than *c*. These are major caveats, and a lot more work is needed to address these issues and strengthen the import of the computed scores. We will simply follow the established methodology of the MG processing approach, despite its current limitations.

4 A NEW METRIC FOR SCOPE PROCESSING DIFFICULTY

The discussion of MGs and sentence processing in Section 3 focused only on the PF-effects of movement: when a phrase moves, the linear order of terminals is altered. We said nothing about the semantic interpretation of derivation trees, let alone how movement interacts with scope. In order to address the scope data of Section 2, we have to disentangle the notions of PF-movement and LF-movement in MGs. We do so by adding LF-movement (Section 4.1) in a manner that still allows us the same index/outdex-format for representing the parser's behavior. We then define **Location Differential (LD)** as a formal method for quantifying how much a sentence's LF and PF differ from each other (Section 4.2). **LD** correctly handles all the scope data discussed in this paper (Sections 4.3–4.6).

4.1 ADDING LF-MOVEMENT

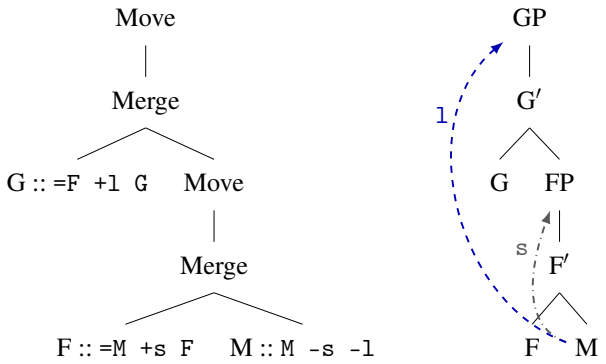
The original formulation of MGs in Stabler 1997 already allows for pure LF-movement, commonly referred to as covert movement. Stabler treats covert movement like any other instance of movement, except that the mover is not actually displaced. That is to say, covert movement is purely a feature checking operation. But keep in mind that in a derivation tree, movers are never displaced to begin with: they always remain in the position where they are merged. From the perspective of derivation trees, then, there is no relevant difference between overt and covert movement. The difference between the two only affects the resulting phrase structure tree.

The concept of a single resulting phrase structure tree, though, needs rethinking for our purposes. Taking inspiration from the inverted T-model, we assume that there are two distinct representations that are built from the same MG derivation tree: an LF-representation, and a PF-representation (cf. Kobele 2006). Movement then comes in three distinct varieties. One triggers displacement at both LF and PF (standard movement), one only at LF (covert movement), and one only at PF (movement followed by reconstruction). With respect to the MG feature calculus, the three kinds of movement behave exactly the same in that they involve the checking of a licensor feature and a matching licensee feature. Derivation trees then serve as a uniform data structure where various movement types can be interleaved as needed to produce distinct LF- and PF-structures.

Consider, for example, the toy derivation tree in (29), where the phrase M undergoes two distinct movement steps triggered by the licensee features -s and -l. We show the canonical derivation tree format on the left and the X′-style format on the right.

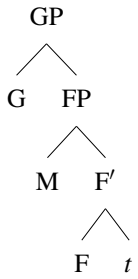
DOI: 10.5334/gjgl.1209

(29)

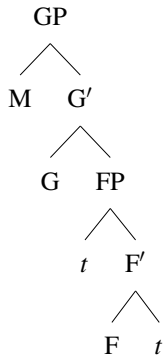


Each application of Move is licensed as it is triggered by matching licensor and licensee features. Now suppose that only s-features trigger standard movement that affects both PF and LF, whereas l-features trigger pure LF-movement. This does not alter the feature calculus at all, and the derivation tree above still describes a well-formed syntactic operation. But it affects what the output structure looks like that is built from this derivation tree. The PF-tree, depicted in (30a), sees M move to Spec,FP via the s-feature, but it does not move any farther because the final movement step is pure LF-movement, which has no effect on the PF-structure. The LF-tree in (30b) also has M move to Spec,FP, but then it subsequently moves to Spec,GP because l-features license movement at LF. Meanwhile, if the l-feature were replaced with a PF-only p-feature, this last movement would be PF-only instead of LF-only, meaning the PF-tree would look like (30b), and the LF-tree would look like (30a).

(30) a.



b.



Thus, MGs allow us to produce distinct PF- and LF-structures from the same derivation tree.

There are many different ways the proposed split between movement types can be implemented. Graf (2012) develops a fully parameterized system of *Movement-generalized MGs* where each movement feature is itself a complex bundle of parameters that fully define what kind of movement is triggered by this feature. In addition, the paper also provides a general methodology for defining constraints on how these movement types may interact (for instance, whether LF-movement across a closer PF-mover constitutes a superiority violation). To keep things as simple as possible, though, we will assume that movement features come in three different types indicated by subscripts.

(31) a. f triggers standard movement, which counts as both LF-movement and PF-movement
b. f_L triggers LF-movement
c. f_P triggers PF-movement

Movement arrows in the X′-style representations will be modified to reflect the type of movement. As before, gray dot-dashed arrows indicate standard movement, whereas blue dashed arrows are used for LF-movement and red dotted arrows for PF-movement. This convention was already used in (30). Another example is provided in ***Figure 2***, where the subject *a technican* undergoes standard movement to Spec,TP while the object *every plane* undergoes QR (i.e. LF-movement) to Spec,*v*P to avoid a type mismatch (Heim & Kratzer 1998). Since there is no general consensus as to what kind of feature may trigger QR, here we use the nondescript feature name sc_L (for scope).

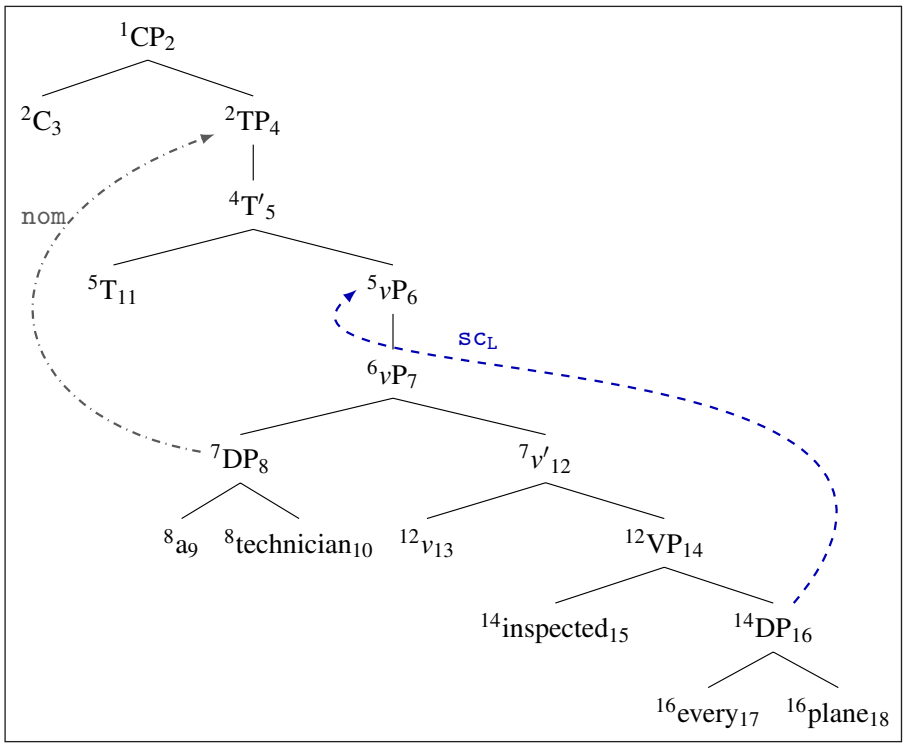


Figure 2 Annotated derivation tree for surface scope reading of *A technician inspected every plane*.

The reader may be wondering why the parser does not prioritize working on *every plane* in ***Figure 2*** once the subject and the T-head have been taken care of. In previous examples, movement of the object to a higher position always resulted in the parser prioritizing the mover over other material. But this was only because the movement had a PF-reflex. Recall that the parser chooses between alternative branches based on linearization. The object in ***Figure 2*** only undergoes LF-movement, which does not affect linearization. Since the object will still be linearized to the right of the verb, the parser won't work on it until the verb has been finished — the parser treats a phrase that only undergoes LF-movement like a phrase that does not move at all.

We would like to reiterate that it isn't terribly important how exactly one implements the three-way split between standard movement, LF-movement, and PF-movement. We provide the subscript solution as a demonstration that the split is easy to encode, not as an argument that this is the best way of doing it. Along the same lines, there are also many different kinds of LF- and PF-representations that can be used with this system. The implementation in the appendices, for example, yields LF phrase structure trees akin to those of Heim & Kratzer (1998), where movement operations leave traces that are coindexed with lambda-abstractors inserted below landing sites. Alternatively, the semantic interpretation could be done directly over derivation trees without any phrase structure intermediary, as in Kobele 2006. Importantly, our core insights are entirely independent of such considerations as they only hinge on MGs' ability to make a distinction between these three movement types in a manner that can be read off derivation trees.

4.2 EXTENDING MG PROCESSING TO LF: THE SLD PRINCIPLE

While our version of MGs now features three distinct movement types, the metrics developed in the MG processing literature were designed exclusively for standard movement. Per the above discussion, the MG processing literature has identified **SumSize** as one of the most useful

metrics for predicting overall processing difficulty. This metric operates on the intuition that the parser has to keep track of movers that it hasn't added to the (derivation tree) structure yet, which incurs additional memory load. Formally, **Size** is expressed as the difference between the step where the mover is first conjectured (the outdex of the final landing site) and the step where the mover is finally added to the structure (the outdex of its mother).

Building on this idea, we propose that when a mover's final PF landing site is distinct from its final LF landing site, processing cost increases the farther apart these two landing sites are.

(32) **Location Differential (LD)**
Given a node n in derivation tree t, let $o(n)$ be n's outdex. Furthermore, let m_n denote n's mother, and let l_n and p_n denote n's final LF and PF landing sites in t, respectively (where m_n qualifies as both an LF and PF landing site). The **Location Differential (LD)** of n is $\text{ld}(n) = |o(p_n) - o(l_n)|$. For root node r, $\text{ld}(r) = 0$.

$o(p_n)$ is the step in the parse at which n's existence as a mover has been hypothesized at PF, and $o(l_n)$ the step at which it has been hypothesized at LF. Since ld(n) is the absolute value of the difference between these two, it serves to measure the number of parse steps in which n has been hypothesized at one of PF or LF, but not the other. If n never undergoes movement or only undergoes standard movement, ld(n) will be 0: by fiat for the root, and because $p_n = l_n$ in the other cases. But if n ever undergoes PF-only or LF-only movement, ld(n) will be non-zero.

As an example, consider once more the derivation tree for *a technician inspected every plane* as depicted in ***Figure 2***. Since the subject only undergoes standard movement to TP, that is its final landing site for PF as well as LF. So if the subject is n, then $o(p_n) = o(l_n) = 4$, meaning ld(n) = 0. By contrast, the object DP *every plane* has an **LD** of 8. This is because its final LF landing site is the highest *v*P node, which has an outdex of 6. Since no PF movement takes place, by the above definition the final PF landing site for this DP is its mother, which has an outdex of 14. Thus, by the definition in (32) this DP has an **LD** of |14–6| = 8. In general, it suffices to identify, for every mover n, the two final landing sites of n and subtract the higher site's outdex from the lower one's.

LD is a natural generalization of **Size** in the sense that the latter can be regarded as measuring the difference between the PF tree and the derivation tree, rather than the PF tree and the LF tree. The reader may nonetheless be wondering about the cognitive motivation for **LD**. We contend that processing in a system with both PF and LF representations is more complex than the picture drawn in the computational parsing literature. Humans do not merely face the task of mapping an input to some syntactic representation. They have to relate the input to a semantic interpretation, with the syntactic structure as the intermediary. Hence we shouldn't just measure the cost of constructing a derivation tree, but also the cost of interpreting this derivation tree. Now suppose that a mover's final PF-landing site is higher than its final LF-landing site. Then the parser not only has to memorize that it is still looking for a mover, it also has to store that this mover will have to be integrated into the LF representation at a lower, as-yet-to-be-determined point in the structure. In the reverse case where the mover's final LF-landing site is higher than its final PF-landing site, the parser instead has to store that this mover cannot yet be linearly ordered with respect to the rest of the structure because its final PF position still remains to be found. The system we present in the appendices makes this increased workload explicit: when a mover has been predicted at PF but not LF (for example), it has to be assigned a "dummy" LF address until it has also been predicted at LF. The dissociation of PF and LF introduces additional bookkeeping that can only be minimized by keeping each mover's LF and PF positions as close together as possible.

LD, like **Size**, does not on its own allow us to compare distinct sentences because it is a property of nodes, not trees. Again we follow the example of **SumSize** and sum the location differentials of all nodes.

(33) **Summed Location Differential (SLD)**
Given a derivation tree t, **SLD** is the sum of ld(n) for all nodes n of t.

Higher **SLD** scores then indicate higher processing difficulty.

DOI: 10.5334/gjgl.1209

(34) **SLD Principle**
Parse *A* incurs a greater processing cost than Parse *B* if *A*'s **SLD** is greater than *B*'s **SLD**.

Recall from Section 3.3 that MG processing metrics—including the **SLD** Principle—should not be applied indiscriminately. Comparisons have to be limited to minimally distinct derivations, and the numerical values only provide us with an ordinal scale for ranking parses in terms of processing cost. Thus, while the **SLD** metric assigns a number to any given sentence, and while the comparison of **SLD** values serves as our metric for processing difficulty, the relative size of the difference between two **SLD**s should not be construed as an indication of the relative difference in processing difficulty of the two parses. There may indeed be a correlation in this regard, but this is an empirical question over and above the one we are attempting to address in this paper, and one that ought to be addressed via experimentation before a proper computational account can be provided.

Now that the **SLD** Principle has been clearly defined, we will show how it garners the right results for all of the cases discussed in Section 2. We will only show those details about derivations and parses that are necessary to illustrate how the **SLD** Principle derives the correct results; full details on the derivations and parses used can be found in the appendices.

4.3 CASE 1: SUBJECT > OBJECT

The first scope preference observation we will account for via the **SLD** Principle is the preference for subjects to scope over direct objects in simple transitives (Kurtzman & MacDonald 1993; Tunstall 1998; Anderson 2004). Our example sentence for this is (8a), *A technician inspected every plane*. We already encountered a surface scope analysis of this sentence in ***Figure 2***, in which the quantified object DP underwent LF-movement to a higher position to avoid a type conflict (see Heim & Kratzer 1998). We saw that the **SLD** for this parse was 8, due to the object's **LD** of 8.

Now compare this to the inverse scope derivation in ***Figure 3***. The object undergoes exactly the same kind of LF-movement as in the surface scope derivation, and consequently its **LD** is still 8. The subject, on the other hand, now undergoes PF-only movement instead of standard movement. As a result, its final PF landing site is TP with outdex 4 while its final LF landing site is the immediately dominating *v*P with outdex 7. This means that the subject's **LD** is 7–4 = 3. Overall, then, the **SLD** of the inverse scope derivation is 8 + 3 = 11, exceeding the **SLD** of 8 for the surface scope parse. Since the **SLD** Principle states that whichever parse has the lowest **SLD** is the easiest to process, we correctly predict that surface scope is less costly than inverse scope.

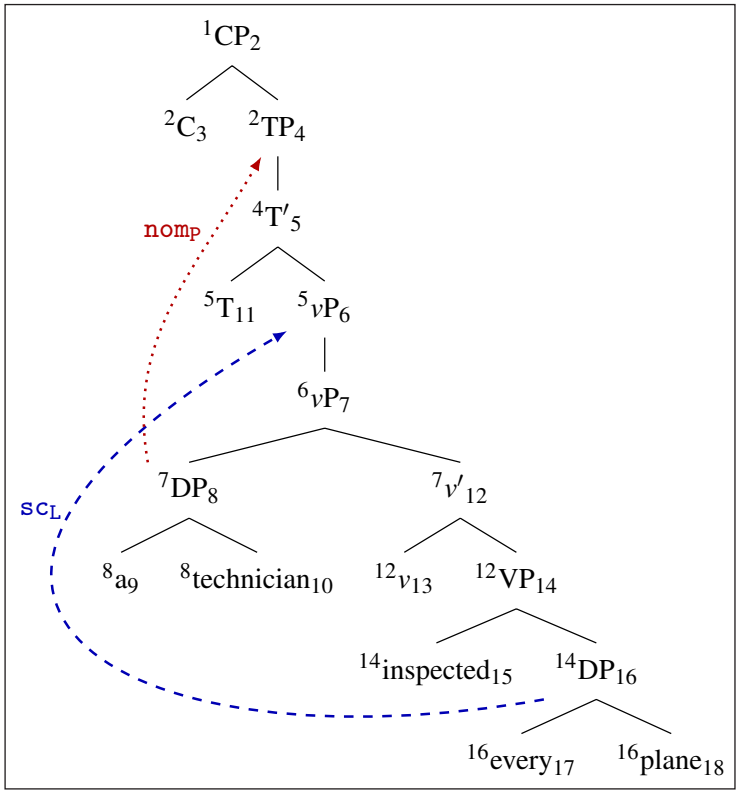


Figure 3 Annotated derivation tree for inverse scope reading of *A technician inspected every plane*.

DOI: 10.5334/gjgl.1209

Before moving on, note that there is another way we could have derived inverse scope. Rather than the subject scoping in its merge position with the object's obligatory movement leapfrogging it, the subject could have scoped in its landing site (Spec,TP), with the object undergoing additional QR to a position above this landing site. The same could be done for the cyclic QR cases that will be discussed in Section 4.5. For brevity's sake we will not go over these possible derivations in this paper, but they are included in the appendices; the resulting predictions are the same across the board.

4.4 CASE 2: SUBJECT/OBJECT VS. NEGATION

Next we discuss the relative scope configurations of subject and object universal quantifiers and sentential negation, as tested by Lee (2009). We only offer an explicit analysis of the English case, but at the end of this subsection we will sketch how the **SLD** Principle could cover Korean as well.

As discussed in Section 2.2, Lee (2009) provides evidence for a surface scope preference with respect to negation, a preference that is unexpected under Wurmbrand's account. In negated sentences with *every* in subject position, as in (10a) repeated below, there is a preference for surface scope, i.e. for *every* to scope above negation. That is, the preferred reading is that no kid fed the doves, rather than the weaker reading that at least one kid withheld their food.

(10a) According to the story, every kid didn't feed the doves in the park. (Lee 2009: p. 93)

As an illustration of why the **SLD** Principle makes the right predictions, we will use the simpler (35) as our example sentence.

(35) Every student did not pass the test.

Consider ***Figure 4***, which is the annotated derivation tree for the inverse scope interpretation of (35). We assume that the subject undergoes PF-movement rather than standard movement. Consequently, its LF position is still under negation, yielding the intended inverse scope reading. The **SLD** of this parse is easy to calculate as there are no other movers besides the subject. Its final PF landing site has outdex 4, and its final LF site has outdex 7. This results in an **LD** of 3, which is also the **SLD** of the whole parse for the inverse scope reading.

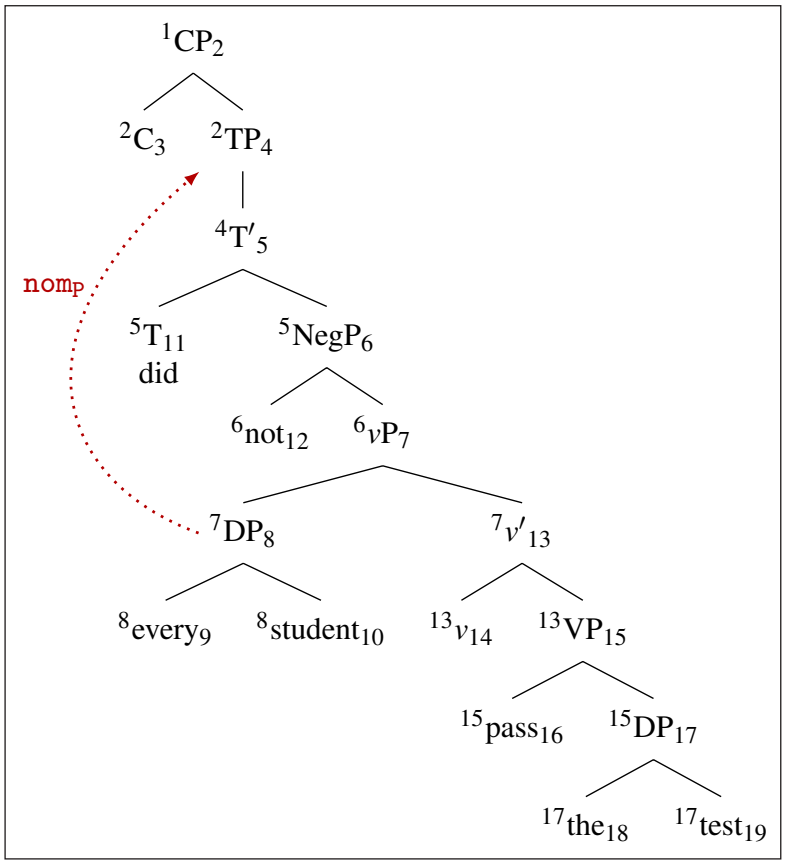


Figure 4 Annotated derivation tree for *Every student did not pass the test* (inverse scope).

DOI: 10.5334/gjgl.1209

The surface scope reading uses almost exactly the same derivation except that the subject now undergoes standard movement, which ensures that both its final PF position and its final LF position are above the negation. The subject now has an **LD** of 0—standard movement never results in a dissociation of LF and PF. As there are no other movement steps, 0 is also the **SLD** of the whole parse. Overall, then, we have an **SLD** of 0 for the surface scope reading and an **SLD** of 3 for the inverse scope reading. The **SLD** Principle once again predicts correctly that surface scope should be less costly than inverse scope.

Lee (2009) also observed a surface scope preference for objects and negation. Universally-quantified direct objects—such as *every candle* in (11a), repeated below—tend to scope under negation (*not* > *every*).

(11a) According to the story, Cindy didn't light every candle last night. (Lee 2009: p. 124)

Much like we did with subjects, we will use the structurally simpler (36) as our test case.

(36) Mary did not feed every patient.

Let us consider inverse scope first, with the corresponding parse shown in ***Figure 5***. Here the subject *Mary* undergoes standard movement to Spec,TP. As always, a single step of standard movement does not induce any dissociation between LF and PF, so the subject's **LD** is 0. As for the object, we assume that it undergoes two steps of LF-movement: first it moves to Spec, *v*P, then it moves to a higher position above negation. (Allowing the object to QR straight past negation instead of first stopping in Spec,*v*P has no meaningful impact.) The final LF landing site has an outdex of 6 while the final PF landing site has an outdex of 15. The **LD** of the object, then, is 15–6 = 9, meaning the overall **SLD** for the inverse scope derivation is 0 + 9 = 9.

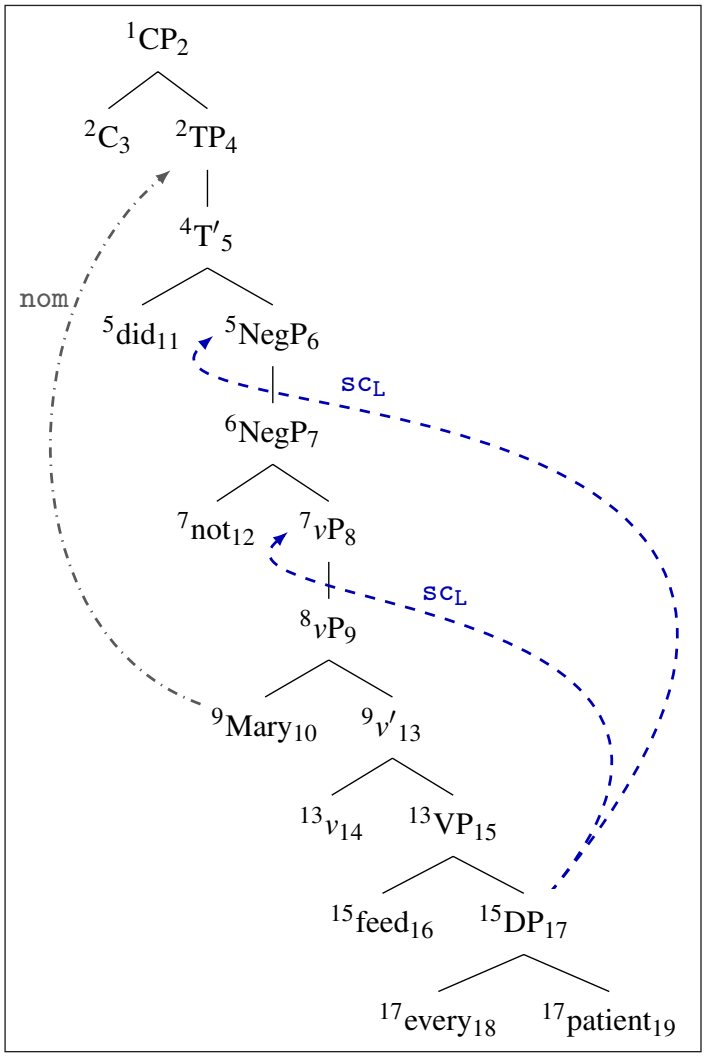


Figure 5 Annotated derivation tree for *Mary did not feed every patient* (inverse scope).

DOI: 10.5334/gjgl.1209

The surface scope reading, shown in ***Figure 6***, only differs in the absence of the second LF-movement step for the object. Its final LF position is now inside *v*P. Consequently, the object's **LD** reduces from 9 to 14–7 = 7. The **SLD** for a surface scope reading thus is 0 + 7 = 7. Once again the surface scope reading has the parse with the lowest **SLD** value and is thus correctly predicted to be preferred.

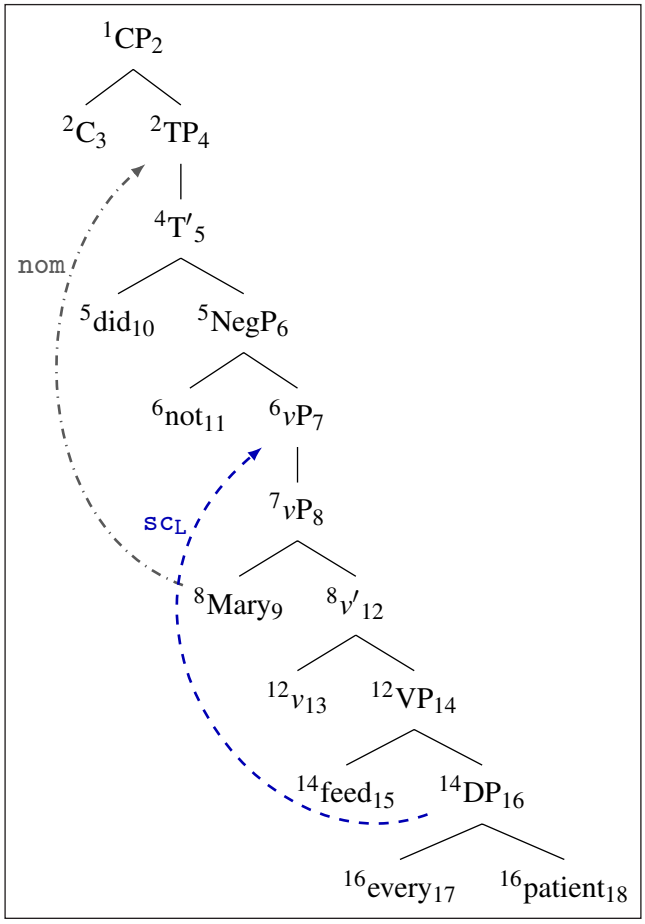


Figure 6 Annotated derivation tree for *Mary did not feed every patient* (surface scope).

This leaves us with Korean, where speakers prefer for both subjects and objects to scope over negation. That, too, follows immediately if one assumes that direct objects in Korean move to some higher functional position, as is indicated by their being linearized to the left of negation and the finite verb. The movement of the direct object could be standard movement or pure PF-movement. But the latter would induce a PF-LF mismatch, which is penalized by the **SLD** Principle. As a result, standard movement—and hence surface scope—is the less costly option. We see, then, that the **SLD** Principle in combination with an object movement analysis of Korean once again derives a surface scope preference as desired. The scope facts of English and Korean are both reflections of the same unifying principle that minimizes PF-LF mismatches, and the observed semantic differences are a consequence of the syntactic differences between these two languages.

4.5 CASE 3: CYCLIC QR

Next we account for the cyclic QR observations analyzed by Wurmbrand (2018). There are two types of observation that we wish to account for. The first are the within-sentence observations: for each sentence in (8), surface scope is easier than inverse scope. The second are the across-sentence observations: inverse scope for (8a) is easier than for (8b), which is easier than for (8c).

(8) a. A technician inspected every plane.
b. A technician tried to inspect every plane.
c. A technician decided to inspect every plane.

DOI: 10.5334/gjgl.1209

Recall from Section 2.1 that Wurmbrand accounts for these facts by means of an analysis in which clausal complements vary in their size, with *try*-complements being *v*Ps, and *decide*-complements including an additional future-shifting head WOLL. Since *v*P is a movement domain, the object *every plane* must undergo an extra QR above embedded *v*P to generate inverse scope in (8b). In (8c), yet another iteration of QR is required as *every plane* must also make a stop in Spec,WOLLP. Thus, on Wurmbrand's "costly trace" analysis we generate the correct across-sentence predictions, in addition to the within-sentence ones.

To illustrate the efficacy of the **SLD** Principle, we will adopt the same syntactic analysis as Wurmbrand, leading to the same predictions with respect to processing difficulty. However, as will be discussed later, we do not need every component of her syntactic analysis in order to capture the scope processing facts: the **SLD** Principle is flexible enough that even with major changes to the syntactic analysis, the correct results are still derived.

In order to determine whether the **SLD** Principle garners the right results, we first need to find the **SLD**s of the surface scope and inverse scope interpretations for each sentence in (8). We already did this for (8a) in Section 4.2; our results were **SLD**s of 8 for surface scope and 11 for inverse scope, correctly predicting a preference for surface over inverse scope. The relevant derivation trees for (8b) and (8c) are shown in ***Figures 7–9*** (we omit the surface scope derivation tree for (8c) because the only difference from ***Figure 7*** is the presence of a WOLLP above the embedded *v*P). Based on these structures, we obtain the values in ***Table 1***. We believe that the reader is able to verify these values on their own by now. The table shows clearly that the **SLD** Principle makes the correct predictions both within-sentence and across-sentence. For each sentence we predict surface scope to be easier than inverse scope, and we also predict inverse scope for (8a) to be easier than inverse scope for (8b), which in turn is easier than inverse scope for (8c).

A few clarifying comments are in order regarding (I) the proposed derivation trees, (II) the **SLD** values for surface scope, and (III) which aspects of Wurmbrand's analysis can be altered without affecting our results.

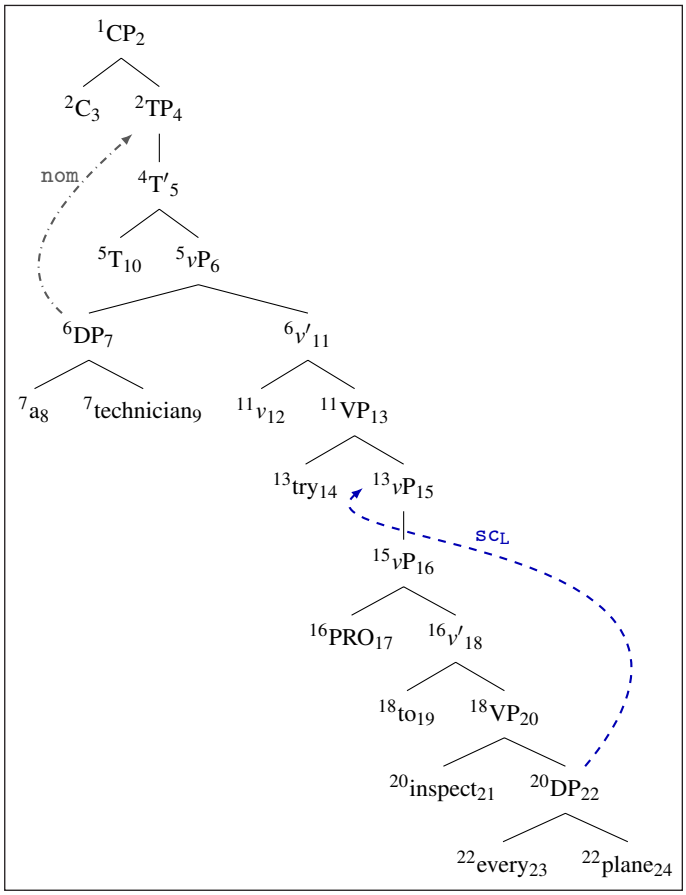


Figure 7 Annotated derivation tree for *A technician tried to inspect every plane* (surface scope).

DOI: 10.5334/gjgl.1209

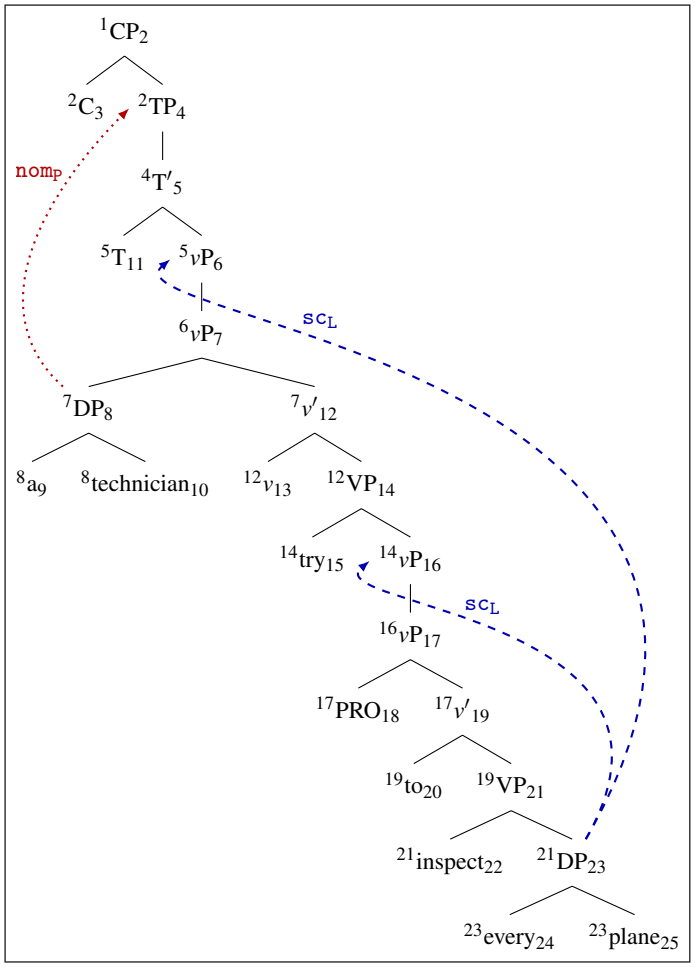


Figure 8 Annotated derivation tree for *A technician tried to inspect every plane* (inverse scope).

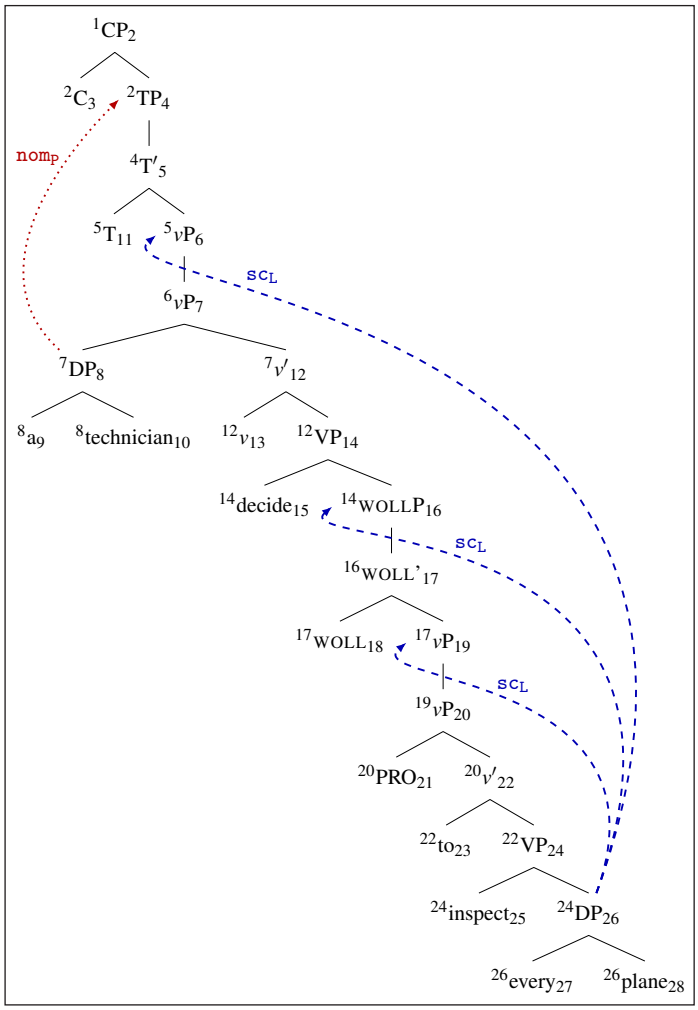


Figure 9 Annotated derivation tree for *A technician decided to inspect every plane* (inverse scope).

EXAMPLE	TYPE	SURFACE SCOPE	INVERSE SCOPE
(8a)	monoclausal	8	11
(8b)	*try* + inf	5	18
(8c)	*decide* + inf	5	21

Table 1 **SLD**s for surface and inverse scope parses for the sentences in (8).

We start with the proposed derivation trees. As before, we assume that objects must undergo QR to avoid type mismatches (Heim & Kratzer 1998). We also assume that infinitives contain a silent PRO as their subject. One could just as well use a movement-based analysis of control (Hornstein 1999) where the subject starts out in the embedded clause and first undergoes standard movement to Spec,*v*P in the matrix clause. Since standard movement never increases the mover's **LD**, this has no effect on the predictions made by the **SLD** Principle. For the surface scope readings, for (8b) and (8c) it also does not matter whether movement to Spec,TP in the matrix clause is standard movement or PF-movement—we assume the former so that we always compare the **SLD**s of the least costly parses for surface scope and inverse scope. For the inverse scope readings, movement of the matrix subject to Spec,TP is always pure PF-movement to make it easier for the embedded object to scope over the matrix subject. However, the appendices show that if this movement is instead standard movement, with the object undergoing additional QR to outscope TP, the same across-sentence results obtain. Finally, we adopt Wurmbrand's analysis where *try* takes a *v*P as a complement, whereas *decide* takes a WOLLP, which in turn contains a *v*P.

Next, we turn to the surprising prediction that surface scope in the monoclausal (8a) should be harder to process than in (8b) and (8c) because the latter have an **SLD** of 5 while the former has an **SLD** of 8. This unexpected prediction stems from a difference in the size of the relevant subjects. In the monoclausal construction, the parser has to fully build the subject DP *a technician* before it can work on the already conjectured object *every plane*. In the infinitival constructions, the parser only has to build a PRO, which takes fewer steps. There are two ways to address this: one could adopt the movement-theory of control, or one could allow the object to scope directly from the position where it is base merged (leading to a surface scope **SLD** of 0 across the board; see the appendices). After all, QR of the object in the surface scope parse is only driven by the need to avoid type mismatches, but many semantic formalisms do not require type-driven object movement (see, e.g., Keenan 2016; Pasternak 2020). Hence the precise predictions of the **SLD** Principle vary based on one's analysis; in line with previous work on MG processing, we consider this a feature rather than a bug as it provides another means of comparing competing analyses.

That being said, the parser and the memory load metrics that build on it aren't necessarily sensitive to all aspects of a given proposal. To wit, the predictions made in this section would stay exactly the same if all intermediate movement steps were omitted. Whereas Wurmbrand's account crucially relies on the number of QR steps and thus necessitates that the object stop in Spec,*v*P and Spec,WOLLP in (8c), the **SLD** Principle is agnostic about this. Since each node's **LD** depends only on the index and outdex of its final landing sites, not any of the intermediate ones, the deciding factor is what LF and PF positions a mover occupies, not how it got there. If it turned out that, say, movers can escape WOLLP without stopping in its specifier, then Wurmbrand would lose the distinction between *try*-infinitives and *decide*-infinitives, while the predictions of the **SLD** Principle would remain the same. We see, then, that the **SLD** Principle allows us to put Wurmbrand's insights on a rigorous quantitative foundation while at the same time reducing the number of syntactic assumptions that are needed to capture the observed processing effects.

4.6 CASE 4: WH-MOVEMENT VS. CYCLIC QR

Finally, let us look at the case of overt *wh*-movement. As mentioned in Section 2.2 this is problematic for Wurmbrand's analysis since, at least intuitively, (overt) cyclic *wh*-movement is considerably easier than (covert) cyclic QR. Or, more concretely, (9a) seems easier to process than an inverse scope interpretation of (9b).

(9) Wurmbrand 2018: p. 25, based on her (29)
a. What did a technician say that John inspected?
b. A technician said that John inspected every plane.

However, the **SLD** Principle gets the *prima facie* correct results without any further stipulation, at least if we follow Karttunen (1977) in treating moved *wh*-phrases as "scoping where they sit" in Spec,CP. To show that this is the case, we will use (37) as our test example:

(37) What did a technician try to inspect?

The annotated derivation tree for (37) can be seen in ***Figure 10*** (we assume that each intermediate landing site checks a different *wh*-feature on the mover, but one may also use a single persistent *wh*-feature as in Stabler 2011a or posit that successive cyclic movement is not feature-triggered at all, as in Kobele 2006 and Graf et al. 2016). Once again we start out with the subject *a technician* scoping in Spec,TP instead of its merge position, adopting the aforementioned principle that what is relevant is the *least* costly parse for each interpretation. Since all of the movements in this derivation are standard movements that affect both LF and PF, the **SLD** for this parse is 0. We therefore rightly predict that overt scope-taking operations are easier to process than their covert counterparts because they keep LF and PF in sync.

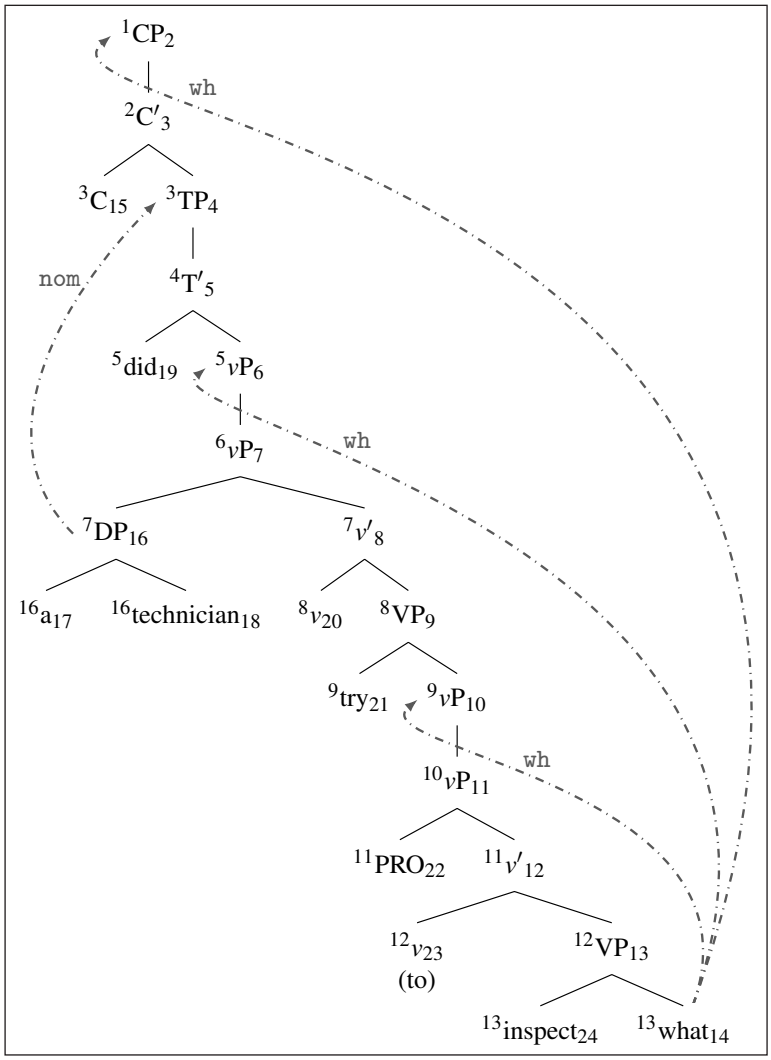


Figure 10 Annotated derivation tree, *What did a technician try to inspect?*

An interesting follow-up question—and one that we must unfortunately leave for future work—is what happens when we turn from overt *wh*-movement to cases in which some or all *wh*-phrases stay *in situ* at PF. This includes languages like Japanese, where true overt *wh*-movement does not exist, as well as English, where only one *wh*- phrase undergoes overt *wh*-movement to the left edge of a given clause, with the others appearing *in situ*:

(38) a. Which technician inspected which plane?
b. *Which technician [which plane]$_1$ inspected t_1?

We cannot yet offer an account of these cases based on the available empirical and theoretical findings. As far as we know, the empirical facts on the processing of moved *wh*-phrases vs. those that stay *in situ* have yet to be clearly established. There also seems to be no consensus in the theoretical literature as to what happens to *in situ wh*-phrases at LF. According to the classic analysis of Huang (1982), PF-*in situ wh*-phrases essentially do at LF what moved *wh*-phrases do at both PF and LF: namely, they move to a clause-initial position. In this case, for *wh-in situ* languages the **SLD** Principle would predict a sentence like (37) to be at least as difficult to process as (8b) because the *wh*-phrases would have to undergo QR-like LF-only movement. However, Kotek (2016) argues for a view in which *in situ wh*-phrases need not LF-move all the way to Spec,CP. Instead, they LF-move only as far as is needed for interpretive reasons—perhaps not at all in many cases (see Kotek & Hackl 2013 for experimental evidence in favor of this view). If this is correct, then the **SLD** Principle would not predict *in situ wh*-phrases to have a significant impact on processing difficulty, except when additional LF-movement is involved. Thus, in order to determine whether the **SLD** Principle generates the right results with respect to the processing of *in situ wh*-phrases, more work needs to be done to establish what the actual results are, as well as what the derived PF and LF structures for the relevant sentences are.

Finally, it is worth noting that the fact that cyclic *wh*-movement adds nothing to a parse's **SLD** does not mean that we necessarily predict that all (grammatical) *wh*-movement should be an absolute breeze to process. The MG processing literature already operates with several metrics, and our paper adds **SLD** into the mix in order to to capture a wide range of scope processing preferences. How exactly these metrics should be weighted is an open problem that goes far beyond the scope of this paper. Our goal was to demonstrate that Wurmbrand's idea of PF-LF mismatches as a source of processing difficulty is a natural fit for the independently supported MG approach to human sentence processing. In combining the two, we can account for data that is problematic for Wurmbrand's original proposal, and we also extend the boundaries of the MG processing framework from syntax into semantics.

5 CONCLUSION

In this paper we have followed Wurmbrand (2018) in adopting the view that extraclausal QR, though fully grammatical, is nonetheless difficult to process—often prohibitively so. We have additionally followed Wurmbrand in postulating that the processing difficulty of extraclausal QR depends on the size of the embedded clause, with the complements of *try*-type verbs being more conducive to extraclausal QR than the complements of *decide*-type verbs.

While we have argued against the particular theory of scope processing difficulty that Wurmbrand offers to account for these observations, we have proposed an alternative metric that is in keeping with a proposed revision that she suggests, in which processing difficulty is in part dependent on the severity of the mismatch between PF and LF representations. This theory was couched in a top-down parser for Minimalist Grammars, thereby embedding it within a framework that has already been used to successfully account for a variety of observations on syntactic processing from the experimental literature. The metric was then shown to make the right predictions for all of the data discussed by Wurmbrand, as well as those that were problematic for her original account.

By using a top-down MG parser to formulate our analysis of scope processing difficulty, we have added to a growing body of work dedicated to using such parsers to account for a variety of syntactic processing effects. The **Summed Location Differential** Principle that we propose assigns a precisely defined numerical score to any Minimalist Grammar derivation tree. As a result, it makes concrete processing predictions for a variety of constructions, many of which remain to be experimentally verified. Our hope is that the introduction of a robust and thus far successful scope processing metric will encourage more empirical work testing the predictions of the **SLD** Principle, so that it may be either further refined or replaced with an equally predictive and more empirically adequate alternative.

ABBREVIATIONS

ACC = accusative, COMP = complementizer, DECL = declarative, NEG = negation, NOM = nominative, PST = past tense, TOP = topic marker

ADDITIONAL FILE

The additional file for this article can be found as follows:

- **Appendices A–C.** Cyclic scope and processing difficulty in a Minimalist parser. DOI: *https://doi.org/10.5334/gjgl.1209.s1*

ACKNOWLEDGEMENTS

For helpful discussion, many thanks to John Drury, Richard Larson, and members of the ZAS Domains Reading Group. Comments from Johan Rooryck and two anonymous reviewers were extremely valuable.

FUNDING INFORMATION

Pasternak's research is funded by DFG Grant #387623969 (*DP-Border*, PIs: Artemis Alexiadou & Uli Sauerland).

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR AFFILIATIONS

Robert Pasternak *orcid.org/0000-0002-3024-1900*
Leibniz-Center for General Linguistics (ZAS), Schützenstraße 18 10117 Berlin, Germany

Thomas Graf
Department of Linguistics and Institute for Advanced Computational Science, Stony Brook University, Stony Brook, NY 11794, USA

REFERENCES

Anderson, Catherine. 2004. *The structure and real-time comprehension of quantifier scope ambiguity*. Evanston, IL: Northwestern University dissertation.

Chomsky, Noam. 1965. *Aspects of the theory of syntax*. Cambridge, MA: MIT Press. DOI: *https://doi.org/10.21236/AD0616323*

Chomsky, Noam. 1995. *The minimalist program*. Cambridge, MA: MIT Press.

Chomsky, Noam. 2000. Minimalist inquiries: The framework. In Roger Martin, David Michaels & Juan Uriagareka (eds.), *Step by step: Essays on Minimalist syntax in honor of Howard Lasnik*, 89–155. Cambridge, MA: MIT Press.

Chomsky, Noam. 2001. Derivation by phase. In Michael Kenstowicz (ed.), *Ken Hale: A life in language*, 1–52. Cambridge, MA: MIT Press.

Chomsky, Noam & George A. Miller. 1963. Introduction to the formal analysis of natural languages. In R. Duncan Luce, Robert R. Bush & Eugene Galanter (eds.), *Handbook of mathematical psychology,* vol. ii, 269–321. New York, NY: Wiley.

De Santo, Aniello. 2019. Testing a Minimalist grammar parser on Italian relative clause asymmetries. In *Proceedings of the ACL workshop on cognitive modeling and computational linguistics (cmcl) 2019*. June 6 2019, Minneapolis, Minnesota. DOI: *https://doi.org/10.18653/v1/W19-2911*

De Santo, Aniello. 2020. *Structure and memory: A computational model of storage, gradience, and priming*. Stony Brook, NY: Stony Brook University dissertation.

Farkas, Donka & Anastasia Giannakidou. 1996. How clause-bounded is the scope of universals? In Teresa Galloway & Justin Spence (eds.), *Semantics and linguistic theory (SALT)* 6. 35–52. DOI: *https://doi.org/10.3765/salt.v6i0.2764*

Fowlie, Meaghan. 2013. Order and optionality: Minimalist grammars with adjunction. In András Kornai & Marco Kuhlmann (eds.), *Proceedings of the 13th meeting on the mathematics of language (MoL 13)*, 12–20.

Fox, Danny. 2000. *Economy and semantic interpretation*. Cambridge, MA: MIT Press.
Fox, Danny. 2002. Antecedent-contained deletion and the copy theory of movement. *Linguistic Inquiry* 33(1). 63–96. DOI: *https://doi.org/10.1162/002438902317382189*
Fox, Danny. 2003. On logical form. In Randall Hendrick (ed.), *Minimalist syntax*, 82–123. Oxford: Blackwell Publishers. DOI: *https://doi.org/10.1002/9780470758342.ch2*
Frey, Werner & Hans-Martin Gärtner. 2002. On the treatment of scrambling and adjunction in Minimalist grammars. In Gerhard Jäger, Paola Monachesi, Gerald Penn & Shuly Wintner (eds.), *Proceedings of the conference on Formal Grammar*, 41–52.
Gärtner, Hans-Martin & Jens Michaelis. 2007. Some remarks on locality conditions and Minimalist Grammars. In Uli Sauerland & Hans-Martin Gärtner (eds.), *Interfaces + recursion = language? Chomsky's Minimalism and the view from syntax-semantics*, 161–196. Berlin: Mouton de Gruyter.
Gärtner, Hans-Martin & Jens Michaelis. 2010. On the treatment of multiple-wh-interrogatives in Minimalist grammars. In Thomas Hanneforth & Gisbert Fanselow (eds.), *Language and logos*, 339–366. Berlin: Akademie Verlag.
Gerth, Sabrina. 2015. *Memory limits in sentence comprehension: A structural-based complexity metric of processing difficulty*. Potsdam: Universität Potsdam dissertation.
Graf, Thomas. 2012. Movement-generalized Minimalist grammars. In Denis Béchet & Alexander J. Dikovsky (eds.), *LACL 2012*, vol. 7351 (Lecture Notes in Computer Science), 58–73. DOI: *https://doi.org/10.1007/978-3-642-31262-5_4*
Graf, Thomas. 2013. *Local and transderivational constraints in syntax and semantics*. Los Angeles, CA: UCLA dissertation. *http://thomasgraf.net/doc/papers/Graf13Thesis.pdf*.
Graf, Thomas. 2014. Models of adjunction in Minimalist grammars. In Glynn Morrill, Reinhard Muskens, Rainer Osswald & Frank Richter (eds.), *Formal Grammar 2014*, vol. 8612 (Lecture Notes in Computer Science), 52–68. Heidelberg: Springer. DOI: *https://doi.org/10.1007/978-3-662-44121-3_4*
Graf, Thomas, Alëna Aksënova & Aniello De Santo. 2016. A single movement normal form for Minimalist grammars. In Annie Foret, Glyn Morrill, Reinhard Muskens, Rainer Osswald & Sylvain Pogodalla (eds.), *Formal Grammar: 20th and 21st international conferences, FG 2015, Barcelona, Spain, August 2015, revised selected papers. FG 2016, Bozen, Italy, August 2016*, 200–215. Berlin, Heidelberg: Springer. DOI: *https://doi.org/10.1007/978-3-662-53042-9_12*
Graf, Thomas, Brigitta Fodor, James Monette, Gianpaul Rachiele, Aunika Warren & Chong Zhang. 2015. A refined notion of memory usage for Minimalist parsing. In *Proceedings of the 14th meeting on the mathematics of language (MoL 2015)*, 1–14. Chicago, IL: Association for Computational Linguistics. DOI: *https://doi.org/10.3115/v1/W15-2301*
Graf, Thomas, James Monette & Chong Zhang. 2017. Relative clauses as a benchmark for Minimalist parsing. *Journal of Language Modelling* 5(1). 57–106. DOI: *https://doi.org/10.15398/jlm.v5i1.157*
Hackl, Martin, Jorie Koster-Hale & Jason Varvoutis. 2012. Quantification and ACD: Evidence from real-time sentence processing. *Journal of Semantics* 29(2). 145–206. DOI: *https://doi.org/10.1093/jos/ffr009*
Harkema, Henk. 2001. A characterization of Minimalist languages. In Philippe de Groote, Glyn Morrill & Christian Retoré (eds.), *Logical aspects of computational linguistics (LACL'01)*, vol. 2099 (Lecture Notes in Artificial Intelligence), 193–211. Berlin: Springer. DOI: *https://doi.org/10.1007/3-540-48199-0_12*
Heim, Irene & Angelika Kratzer. 1998. *Semantics in generative grammar*. Oxford: Blackwell.
Hornstein, Norbert. 1999. Movement and control. *Linguistic Inquiry* 30. 69–96. DOI: *https://doi.org/10.1162/002438999553968*
Huang, C. T. James. 1982. Move WH in a language without WH movement. *The Linguistic Review* 1(4). 369–416. DOI: *https://doi.org/10.1515/tlir.1982.1.4.369*
Hunter, Tim. 2015. Deconstructing merge and move to make room for adjunction. *Syntax* 18. 266–319. DOI: *https://doi.org/10.1111/synt.12033*
Joshi, Aravind K. 1990. Processing crossed and nested dependencies: an automaton perspective on the psycholinguistic results. *Language and Cognitive Processes* 5(1). 1–27. DOI: *https://doi.org/10.1080/01690969008402095*
Karttunen, Lauri. 1977. Syntax and semantics of questions. *Linguistics and Philosophy* 1(1). 3–44. DOI: *https://doi.org/10.1007/BF00351935*
Keenan, Edward L. 2016. *In situ* interpretation without type mismatches. *Journal of Semantics* 33(1). 87–106.
Kennedy, Christopher. 1997. Antecedent-contained deletion and the syntax of quantification. *Linguistic Inquiry* 28(4). 662–688.
Kobele, Gregory M. 2006. *Generating copies: An investigation into structural identity in language and grammar*. Los Angeles, CA: UCLA dissertation. *http://home.uchicago.edu/~gkobele/files/Kobele06GeneratingCopies.pdf*.
Kobele, Gregory M. 2008. Across-the-board extraction and Minimalist grammars. In *Proceedings of the ninth international workshop on Tree Adjoining Grammars and related frameworks*.
Kobele, Gregory M., Christian Retoré & Sylvain Salvati. 2007. An automata-theoretic approach to Minimalism. In James Rogers & Stephan Kepser (eds.), *Model theoretic syntax at 10*, 71–80.

Kobele, Gregory M., Sabrina Gerth & John T. Hale. 2013. Memory resource allocation in top-down Minimalist parsing. In Glyn Morrill & Mark-Jan Nederhof (eds.), *Formal grammar: 17th and 18th international conferences*, 32–51. Springer. DOI: *https://doi.org/10.1007/978-3-642-39998-5_3*

Kotek, Hadas. 2016. Covert partial *wh*-movement and the nature of derivations. *Glossa: a journal of general linguistics* 1(1): 25. 1–19. DOI: *https://doi.org/10.5334/gjgl.49*

Kotek, Hadas & Martin Hackl. 2013. An experimental investigation of interrogative syntax/semantics. In Maria Aloni, Michael Franke & Floris Roelofson (eds.), *Proceedings of the 19th Amsterdam Colloquium*, 147–154.

Kurtzman, Howard S. & Maryellen C. MacDonald. 1993. Resolution of quantifier scope ambiguities. *Cognition* 48(3). 243–279. DOI: *https://doi.org/10.1016/0010-0277(93)90042-T*

Larson, Richard K. & Robert May. 1990. Antecedent containment or vacuous movement: Reply to Baltin. *Linguistic Inquiry* 21(1). 103–122.

Lee, So Young. 2019. A Minimalist parsing account of attachment ambiguity in English and Korean. *Journal of Cognitive Science* 3(19). 291–329. DOI: *https://doi.org/10.17791/jcs.2018.19.3.291*

Lee, Sunyoung. 2009. *Interpreting scope ambiguity in first and second language processing: Universal quantifiers and negation*. Manoa, HI: University of Hawai'i dissertation.

Liu, Lei. 2018. Minimalist parsing of heavy NP shift. In *Proceedings of the 32nd pacific asia conference on language, information and computation*. Hong Kong: Association for Computational Linguistics. *https://www.aclweb.org/anthology/Y18-1047*.

Moulton, Keir. 2007. Scope relations and infinitival complements. University of Massachusetts Amherst, Ms.

Pasternak, Robert. 2020. Compositional trace conversion. *Semantics & Pragmatics* 13(14). Early Access. DOI: *https://doi.org/10.3765/sp.13.14*

Pereira, Fernando C.N. & David Warren. 1983. Parsing as deduction. In *21st annual meeting of the association for computational linguistics*, 137–144. Cambridge, MA: MIT. DOI: *https://doi.org/10.3115/981311.981338*

Rambow, Owen & Aravind K. Joshi. 1994. A processing model for free word order languages. In C. Clifton, L. Frazier & K. Rayner (eds.), *Perspectives on sentence processing*, 267–301. Mahwah, NJ: Lawrence Erlbaum Associates.

Ruys, E. G. 2015. A Minimalist condition on semantic reconstruction. *Linguistic Inquiry* 46(3). 453–488. DOI: *https://doi.org/10.1162/LING_a_00189*

Sikkel, Klaas. 1997. *Parsing schemata* (Texts in Theoretical Computer Science). Berlin: Springer. DOI: *https://doi.org/10.1007/978-3-642-60541-3*

Stabler, Edward P. 1997. Derivational minimalism. In Christian Retoré (ed.), *Logical aspects of computational linguistics* (vol. 1328 of Lecture Notes in Computer Science), 68–95. Berlin: Springer. DOI: *https://doi.org/10.1007/BFb0052152*

Stabler, Edward P. 2013. Two models of minimalist, incremental syntactic analysis. *Topics in Cognitive Science* 5. 611–633. DOI: *https://doi.org/10.1111/tops.12031*

Stabler, Edward P. 2003. Comparing 3 perspectives on head movement. In A. Mahajan (ed.), *Syntax at sunset 3: Head movement and syntactic theory*, vol. 10 (UCLA Working Papers in Linguistics), 178–198. Los Angeles, CA: UCLA.

Stabler, Edward P. 2006. Sidewards without copying. In Gerald Penn, Giorgio Satta & Shuly Wintner (eds.), *Formal Grammar '06, proceedings of the conference*, 133–146. Stanford: CSLI.

Stabler, Edward P. 2011a. Computational perspectives on Minimalism. In Cedric Boeckx (ed.), *Oxford handbook of linguistic Minimalism*, 617–643. Oxford: Oxford University Press. DOI: *https://doi.org/10.1093/oxfordhb/9780199549368.013.0027*

Stabler, Edward P. 2011b. Top-down recognizers for MCFGs and MGs. In *Proceedings of the 2nd workshop on cognitive modeling and computational linguistics*, 39–48.

Stanojević, Miloš & Edward P. Stabler. 2018. A sound and complete left-corner parser for Minimalist grammars. In *Proceedings of the 8th workshop on cognitive aspects of computational language learning and processing*, 65–74. DOI: *https://doi.org/10.18653/v1/W18-2809*

Syrett, Kristen. 2015a. Experimental support for inverse scope readings of finite-clause-embedded antecedent-contained-deletion sentences. *Linguistic Inquiry* 46(3). 579–592. DOI: *https://doi.org/10.1162/LING_a_00194*

Syrett, Kristen. 2015b. QR out of a tensed clause: Evidence from antecedent-contained deletion. *Ratio* 28(4). 395–421. DOI: *https://doi.org/10.1111/rati.12107*

Syrett, Kristen & Jeffrey Lidz. 2011. Competence, performance, and the locality of quantifier raising: Evidence from 4-year-old children. *Linguistic Inquiry* 42(2). 305–337. DOI: *https://doi.org/10.1162/LING_a_00043*

Tanaka, Misako. 2015a. Asymmetries in long distance QR. In Anna E. Jurgensen, Hannah Sande, Spencer Lamoureux, Kenny Baclawski & Alison Zerbe (eds.), *Proceedings of the 41st annual meeting of the Berkeley Linguistic Society*, 493–501. Berkeley, CA: University of California, Berkeley Linguistic Society. DOI: *https://doi.org/10.20354/B4414110000*

Tanaka, Misako. 2015b. *Scoping out of adjuncts: Evidence for the parallelism between QR and wh-movement*. London, UK: University College London dissertation.

Torr, John & Edward P. Stabler. 2016. Coordination in Minimalist grammars: Excorporation and across the board (head) movement. In *Proceedings of the 12th international workshop on tree adjoining grammars and related formalisms (TAG+12)*, 1–17. Düsseldorf, Germany. *https://www.aclweb.org/anthology/W16-3301*.

Torr, John, Miloš Stanojević, Mark Steedman & Shay Cohen. 2019. Wide-coverage neural A* parsing for Minimalist grammars. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)*. Florence, Italy: Association for Computational Linguistics. DOI: *https://doi.org/10.18653/v1/P19-1238*

Tunstall, Susanne. 1998. *The interpretation of quantifiers: Semantics & processing*. Amherst, MA: University of Massachusetts Amherst dissertation.

Wurmbrand, Susi. 2001. *Infinitives*. Berlin: Mouton de Gruyter. DOI: *https://doi.org/10.1515/9783110908329*

Wurmbrand, Susi. 2014a. Restructuring across the world. In Ludmila Veselovská & Markéta Janebová (eds.), *Complex visibles out there. proceedings of the Olomouc Linguistics Colloquium 2014: Language Use and Linguistic Structure*, 275–294. Olomouc: Palacký University.

Wurmbrand, Susi. 2014b. Tense and aspect in English infinitives. *Linguistic Inquiry* 45(3). 403–447. DOI: *https://doi.org/10.1162/LING_a_00161*

Wurmbrand, Susi. 2015. Restructuring cross-linguistically. In Thuy Bui & Deniz Özyıldız (eds.), *Proceedings of the North East Linguistic Society* 45. 227–240. Amherst, MA: Graduate Linguistic Student Association (GLSA), University of Massachusetts Amherst.

Wurmbrand, Susi. 2018. The cost of raising quantifiers. *Glossa: a journal of general linguistics* 3(1): 19. 1–39. DOI: *https://doi.org/10.5334/gjgl.329*

Zhang, Chong. 2017. *Stacked relatives: their structure, processing and computation*. Stony Brook, NY: Stony Brook University dissertation.

TO CITE THIS ARTICLE:
Pasternak, Robert and Thomas Graf. 2021. Cyclic scope and processing difficulty in a Minimalist parser. *Glossa: a journal of general linguistics* 6(1): 8. 1–34. DOI: *https://doi.org/10.5334/gjgl.1209*

Submitted: 03 February 2020
Accepted: 03 October 2020
Published: 25 January 2021

Glossa: a journal of general linguistics is a peer-reviewed open access journal published by Ubiquity Press.