Psycholinguistic Adequacy of Left-corner Parsing for Minimalist Grammars

Lei Liu
Institute of Linguistics
Leipzig University
lei.liu@uni-leipzig.de

1 Introduction

How difficult a sentence is to process by humans is affected by, among many factors, the syntactic structure of the sentence. For example, in English, a center-embedding structure (1a) is generally more difficult to process than left- and right-embeddings (2a and 3a). As the number of layers increases, center-embeddings quickly become impossible to process (1b), while processing difficulties for left- and right-embeddings (2b and 3b) stay relatively the same.

(1) Center-embedding
  a. The rat that the cat bit is here.
  b. # The cheese that the rat that the cat bit ate is here.

(2) Left-embedding
  a. The rat's cheese is here.
  b. The rat's cheese's eyes are missing.

(3) Right embedding
  a. The rat that ate cheeses is here.
  b. The rat that ate the cheese that had eyes is here.

According to Resnik (1992), this processing difficulty difference between embedding structures is accounted for by the memory storage requirements of a parser to build corresponding structures. A left-corner (LC) parser for context-free grammars (CFGs) requires a memory space that is proportional to the height of the syntactic tree when building center-embedding structures. In contrast, when it builds left- and right-embeddings, only a constant amount of memory space is needed irrespective of the number of embedded layers. Based on the modeling results, Resnik (1992) argues that left-corner parsing is psychologically plausible as a model for human sentence processing.

Subsequent studies adopt the more sophisticated, better linguistically motivated Minimalist Grammars (MGs, Stabler 1996, 2011), aiming to develop a linking theory between processing phenomena and grammar formalisms. Studies have shown that the processing contrast between center- and right-embedding structures is derivable from the memory resources needed by a top-down parser for MGs (Kobele et al., 2013; Graf et al., 2017, among others). It is a common hypothesis in the psycholinguistic literature that processing difficulties correlate with memory usage (Kaplan, 1975; Joshi, 1990; Gibson, 1998, 2000, among others). Crucial to memory usage measurement in MG parsing models is the concept of *Tenure*, which measures the duration parse items remain in memory. Studies along this line have shown that tenure-based complexity metrics are successful in accounting for many processing phenomena cross-linguistically (e.g., verb-clusters (Kobele et al., 2013); stack relative clauses in Mandarin and English (Zhang, 2017), attachment ambiguity in English and Korean (Lee, 2018), gradient of difficulty in Italian relative clauses (De Santo, 2019), end-weight preference in English and Mandarin (Liu, 2022), among others).

However, just like in top-down CFG parsing, complexity metrics in top-down MG parsing cannot model the processing contrast between left- and center-embedding (Kobele et al., 2013). Moreover, while LC parsing for CFGs can model how humans process left-, center-, and right-embeddings, less is known about whether the results hold in LC parsing for MGs. Even less is understood regarding possible complexity metrics for this processing model.

This paper aims to address these gaps. We show that when modeling left-, center-, and right-embeddings, LC parser for MGs behaves similarly to that for CFGs: the parser requires memory resources that are proportional to the number of embedding layers to parse center-embedding structures. In contrast, only a constant amount of memory resources is needed to parse left- and right-embeddings. In doing so, we propose a complexity metric derived directly from the concept of tenure. The results, if on the right track, indicate that left-corner parsing for MGs is psycholinguistically adequate as a model for human sentence processing.

2 Left-corner Parsing for Minimalist Grammars

The left-corner parser for MGs used in this study is based on Stanojević and Stabler (2018) and Hunter et al. (2019). Intuitively, for any parsing step, the parser can perform one or more of the following operations: **shift**, which reads in the next word; **LC predict**, which creates and stores a parse item of the form `A => B`[1] where `A` is the sister node and `B` the mother node of the current input; and **complete**, which replaces a parse item `A => B` with `B` when `A` is built or confirmed from the input. In addition, arc strategies are specified based on whether stored parse items can **connect** with one another. Also, following Hunter et al. (2019), the parser is allowed to directly predict the landing site when the current word is a movement licensor (**unmove**).

Based on those operations, we can already calculate the steps any parse item is stored in the parser's memory – this is precisely the definition of tenure in top-down MG parsing literature. In our calculation, we assume that the parser always finds the correct structure if there is one. This highlights how syntactic structures affect processing predictions. For example, (4) is a record of the LC MG parser's behavior when it builds the sentence (with functional heads) *The rat t v ate cheeses*. We can see, for instance, that the parse item `v' => vP` remains in memory at steps 4 and 5. It has a tenure of 2. We call this *item tenure* to distinguish from the same concept used in top-down MG parsing work.

(4)

Step	**parse item**
1. shift the::	the::
2. LC the::	NP => DP
3. shift rat:: + complete	DP:
4. LC the rat:	v' => vP
5. shift t::	t:: v' => vP
6. LC t:: + unmove + connect	v' => TP
7. shift v::	v:: v' => TP
8. LC v:: + connect	VP => TP
9. shift ate:	ate:: VP => TP
10. LC ate:: + connect	DP => TP
11. shift cheeses:: + complete	TP

[1]This is a much simplified representation of parse items. For example, tree nodes are used instead of lexical items and their features. Information such as string span and movement chains is also ignored.

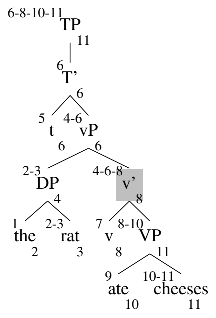
With the current setup of parser operations and steps, we have tree annotations that are condensed yet complete representations of the parser's behaviors. We still use superscript (index) and subscript (outdex) to indicate when a node is introduced to and removed from the memory. To faithfully represent LC parser's behavior, additional specifications are needed.

First, a LC parser can update its prediction regarding a particular node a few times. For example, a node in a LC parse can generally be predicted twice: once as a mother node and once as a sister node in two separate LC predictions. Also, in an arc-eager parse, structures already built can be used to form new structures. When this happens, the parser updates its predictions regarding the "old" nodes that now form new structures. We append to a node's index all the steps at which the parser makes updates to it and connect them with a dash ("-").

Second, a node is removed from the memory when the parser LC predicts or unmoves based on that node, or for an arc-eager parse, when two parse items connect at that particular node. We put these steps to the outdex of the node.

After specifying how indices and outdices are assigned, we are ready to represent a LC MG parser's behavior using tree annotations. For example, (5) is a tree annotation for the same sentence discussed earlier.

(5)



If we zoom in to the node ${}^{4\text{-}6\text{-}8}v'_{8}$, the index 4-6-8 indicates that the parser updates its predictions regarding the v' node three times. First, the node is predicted at step 4 from a LC prediction based on its sister node, DP. The corresponding parse item created and stored at this step is `v' => vP`. Second, v' is updated at step 6, when the parse item containing the node is connected to higher structures. Finally, it is updated again at step 8 from a LC prediction based on v. The outdex of v' is also 8 because at step 8, v' is where two parse items connect (`VP => v', v' => TP`).

Tree annotation allows for a visually easier calculation of *item tenure*. On the same node ${}^{4\text{-}6\text{-}8}v'_{8}$, the steps between its first and second updates are 2, which is the *item tenure* of the item `v' => vP`. We have seen this when discussing the parsing steps. The steps between its second and third updates are also 2. This is the *item tenure* of the item `v' => TP`. We can recover this by looking for the nodes with matching update records.

Based on *item tenure*, one can come up with a variety of complexity metrics. Here we explore just one such possibility: Maximal *item tenure* (MaxT$_{item}$), which is the maximal duration that any item remains in memory. For example, (5) has a MaxT$_{item}$ of 2, found on multiple nodes. Next, I show that MaxT$_{item}$ is successful in predicting the processing differences between left-, center-, and right-embeddings.

3 Modeling Results and Conclusions

The processing phenomena we model are the processing contrasts between left-, center-, and right-embedding structures. For each embedding direction, we include two layer conditions, 1-layer and 2-layer. For each parse, both an arc-standard and an arc-eager variant are used. A total of 12 ($= 3 \times 2 \times 2$) parses are constructed. Only the arc-eager results are discussed in this paper. All the target sentences are in (1-3).

We expect to derive that center-embedding is more difficult to process than both left- and right-embedding under both layer conditions. We also expect to find a constant difficulty measurement for left- and right-embedding under the two layer conditions, but a steep difficulty increase for center-embedding as the number of layers increases.

The results are summarized in Table 1.

Parser	Left	center	right
LC$_{MG}$ (arc-standard)	$O(1)$	$O(n)$	$O(n)$
LC$_{MG}$ (arc-eager)	$O(1)$	$O(n)$	$O(1)$
C.f. Human parser	$O(1)$	$O(n)$	$O(1)$

Table 1: Modeling results (Format and human parser results from Resnik 1992, Arc-standard results not discussed in this paper)

Following Resnik (1992), $O(1)$ means constant memory cost, and $O(n)$ means memory cost that is proportional to the size of the tree (n). Overall, for the arc-eager variant of LC parsing for MGs, as the number of layers increases, MaxT$_{item}$ remains the same for left- and right-embeddings, but grows as the number of layers grows in center-embeddings. Tree annotation excerpts for arc-eager parses of all the target sentences can be found in Appendix A.

For left-embeddings, MaxT$_{item}$ is 2 under both layer conditions, predicting that the parser requires only a constant amount of memory space to process left-embeddings. This prediction is the same as that based on a LC parser for CFGs. Given the syntactic assumptions in our model, the DPs containing left-embeddings are structurally the same as those derived via CFGs. We thus expect similar behaviors and similar processing predictions from the two parsers.

For center-embeddings, MaxT$_{item}$ is 10 under 1-layer condition, and 24 under 2-layer condition. This predicts that as the number of layers increases, center-embeddings become increasingly difficult to process. Center-embeddings are object relative clauses where the object linearly precedes the subject and its verb. Accordingly, the LC parser builds the relativized object and stores the parse item that has the object as its left-corner. Only when the parser operates on the verb can it use the parse item waiting in memory to build new structures. During the wait, the parser builds the subject, whose size grows with the number of embedding layers. MaxT$_{item}$ found on the waiting item reflects exactly this proportional increase in memory requirement when the number of embedding layers increases for center-embeddings.

For right-embeddings, MaxT$_{item}$ is 6 under both layer conditions. This predicts that the parser, like in the left-embedding case, requires a constant amount of memory space to process right-embeddings. Based on a promotion analysis of relative clauses, the item stored for the longest duration is held in memory between the parser operating on the first input word (*the*) and the relative pronoun (*that*). Increasing the number of layers does not add more material – i.e., more work for the parser – between the two input words. Hence a constant memory load is expected across layer conditions for right-embeddings.

To conclude, the modeling results suggest that left-corner parsing for MGs successfully derives human processing differences in left-, center-, and right-embeddings. MaxT_{item} is shown to be a valid complexity metric for our processing model. The results extend the parsing account for this processing contrast to another grammar formalism, MGs and suggest that left-corner parsing for MGs is viable as a psycholinguistically adequate model for human sentence processing. The proposed tree annotation scheme invites future research on the space of proper complexity metrics for LC parsing for MGs.

References

Aniello De Santo. 2019. Testing a minimalist grammar parser on italian relative clause asymmetries. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 93–104.

Edward Gibson. 1998. Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68(1):1–76.

Edward Gibson. 2000. The dependency locality theory: A distance-based theory of linguistic complexity. *Image, language, brain*, 2000:95–126.

Thomas Graf, James Monette, and Chong Zhang. 2017. Relative clauses as a benchmark for minimalist parsing. *Journal of Language Modelling*, 5(1):57–106.

Tim Hunter, Miloš Stanojević, and Edward Stabler. 2019. The active-filler strategy in a move-eager left-corner minimalist grammar parser. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 1–10.

Aravind K Joshi. 1990. Processing crossed and nested dependencies: An automation perspective on the psycholinguistic results. *Language and cognitive processes*, 5(1):1–27.

Ronald M Kaplan. 1975. *Transient processing load in relative clauses*. Ph.D. thesis, Harvard University.

Gregory M Kobele, Sabrina Gerth, and John Hale. 2013. Memory resource allocation in top-down minimalist parsing. In *Formal Grammar*, pages 32–51. Springer.

So Young Lee. 2018. A minimalist parsing account of attachment ambiguity in english and korean. *Journal of Cognitive Science*, 19(3):291–329.

Lei Liu. 2022. *Phrasal Weight Effect on Word Order*. Ph.D. thesis, State University of New York at Stony Brook.

Philip Resnik. 1992. Left-corner parsing and psychological plausibility. In *COLING 1992 Volume 1: The 14th International Conference on Computational Linguistics*.

Edward Stabler. 1996. Derivational minimalism. In *International Conference on Logical Aspects of Computational Linguistics*, pages 68–95. Springer.

Edward P Stabler. 2011. Computational perspectives on minimalism. *Oxford handbook of linguistic minimalism*, pages 617–643.

Miloš Stanojević and Edward Stabler. 2018. A sound and complete left-corner parsing for minimalist grammars. In *Proceedings of the Eight Workshop on Cognitive Aspects of Computational Language Learning and Processing*, pages 65–74.

Chong Zhang. 2017. *Stacked Relatives: Their Structure, Processing and Computation*. Ph.D. thesis, State University of New York at Stony Brook.

A Tree annotations for Left-, right-, and center-embeddings based on arc-eager LC MG parser

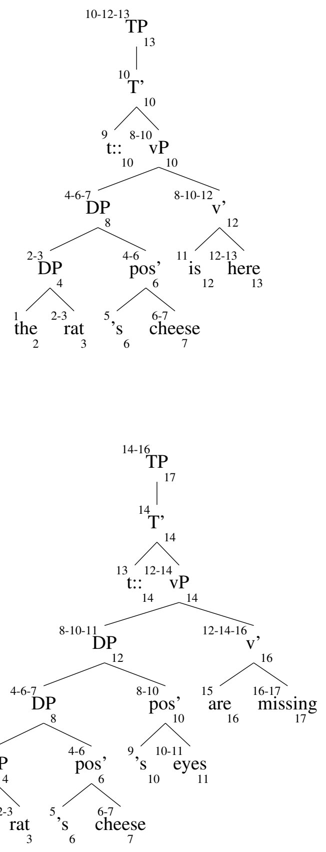


Figure 1: Tree annotations for left-embedding

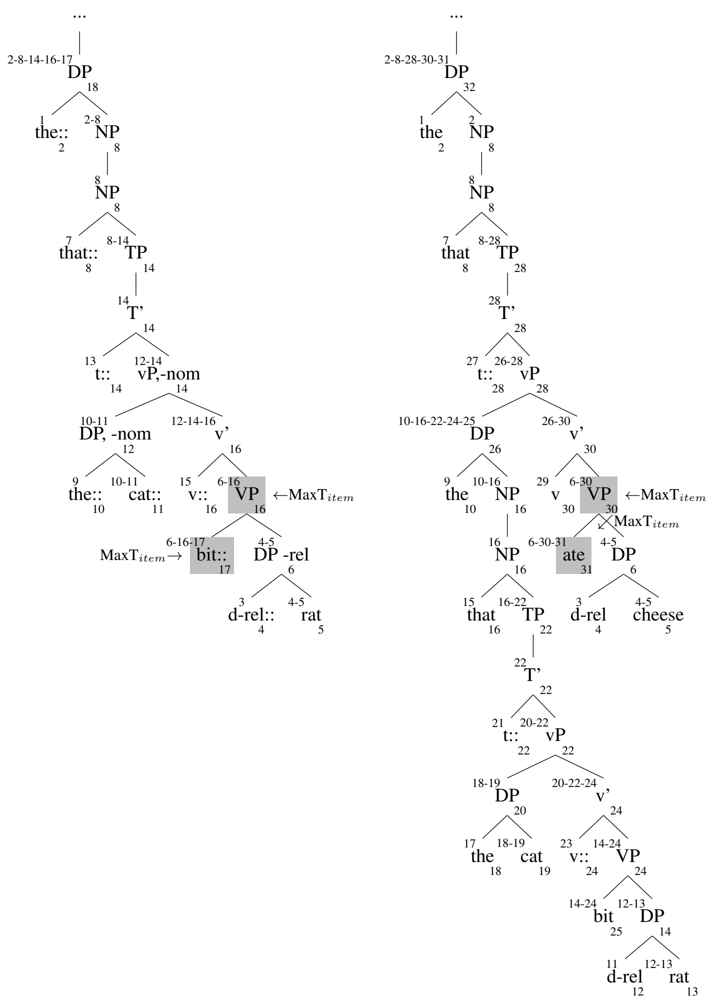


Figure 2: Tree annotations for center-embedding

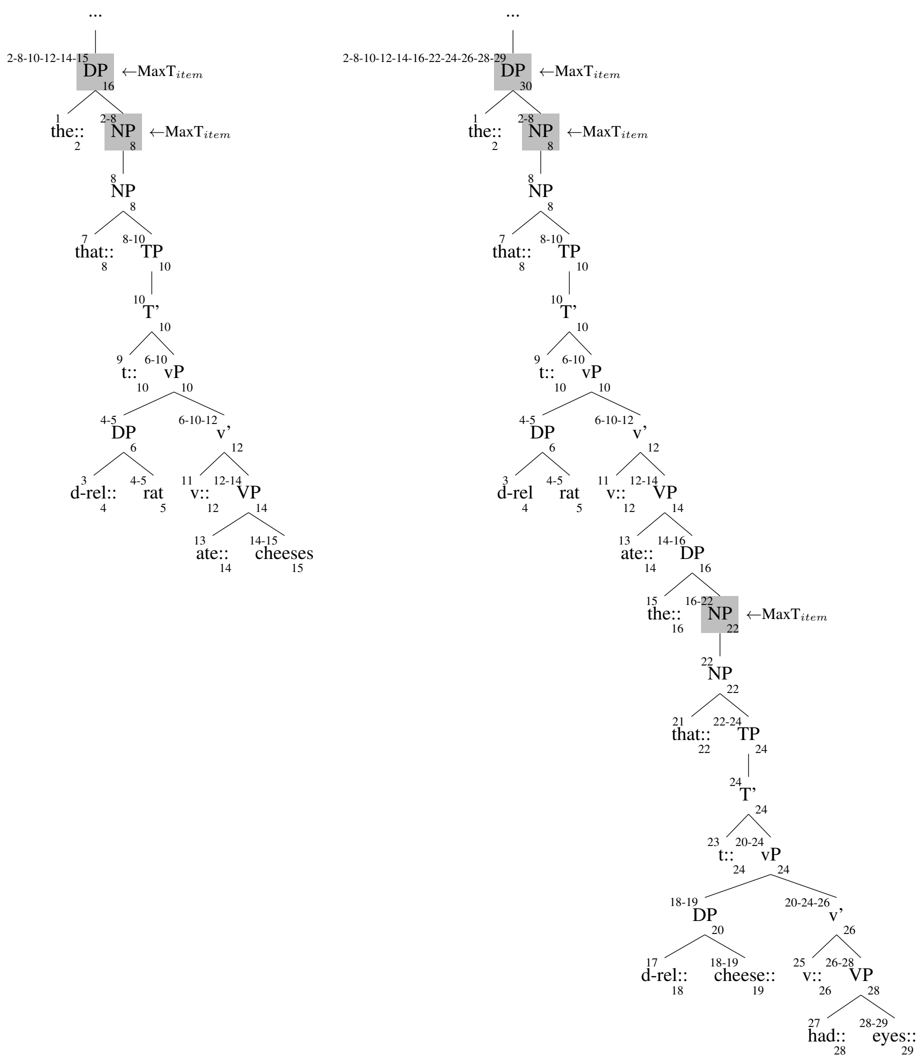


Figure 3: Tree annotations for right-embedding