A Minimalist Approach to Facilitatory Effects in Stacked Relative Clauses

Aniello De Santo
Department of Linguistics
University of Utah
aniello.desanto@utah.edu

Abstract

A top-down parser for Minimalist grammars (MGs; Stabler, 2013) can successfully predict a variety of off-line processing preferences, via metrics linking parsing behavior to memory load (Kobele et al., 2013; Gerth, 2015; Graf et al., 2017). The increasing empirical coverage of this model is intriguing, given its close association to modern minimalist syntax. Recently however, Zhang (2017) has argued that this framework is unable to account for a set of complexity profiles reported for English and Mandarin Chinese stacked relative clauses. Based on these observations, this paper proposes extensions to this model implementing a notion of *memory reactivation*, in the form of memory metrics sensitive to repetitions of movement features. We then show how these metrics derive the correct predictions for the stacked RC processing contrasts.

1 Introduction

This paper expands on a line of work investigating how a top-down parser for Minimalist grammars (MGs; Stabler, 2013) can be combined with complexity metrics to relate parsing behavior to memory usage, and successfully used to model off-line processing preferences (Kobele et al., 2013; Gerth, 2015; Graf et al., 2017). As MGs are close to modern syntactic theory, this model allows us to investigate the psycholinguistic effects of fine-grained grammatical details. Since one of the essential aspects of the MG approach is the interpretability of the linking hypothesis, it is crucial to probe the empirical coverage of the model. If its predictions prove to be consistently sound, the MG model could be used to contribute experimental insights to the development of grammatical theories (Miller and Chomsky, 1963; Bresnan, 1978; Joshi, 1990, a.o.).

In this sense, while the MG model has been used to account for a variety of phenomena cross-linguistically, Zhang (2017) has recently argued that the existing implementation seems unable to reproduce the complexity profiles she found for the processing of *stacked relative clauses* (RCs) in English and Mandarin Chinese.

Consider the following example of a stacked RC construction, in which a noun phrase (*the reporter*) is modified by two relative clauses — an Object RC (ORC; RC_1), and a Subject RC (SRC; RC_2):

(1) The reporter [$_{RC_1}$ who the senator attacked __ last year] [$_{RC_2}$ who __ received the Pulitzer yesterday] is facing a public trial.

In such cases, the parser has to resolve a dependency between *the reporter*, and two integration sites within the RCs: one in object position (RC_1), and one in subject position (RC_2).

In order to investigate the processing profile for this kind of constructions, Zhang (2017) conducted a series of self-paced reading experiments showing that English and Mandarin Chinese stacked relatives are processed faster when RC_1 and RC_2 are of the same type (e.g., both ORCs, as in 2) than when they are of different types (as in 1).

(2) The reporter [$_{RC_1}$ who the senator attacked __last year] [$_{RC_2}$ who the actor pushed __ yesterday] is n facing a public trial.

The intuition here seems to be that seing the first RC induces *facilitatory effects* in processing the second RC, when these have the same underlying syntactic structure. According to Zhang, no existing metric can account for this *parallelism* effect. This is not surprising *per se*, as the MG parser does not keep track of the relation between structurally independent clauses. However, this result is made even more compelling by the fact that the MG parser has been strikingly successful in accounting for asymmetries in the processing of RCs cross-linguistically (Graf et al., 2017; De Santo, 2019, a.o.).

Following insights from the psycholinguistic literature on *syntactic priming*, in this paper we

propose a non-probabilistic extension to the MG model that is able to account for effects of structural repetition on memory burden. To do so, the model will consider how successive occurrences of identical movement types might affect overall memory costs.

2 MG Parsing

MGs (Stabler, 1996, 2011) are a lexicalized formalism, in which a grammar is a set of lexical items (LIs) consisting of a phonetic form and a finite, non-empty string of features. LIs are assembled via two feature checking operations: *Merge* — encoding subcategorization — and *Move* — allowing for long-distance movement dependencies. The fundamental data structure in MGs is a *derivation tree*, which encodes the sequence of Merge and Move operations required to build the phrase structure tree for a given sentence (Michaelis, 1998; Harkema, 2001). In a derivation tree, all leaf nodes are labeled by LIs, while unary and binary branching nodes are labeled as Move or Merge, respectively.

Crucially, derivation trees differ from phrase structure trees in that they allow moved phrases to stay in their base position, as their landing site can be fully reconstructed from the feature calculus (cf. Fig. 1a and 1b). Because of this, MG derivation trees form a regular tree language, and thus allow us to exploit simple variants of established parsing algorithms for context-free grammars.

2.1 Top-Down MG Parsing

In this paper, we are specifically interested in Stabler (2013)'s top-down parser for MGs, as a variant of a standard depth-first, top-down parser for CFGs. This parser hypothesizes sentence structure top-down, verifies that the words in the structure match the input string, and outputs an encoding of the sentence in the form of a derivation tree. Obviously, direct top-to-bottom and left-to-right scanning of the leaf nodes yields the wrong word order, as the string yield of a derivation tree is not the phrase structure tree's surface order. To account for this, the MG parser keeps tracks of the derivational operations which affect the linear word order. This is memory's role: if a node is hypothesized at step i, but cannot be worked on until step j, it is stored for $j-i$ steps in a priority queue.

To make this traversal strategy easily accessible to the reader, we follow a notation introduced by Kobele et al. (2013). Practically, the annotation indicates for each node in the tree when it is first conjectured by the parser (*index*, the superscript) and placed in the memory queue, and at what point it is considered completed and flushed from memory (*outdex*, the subscript). Moreover, since the details of the feature calculus are mostly irrelevant to the model adopted here, we discard the features of each LI, and label internal nodes as standard in minimalist syntax. We also explicitly include dashed arrows indicating movement relations (cf. Fig. 1b and 1c). Finally, note that Stabler's parser is equipped with a search beam to navigate the parse forest. However, we want to focus exclusively on the effect of structure building on memory usage. Thus, we assume a deterministic parser equipped with a perfect oracle, which always makes the right choices when constructing a tree (Kobele et al., 2013).

2.2 Measuring Memory Usage

The behavior of Stabler (2013)'s MG parser has been linked to processing difficulty with complexity metrics measuring how building a derivation tree affects memory (Kobele et al., 2013; Gerth, 2015; Graf et al., 2017). Henceforth, we refer to this combination as the *MG model*. The model refers to memory usage as TENURE — how long a node is kept in memory — and SIZE — how much information is stored in a node (Rambow and Joshi, 1994; Gibson, 2000).

Tenure can be easily computed for a node n via the annotation schema of Kobele *et al.*, as the difference between its index and its outdex.[1] In Fig. 1c, *do* is introduced in memory at step 3, as soon as the parser predicts that C' should be expanded in a subtree. However, *do* cannot be scanned until the first word in the input (*who*) is found (at step 8 and 9). Thus, tenure for *do* is $10-3=7$.

Defining size in an intuitive way is slightly trickier, as it relies on how information about movers is stored by the top-down parser. For our purposes, size can be understood as measuring the hierarchical length of a movement dependency, computed as the index of a mover minus the index of its target site (for details, see Graf et al., 2015). Considering again the tree in Fig. 1c, the size of *who* is $8-1=7$.

These general concepts can then be used to define a vast set of complexity metrics measuring processing difficulty over a full tree, thus providing a system to contrast memory burden across derivations. For instance, tenure has been associated to metrics like MAXT $:= max(\{tenure\text{-}of(n)\})$ — the maximum amount of time any node stays in memory during processing. Consider again the tree in Fig. 1c. Tenure

[1] We refer to tenure values ≤ 2 as *trivial*, since it is not due to extra waiting time in the priority queue but to the binary nature of derivation trees (Graf and Marcinek, 2014).

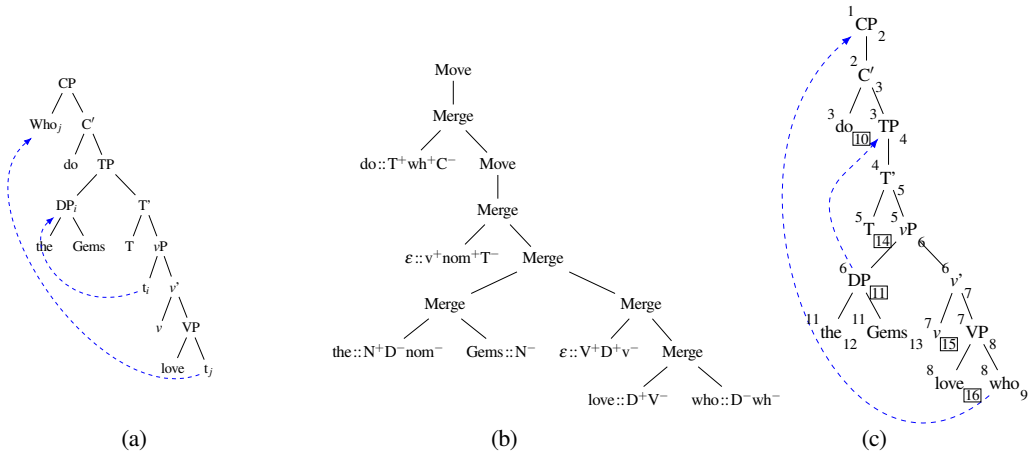


Figure 1: Phrase structure tree (a), full MG derivation tree (b), and simplified, annotated MG derivation for *Who do the Gems love?* Boxed nodes have tenure value greater than 2, following (Graf and Marcinek, 2014).

in this tree is mostly driven by the movement of the embedded object, thus MaxT is measured at T and it is equal to $14-5=9$. Similarly, we can define a size-based metric measuring the overall cost of movement dependencies across a derivation as SumS. In Fig. 1c SumS is given by the length of the object movement plus the length of the subject movement: $(8-1)+(6-3)=10$.

The metrics discussed so far are the most straightforward in quantifying memory usage with respect to the geometry of a derivation tree. However, it is possible to refine them to be sensitive to different aspects of the structure building process. For instance, intermediate movement steps do not affect the tree traversal strategy of the MG parser, and thus do not affect tenure-based metrics. For this reason, past work tends to ignore intermediate landing sites in computing memory metrics (Graf et al., 2015; De Santo, 2019). However, as size-based measures *do* vary depending on whether we consider intermediate movement steps or not, Graf and Marcinek (2014) propose a variant (e.g. SumS′) to take those into account.

Graf and Marcinek (2014) also define *recursive* variants — e.g. MaxT$_R$, listing the tenure of all nodes in a derivation in descending order. Recursive metrics are interesting in that they highlight the contrast between a derivation that has a high MaxT value just on a single node, versus a derivation that has the same identical high value, but on multiple nodes. For instance, take two derivations t_1 and t_2, such that MaxT$_R(t_1)=[15,5,5]$ and MaxT$_R(t_2)=[15,15,15]$. These two derivation trees receive exactly the same score under MaxT (15), and thus are predicted to be equally difficult. However, we can contrast the two derivations over MaxT$_R$, by a pointwise comparison of the two lists. MaxT$_R$ then predicts t_1 to be easier than t_2. Note that these variants can also be mixed, so that SumS$'_R$ is a recursive version of SumS that also takes intermediate movement steps into account.

Finally, Graf et al. (2015) introduce *ranked metrics* $\langle M_1, M_2, \ldots, M_n\rangle$, inspired by constraint ranking in Optimality Theory (Prince and Smolensky, 2008). Take the ranked metric $\langle$MaxT,SumS$\rangle$. As a lower ranked metric matters only if all higher ranked metrics have failed to pick out a unique winner, if two constructions result in a *tie* over MaxT, then their SumS values will decide which of the two is the winner.

At this point, the reader might concerned about the amount of possible metric combinations. However, the issue of potential empirical indeterminacy is addressed by searching for a restricted set of metrics that account for a variety of diverse phenomena across languages. Notably, there is increasing evidence that a ranked combination of $\langle$MaxT,SumS$\rangle$ accounts for a vast set of off-line processing contrasts cross-linguistically (Graf et al., 2017; Liu, 2018; Lee, 2018; De Santo, 2019; De Santo and Shafiei, 2019; De Santo, 2020). Given these promising results, it is important to investigate empirical limitations of the MG model. This is the focus of the rest of this paper.

3 Processing Stacked Relative Clauses

As mentioned before, stacked RCs are constructions in which a relativized noun phrase is modified by two relative clauses. While relative clauses have been focus of extensive experimental investigation cross-linguistically, not much work can be found on the comprehension of stacked RCs. To address this gap, Zhang (2017) explored their processing profile in English and Mandarin Chinese, in a 2×2 design

Language	Processing Contrast	Example #
English	$SS < OS$	3.a < 3.b
	$OO < SO$	3.d < 3.c
Mandarin	$SS < OS$	4.a < 4.b
	$OO < SO$	4.d < 4.c

Table 1: Summary of processing preferences for the stacked RCs effects evaluated in this paper. The first letter in each acronym indicates the type of the first RC, and the second letter the type of the second RC — i.e., SS stands for an *SRC* stacked above an *SRC*. $x < y$ means that x is *easier* to process than y.

crossing extraction type (subject or object) with the position of the RC (RC_1 or RC_2). Zhang reports faster reading times when RC_1 and RC_2 are of the same type (both SRCs or both ORCs), than when they are of different types (i.e. SS < OS and OO < SO; where $x < y$ means that x is *easier* to process than y, see Table 1). Crucially, she argues that the MG model would fail to capture these complexity profiles. In what follows, we first confirm this observation.

3.1 Modeling Assumptions

We test the MG model over stacked RC preferences, considering test cases for English as in (3), and Mandarin Chinese as in (4):

(3) a. The horse that kicked the elephant that chased the wolf left home **SS**

b. The horse that the elephant kicked that chased the wolf left home **OS**

c. The horse that kicked the elephant that the wolf chased left home **SO**

d. The horse that the elephant kicked that the wolf chased left home **OO**

(4) a. Nage tile xiaoma de
DEM kick-PERF horse REL
zhuile daxiang de gongniu
chase-PERF elephant REL bull
likaile jia
leave-PERF home
'The bull that kicked the horse that chased the elephant left home.' **SS**

b. Nage
DEM
xiaoma tile de zhuile
horse kick-PERF REL chase-PERF
daxiang de gongniu likaile jia
elephant REL bull leave-PERF home
'The bull that the horse kicked that chased the elephant left home.' **OS**

c. Nage tile
DEM kick-PERF
xiaoma de daxiang zhuile
horse REL elephant chase-PERF
de gongniu likaile jia
REL bull leave-PERF home
'The bull that kicked the horse that the elephant chased left home.' **SO**

d. Nage xiaoma
DEM horse
tile de daxiang zhuile
kick-PERF REL elephant chase-PERF
de gongniu likaile jia
REL bull leave-PERF home
'The bull that the horse kicked that the elephant chased left home.' **OO**

For reference, the first letter in each acronym indicates the type of the first RC, and the second letter the type of the second RC — for instance, SS stands for an *SRC* stacked above an *SRC*. Notably, while English RCs are post-nominal, they are pre-nominal in Mandarin Chinese.

As the MG model is sensitive to fine-grained structural information, we consider two RC analyses: a promotion (Kayne, 1994), and a wh-movement analysis (Chomsky, 1977). Importantly, both analyses assume that the syntactic derivation of RCs in Mandarin Chinese is complicated by a sequence of remnant movement operations necessary to arrive at the prenominal word order. Figure 2 shows an example of an annotated stacked RC derivation fed to the model. While the features triggering movement are not relevant to the current implementation of the MG metrics, the tree has feature-annotated movement arrows for reasons that will be clarified in Section 4.

3.2 Original Model: Results

To match Zhang (2017)'s experimental data, the parser should predict the set of processing preferences summarized in Table 1.[2] Confirming Zhang (2017)'s observations, the MG model is not able to account for the facilitatory effects found in stacked RCs. Notably, the main metric discussed in the past literature as a good predictor of processing difficulty for RCs cross-linguistically ($\langle$MaxT, SumS$\rangle$) correctly accounts

[2]Technically, the parser is also able to make predictions about all other conceivable comparisons. However, we limit the discussion to those that can be directly extracted from Zhang (2017)'s analysis.

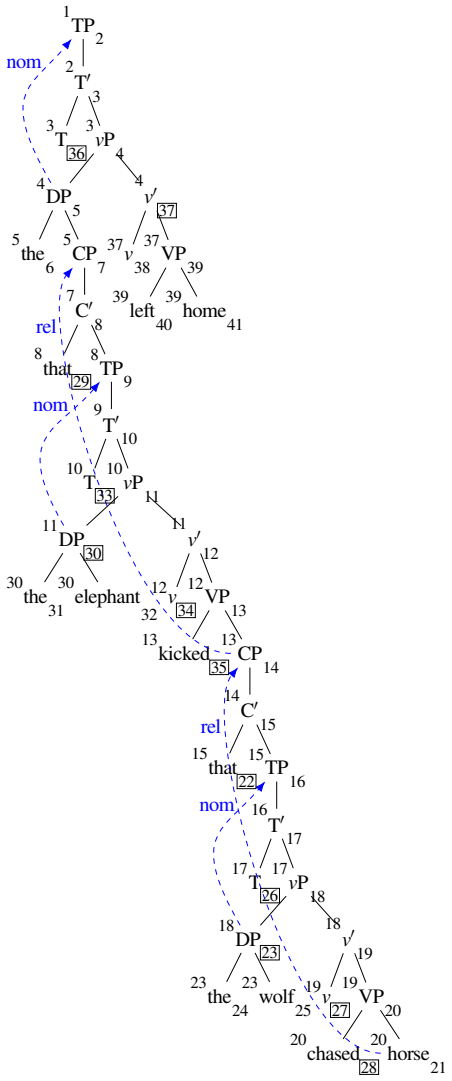


Figure 2: Annotated derivation for an English stacked RC (OO), built following the promotion analysis. Movement arrows are labelled with their respective Move feature.

for the SS $<$ OS contrast in both languages (Table 2). However, this seems to be an accident due to SUMS favoring a subject gap over an object gap, and in fact leads to the wrong prediction in the *OO* $<$ *SO* case. Importantly, these results hold both for English and Mandarin — thus independently of whether the RC is pre-nominal or post-nominal. As the end goal is to provide a model of sentence processing consistent across phenomena *and* languages, the MG approach in its current form remains unsatisfactory.

4 Feature Sensitive Metrics & Memory Reactivation

Zhang (2017) argues that the preference for symmetrical (SS, OO) constructions over asymmetrical (SO, OS) ones derives from similar structural configurations being observed twice in a row — processing

		OO $<$ *SO*	SS $<$ OS
English	MAXT	Tie	Tie
	SUMS	×	✓
Mandarin	MAXT	Tie	Tie
	SUMS	×	✓

Table 2: Summary of the performance of ⟨MAXT, SUMS⟩ on staked RCs in Mandarin Chinese and English under a promotion *and* a wh-movement analysis.

the underlying structure for the first RC *reduces* the processing load for the second RC. Failure on these effects is then unsurprising, as the existing metrics are *by design* unable to account for effects due to similar structural configurations being built twice in a row.

Interestingly, Zhang (2017)'s account of these processing profiles mirrors a wider psycholinguistic phenomenon known as *syntactic priming*. In particular, activation-based accounts (Troyer et al., 2011; Reitter et al., 2011) of structural priming effects suggest that *residual memory activation* of a previously encountered syntactic structure leads to short-term priming effects (Pickering and Branigan, 1998).

Recall now that, while MGs are a rich, lexicalized encoding of current minimalist analyses, the existing literature on MG parsing has consciously adopted metrics that ignore the feature-based component of MG trees. However, in MGs the features carried by a lexical item express all the information needed to reconstruct that item's argument structure. In fact, MG features explicitly capture the sequence of Merge and Move operations that the parser has to resolve to build a syntactic derivation. Thus, it should be possible to make the parser aware of structural operations that have recently taken place, by making it sensitive to recurring feature configurations. Residual activation approaches then give us a way to investigate priming phenomena with an MG parser, by reintegrating features into the MG derivations and introducing metric measuring *feature reactivation*. [3]

Following these ideas, this section explores different complexity metrics measuring a notion of reactivation as associated to movement features.

[3] Importantly, the fact that we take memory activation processes as an inspiration is not to be interpreted as a stance against other accounts of syntactic priming effects (for instance, implicit learning accounts (Bock and Griffin, 2000; Bock et al., 2007)). However, this approach is in line with the choice to ignore the effects of probabilistic information on processing difficulty — and, in fact, to put aside all other factors apart from purely structural ones.

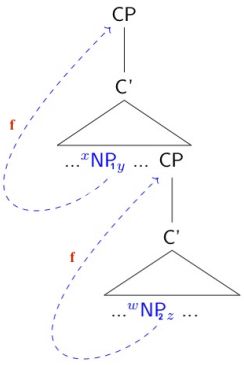


Figure 3: Example tree for memory reactivation.

4.1 Encoding Feature Reactivation

The first question to ask is, of course, how to encode reactivation in the MG parser. Here, we follow a procedure based on how the parser keeps track of movers.

Intuitively, movers are stored by the MG parser in specific memory cells, each dedicated to a particular movement type (i.e., feature). Movers triggered by the same feature are thus stored in the same cell. We assume now that if a memory cell has been inactive for a long time, storing a mover in that cell comes with a certain cost. However, if that memory cell has recently stored another mover, putting/maintaining the next mover into it should be less costly.[4]

This procedure is implemented by counting the number of parsing steps between movements of the same type, thus effectively accounting for the derivational time between two movers. Given Kobele et al. (2013)'s annotation schema, REACTIVATION (R) is computed as:

R For each node m_i associated to a movement feature f^-, reactivation is $i(m_i) - o(m_{i-1})$; the index of m_i minus the outdex of the closest preceding node associated to f^-, if it exists.

Consider the derivation in Figure 3, with two NP movers associated to a feature f. R for NP_2 is measured by subtracting from its index the outdex of the previous node associated to f (NP_1; $w-y$).

This definition of reactivation essentially indexes how costly it is to store some kind of movers compared to others, based on what has been previously observed during the course of the parse. Note, however, that reactivation is supposed to encode facilitatory effects induced by structural repetition. According to the definition above, there is no reactivation value assigned to movement features appearing for the first time. This might lead to issues in our comparative approach, as derivations without movement repetitions would have non-existent reactivation values — and thus, counterintuitively, might be evaluated as recruiting fewer memory resources. To account for that, REACTIVATION is operationalized as a metric as:

$$\mathrm{R}(m_i) := \begin{cases} 1 - \frac{1}{i(m_i) - o(m_{i-1})}, \text{ if } (i(m_i) - o(m_{i-1})) > 0 \\ 0, \text{ if } (i(m_i) - o(m_{i-1})) \leq 0 \\ 1, \text{ if } \neg\exists o(m_{i-1}) \end{cases}$$

In short, to guarantee that each node in the derivation is assigned a reactivation value, we assume that a movement node has an encoding cost of 1 — which can be lowered if a mover associated to the same feature had been previously seen (and has been removed from memory). If there is not such an antecedent (i.e., if that node is the first in the derivation to be associated to a movement feature of type f), reactivation is capped at 1. Once reactivation is defined, what remains to be explored is how it interacts with the original storage-based metrics in the MG literature. The insight that reactivation should directly impact the cost of maintaining a node in memory can be formalized in metrics that weight the tenure value of a node based on its feature reactivation. We implement this idea as follows. For each node m_i associated to a movement feature f, its BOOST (BT) is:

$$\mathrm{BT} := \textsc{Tenure}(m_i) * \mathrm{R}(m_i)$$

Since reactivation is meant to encode a facilitatory effect, values for BT will decrease as the number of similar movement dependencies in a derivation increases.

4.1.1 Defining Metrics

As for tenure and size, we define multiple metrics that compute reactivation and boost values over a full derivation. Let $\mathscr{M}$ be the set of movement features in a derivation tree T, and M^f the set of all nodes m associated to a feature $f^- \in \mathscr{M}$ in the string yield of T. Then, to evaluate the overall effect of reactivated nodes during the entire parse, we consider:

$$\textsc{SumR} := \sum_{f \in \mathscr{M}} \sum_{m_i \in M^f} R(m_i)$$

$$\textsc{MaxR} := max(\{max(\{R(m_i) | m_i \in M^f\} | f \in \mathscr{M})\})$$

[4] Note that this is different from the effect of having similar elements occupying a memory cell *at the same time* (Van Dyke and McElree, 2006; Jäger et al., 2015, a.o.). The reader might wonder how facilitatory effects due to memory reactivation interact with classical *interference* effects among multiple elements sharing similar features stored in memory. Recall though that we are only considering movement features, and that the SMC excludes cases in which multiple movers of the same type are in memory at the same time.

Load Type	
$\mathrm{T}(m_i)$	$o(m_i) - i(m_i)$
$\mathrm{S}(m_i)$	$i(m_i) - o(m_j)$, with m_j target of m_i
$\mathrm{R}(m_i)$	$1 - \frac{1}{i(m_i) - o(m_{i-1})}$
$\mathrm{BT}(m_i)$	$\textsc{Tenure}(m_i) * \mathrm{R}(m_i)$
Filtered Metrics	
M'	M with intermediate movement steps
M_R	applies M recursively
Metric Type	
MaxM	$max(\{M(m), \forall m \text{ in a derivation}\})$
SumM	$sum(\{M(m), \forall m \text{ in a derivation}\})$
AvgM	$avg(\{M(m), \forall m \text{ in a derivation}\})$

Table 3: Summary of memory load types.

$$\textsc{AvgR} := \frac{\textsc{SumR}}{\sum_{f \in \mathscr{M}} |M^f|}$$

SumBT, MaxBT, and AvgBT are computed for boost mirroring those defined for reactivation.

5 Testing Feature Reactivation[5]

As before, we compare the MG parser's performance of the *OO* vs. *SO* and *SS* vs. *OS* contrasts for both English and Mandarin Chinese, again under a promotion analysis (Kayne, 1994), and a wh-movement analysis (Chomsky, 1977) of RCs.

As pointed out above, syntactic choices are a crucial degree of freedom for the modeling approach. Extending the model with metrics sensitive to feature reactivation complicates this picture even further, as we will have to commit to a set of features consistent with the derivational operations posited by each analysis (Adger, 2003; Collins and Stabler, 2016). As this paper constitutes a first exploration of feature-based MG metrics, it seems reasonable to be as conservative as possible in the choice of feature overlap. Specifically, since reactivation metrics are restricted to movement operations, we only focus on whether particular movement steps could reasonably be triggered by the same feature, or not. For instance, we assume that movement of the head of the RC is triggered by the same feature in both subject and object RCs (see Appendinx A). When in doubt — i.e., in cases in which both interpretations are consistent with what discussed in the literature — we assume that movement was triggered by distinct features.

[5]Code and MG derivations for all simulations can be found at: https://github.com/aniellodesanto/mgproc_react.

5.1 Modeling Results

With all preliminaries in place, we can now look at how reactivation-based metrics perform on the processing of stacked RC. A reminder of the memory types introduced in this paper, and short-hand notations for filtered metrics is in Table 3.

In the vast space of possible metrics we found a reduced selection of ranked metrics which — depending of the syntactic analysis — lead to the correct predictions. With a promotion analysis, the correct contrasts are predicted by a metric ranking MaxR′ first: ⟨MaxR′, AvgBT⟩. This metric picks on the series of remnant movement dependencies in the construction of stacked RCs, but fails to work with a wh-movement analysis due to the whole RC movement increasing the tenure value that modulates AvgBT. In order to correctly account for the way movement dependencies are structured in the wh-movement analysis, it is necessary to rank a boost based metric high. Specifically, ⟨MaxBT, $\textsc{MaxR}'_R$⟩ makes the correct predictions in this latter cases. In what follows, we discuss the results by metric *rank*: rank 1 indicates metrics evaluated individually, while rank 2 refers to metrics evaluated as ranked (ordered) pairs, as discussed in Section 2.2.

Rank 1 Metrics First, we consider the performance of individual reactivation metrics (rank 1 metrics, e.g. MaxR). None of these metrics is able to account for the stacked RC contrasts both in English and in Mandarin. Precisely, a majority of the new metrics predicts the English processing profile correctly. Most of the same metrics also successfully account for the $OO < SO$ contrast in Mandarin. What these metrics are unable to capture is the Mandarin $SS < OS$ asymmetry, due to the pre-nominal nature of RCs in the language. These results hold independently of syntactic analysis. Things improve when we start looking at ranked metrics. Consistently with previous literature (and with the goal of keeping the space of possible metrics contained), we focus on metrics of rank 2.

Rank 2 Metrics For the promotion analysis, consider the performance of ⟨MaxR′, AvgBT⟩ (Table 5). AvgBT makes the correct prediction on every contrast, except the $SS < OS$ case in Mandarin. MaxR′ helps with this, as it ties on every contrast except the one missed by the boost based metrics. Note that this result depends on the *prime* variant MaxR′ — MaxR leads to predicting a tie in Mandarin. Thus, it looks like intermediate movement steps significantly contribute to deriving the correct processing profiles.

However, no metric ranking MaxR′ first succeeds

Language	Processing Contrast	⟨MaxR′, AvgBT⟩		⟨MaxBT, MaxR′$_R$⟩	
		Promotion	Wh-movement	Promotion	Wh-movement
English	$OO < SO$	✓	✓	✓	✓
	$SS < OS$	✓	×	✓	✓
Mandarin	$OO < SO$	✓	✓	✓	✓
	$SS < OS$	✓	✓	×	✓
English	$SRC < ORC$	✓	×	✓	✓
Mandarin	$ORC < SRC$	✓	✓	×	✓

Table 4: Summary of processing preferences for stacked RCs and single RCs effects by contrast and analysis, as predicted by ⟨MaxR′, AvgBT⟩ and ⟨MaxBT, MaxR′$_R$⟩.

when using a wh-movement analysis of RCs. For instance, under this analysis MaxR′ fails on the English $SS < OS$ contrast, predicting the opposite processing preference (Table 5). What seems to be needed to account for the structural relations in each language under this analysis, is to consider metrics ranking a max, boost-based metrics as their highest metric, and some recursive variant of R′ ranked lowest.

Table 5 illustrates the performance of ⟨MaxBT, MaxR′$_R$⟩. MaxBT makes the correct predictions for English, but fails to account for the $SS < OS$ contrast in Mandarin. MaxR′$_R$ is then necessary to discriminate in that case. Note that the recursive variant of MaxR′ is necessary, as MaxR′ by itself also predicts a tie on this contrast. Moreover, this metric (and, in fact, all metrics ranking MaxBT high) fails when considering a promotion analysis of RCs (Table 5).

			MaxR′	AvgBT
Promotion	English	$OO < SO$	Tie	✓
		$SS < OS$	✓	✓
	Mandarin	$OO < SO$	Tie	✓
		$SS < OS$	✓	×
Wh	English	$OO < SO$	Tie	✓
		$SS < OS$	Tie	×
	Mandarin	$OO < SO$	Tie	✓
		$SS < OS$	✓	×
			MaxBT	MaxR′$_R$
Promotion	English	$OO < SO$	✓	✓
		$SS < OS$	✓	×
	Mandarin	$OO < SO$	Tie	✓
		$SS < OS$	×	✓
Wh	English	$OO < SO$	✓	✓
		$SS < OS$	✓	×
	Mandarin	$OO < SO$	Tie	✓
		$SS < OS$	Tie	✓

Table 5: Individual performances of MaxR′ andAvgBT, and of MaxBT and MaxR′$_R$.

5.2 Additional Simulations

As a first step in a broader evaluation of reactivation metrics, we look at the processing asymmetry for single subject (SRC) and object (ORC) relative clauses in English and Mandarin. While English native speakers notoriously show a marked preference for SRC < ORC, conflicting evidence has been reported for Mandarin (Gibson and Wu, 2013, a.o.). In this respect, Zhang (2017)'s own experiments report an ORC < SRC preference in Mandarin. Here, we follow Zhang (2017), as we relied on her results for the stacked RC cases.

As shown in Table 4, the same metrics successful in the stacked RC cases also perform well on the single RC cases. Crucially, the SRC vs. ORC asymmetry has not been directly associated to memory reactivation effects in the psycholinguistic literature. Thus, it is surprising that metrics solely relying on such mechanism seem able to reproduce this contrast. Importantly, the metrics correctly replicate both the difference between the Mandarin and English preferences for single RCs, and their shared profiles for the stacked cases. For the promotion analysis, this is due to MaxR' picking up on the extra remnant movement steps necessary to the prenominal RC configuration, and leading to the Mandarin preference for an object gap in the single RC case. That preference is then subsumed by the general parallelism effects that arise in the stacked cases.

6 Conclusion

The MG parser enriched with feature reactivation successfully predicts the stacked RC contrasts reported for English and Mandarin, on two syntactic analyses of RCs. The idea that encoding something in memory comes at a cost has been well motivated by psycholinguistic insights (Van Dyke and McElree, 2006; McElree et al., 2003; McElree, 2006; Lewis et al., 2006; Villata et al., 2018), but has been lacking from the definitions of memory usage adopted by the MG approach. Overall then, these results are encouraging, as they demonstrate how the MG model can be refined to extend its empirical coverage, in ways that maintain (or even improve) the cognitive plausibility of its memory mechanisms.

While earlier MG processing results did not vary significantly depending on the choice of syntactic analysis, that is not the case for reactivation metrics. In future, it will crucial to understand why these results heavily depend on a very specific analysis. As mentioned, the number of possible metrics exponentially increases the degrees of freedom of the model. Following what was done for the set of original metrics, this issue can be addressed by careful empirical testing. By establishing a small number of metrics that transparently account for a variety of cross-linguistic asymmetries, it would then be possible to leverage experimental results to choose among competing syntactic choices (Rambow and Joshi, 1994; Kobele et al., 2013). This paper offered the first steps in this direction, by comparing the results on the new metrics both over the phenomenon that inspired them, and single RCs across structurally different languages. The obvious next step is an extensive evaluation of the performance of reactivation metrics over the full set of phenomena modeled by the MG model in the past. Moreover, the focus on stacked RCs was motivated by Zhang (2017) using them as an argument against existing formulations of the MG parsing model. A necessary step will be to move away from such a specific construction, and test the new metrics' performance over better attested cases of syntactic priming and parallelism effects (Pickering and Ferreira, 2008; Dubey et al., 2008, 2006; Mahowald et al., 2016). Finally, extending the definition of feature reactivation from Move features to Merge features would allow us to reproduce on a wider variety of sentence processing phenomena (e.g., interference effects; Van Dyke and McElree, 2006).

Acknowledgments

I would like to thank Thomas Graf, Mark Aronoff, John Baylin, and Jon Sprouse for their feedback on different stages of this research. I am also grateful to the anonymous reviewer for their constructive comments and insights.

References

David Adger. 2003. *Core syntax: A minimalist approach*, volume 33. Oxford: Oxford University Press.

Kathryn Bock, Gary S Dell, Franklin Chang, and Kristine H Onishi. 2007. Persistent structural priming from language comprehension to language production. *Cognition*, 104(3):437–458.

Kathryn Bock and Zenzi M Griffin. 2000. The persistence of structural priming: Transient activation or implicit learning? *Journal of experimental psychology: General*, 129(2):177.

Joan Bresnan. 1978. A realistic transformational grammar. *Linguistic theory and psychological reality*, pages 1–59.

Noam Chomsky. 1977. On wh-movement. *Formal syntax*, pages 71–132.

Chris Collins and Edward Stabler. 2016. A formalization of minimalist syntax. *Syntax*, 19(1):43–78.

Aniello De Santo. 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.

Aniello De Santo. 2020. MG parsing as a model of gradient acceptability in syntactic islands. *Proceedings of the Society for Computation in Linguistics*, 3(1):53–63.

Aniello De Santo and Nazila Shafiei. 2019. On the structure of relative clauses in Persian: Evidence from computational modeling and processing effects. In *Talk at the 2nd North American Conference in Iranian Linguistics (NACIL2)*, April 19-21 2019, University of Arizona.

Amit Dubey, Frank Keller, and Patrick Sturt. 2006. Integrating syntactic priming into an incremental probabilistic parser, with an application to psycholinguistic modeling. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 417–424, Sydney, Australia. Association for Computational Linguistics.

Amit Dubey, Frank Keller, and Patrick Sturt. 2008. A probabilistic corpus-based model of syntactic parallelism. *Cognition*, 109(3):326–344.

Sabrina Gerth. 2015. *Memory Limitations in Sentence Comprehension: A Structural-based Complexity Metric of Processing Difficulty*, volume 6. Universitätsverlag Potsdam.

Edward Gibson. 2000. The dependency locality theory: a distance-based theory of linguistic complexity. In *2000, Image, Language, Brain: Papers from the First Mind Articulation Project Symposium*, pages 95–126. MIT press.

Edward Gibson and H-H Iris Wu. 2013. Processing chinese relative clauses in context. *Language and Cognitive Processes*, 28(1-2):125–155.

Thomas Graf, Brigitta Fodor, James Monette, Gianpaul Rachiele, Aunika Warren, and 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)*, pages 1–14, Chicago, USA. Association for Computational Linguistics.

Thomas Graf and Bradley Marcinek. 2014. Evaluating evaluation metrics for minimalist parsing. In *Proceedings of the 2014 ACL Workshop on Cognitive Modeling and Computational Linguistics*, pages 28–36.

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

Henk Harkema. 2001. A characterization of minimalist languages. In *International Conference on Logical Aspects of Computational Linguistics*, pages 193–211. Springer.

Lena A Jäger, Felix Engelmann, and Shravan Vasishth. 2015. Retrieval interference in reflexive processing: experimental evidence from mandarin, and computational modeling. *Frontiers in psychology*, 6:617.

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

Richard S Kayne. 1994. *The antisymmetry of syntax*. 25. MIT Press.

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.

Richard L Lewis, Shravan Vasishth, and Julie A Van Dyke. 2006. Computational principles of working memory in sentence comprehension. *Trends in cognitive sciences*, 10(10):447–454.

Lei Liu. 2018. Minimalist Parsing of Heavy NP Shift. In *Proceedings of PACLIC 32 The 32nd Pacific Asia Conference on Language, Information and Computation*, The Hong Kong Polytechnic University, Hong Kong SAR.

Kyle Mahowald, Ariel James, Richard Futrell, and Edward Gibson. 2016. A meta-analysis of syntactic priming in language production. *Journal of Memory and Language*, 91:5–27.

Brian McElree. 2006. Accessing recent events. *Psychology of learning and motivation*, 46:155–200.

Brian McElree, Stephani Foraker, and Lisbeth Dyer. 2003. Memory structures that subserve sentence comprehension. *Journal of memory and language*, 48(1):67–91.

Jens Michaelis. 1998. Derivational minimalism is mildly context–sensitive. In *International Conference on Logical Aspects of Computational Linguistics*, pages 179–198. Springer.

George A. Miller and Noam Chomsky. 1963. Finitary models of language users. In R. Luce, R. Bush, and E. Galanter, editors, *Handbook of Mathematical Psychology*, volume 2. John Wiley, New York.

Martin J Pickering and Holly P Branigan. 1998. The representation of verbs: Evidence from syntactic priming in language production. *Journal of Memory and language*, 39(4):633–651.

Martin J Pickering and Victor S Ferreira. 2008. Structural priming: A critical review. *Psychological bulletin*, 134(3):427.

Alan Prince and Paul Smolensky. 2008. *Optimality Theory: Constraint interaction in generative grammar*. John Wiley & Sons.

Owen Rambow and Aravind K Joshi. 1994. A processing model for free word order languages. *Perspectives on Sentence Processing*.

David Reitter, Frank Keller, and Johanna D Moore. 2011. A computational cognitive model of syntactic priming. *Cognitive science*, 35(4):587–637.

Edward P 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. In *The Oxford Handbook of Linguistic Minimalism*.

Edward P Stabler. 2013. Two models of minimalist, incremental syntactic analysis. *Topics in cognitive science*, 5(3):611–633.

Melissa Troyer, Timothy J O'Donnell, Evelina Fedorenko, and Edward Gibson. 2011. Storage and computation in syntax: Evidence from relative clause priming. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33.

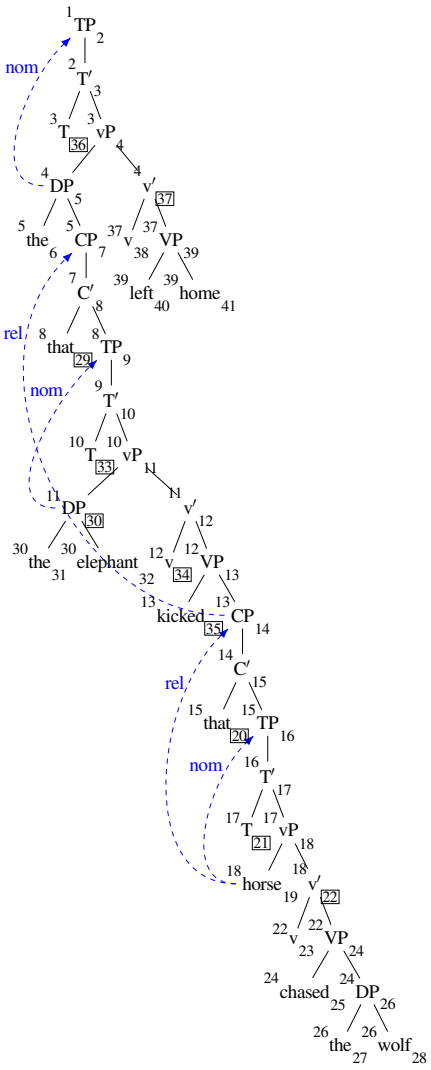
Julie A Van Dyke and Brian McElree. 2006. Retrieval interference in sentence comprehension. *Journal of Memory and Language*, 55(2):157–166.

Sandra Villata, Whitney Tabor, and Julie Franck. 2018. Encoding and retrieval interference in sentence comprehension: Evidence from agreement. *Frontiers in psychology*, 9:2.

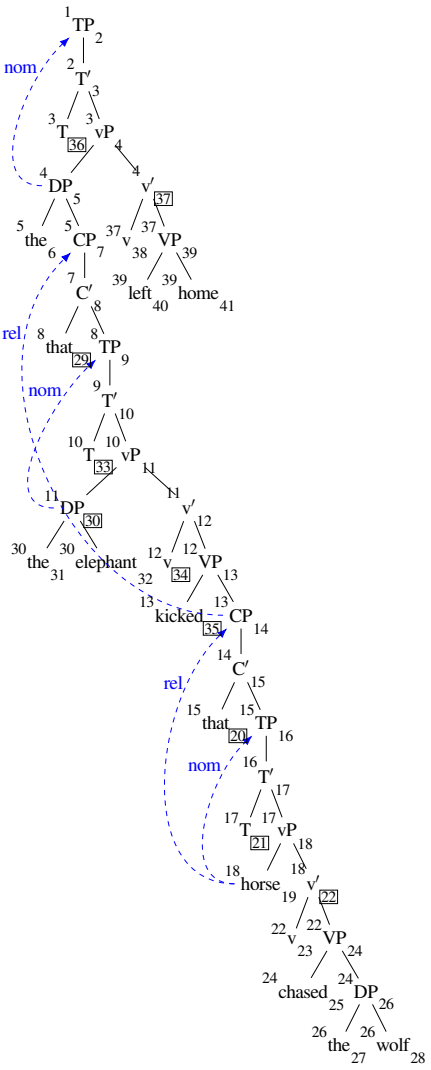
Chong Zhang. 2017. *Stacked Relatives: Their Structure, Processing and Computation*. Ph.D. thesis, State University of New York at Stony Brook.

A Appendix: Full Set of Derivation Trees

This appendix contains the full set of annotated derivation trees for Stacked RC in English and Mandarin Chinese used as input to the MG parser.

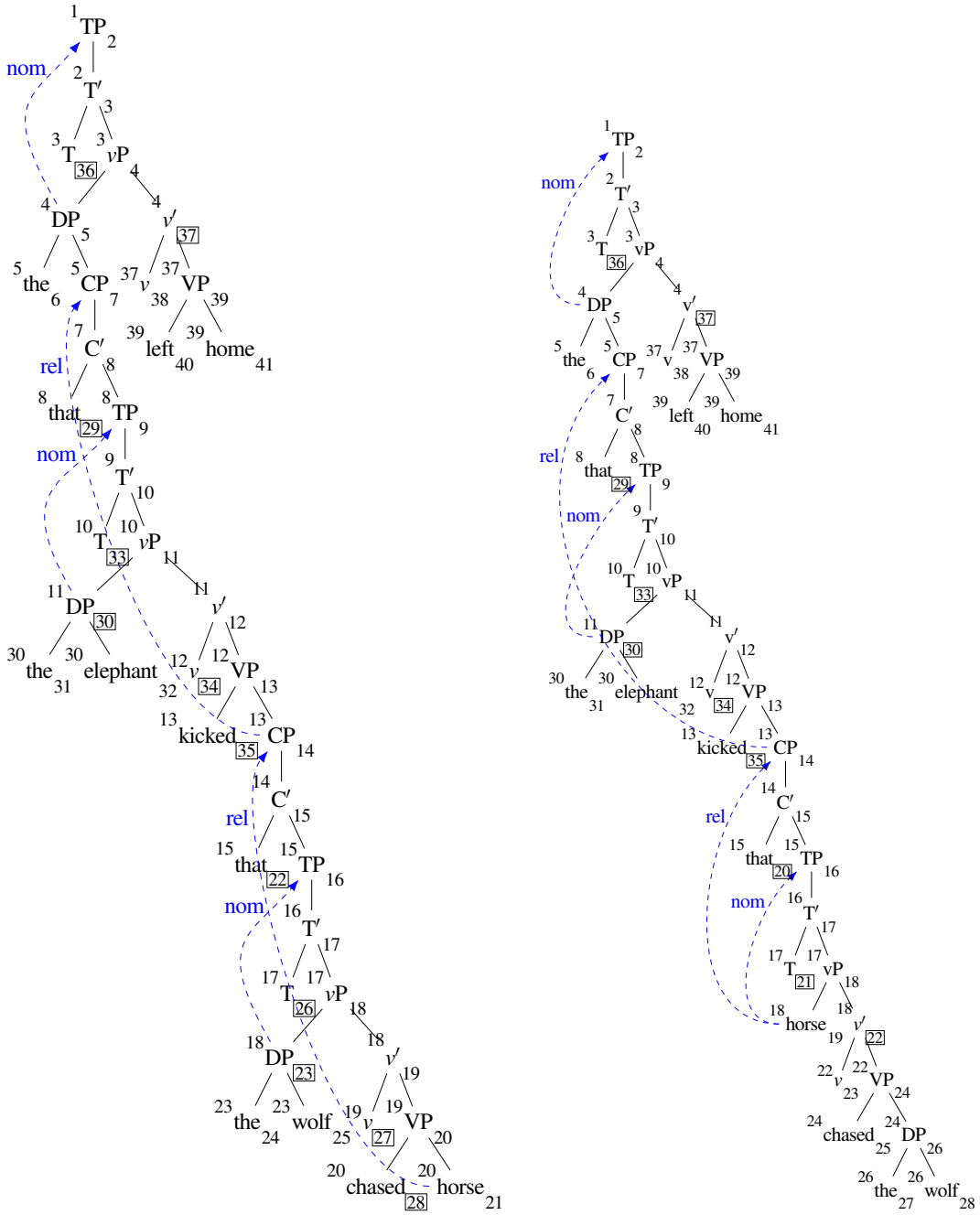


(a) SS



(b) OS

Figure 4: Annotated English stacked RC (SS vs OS), built following the promotion analysis.



(a) OO (b) SO

Figure 5: Annotated English stacked RC (OO vs SO), built following the promotion analysis.

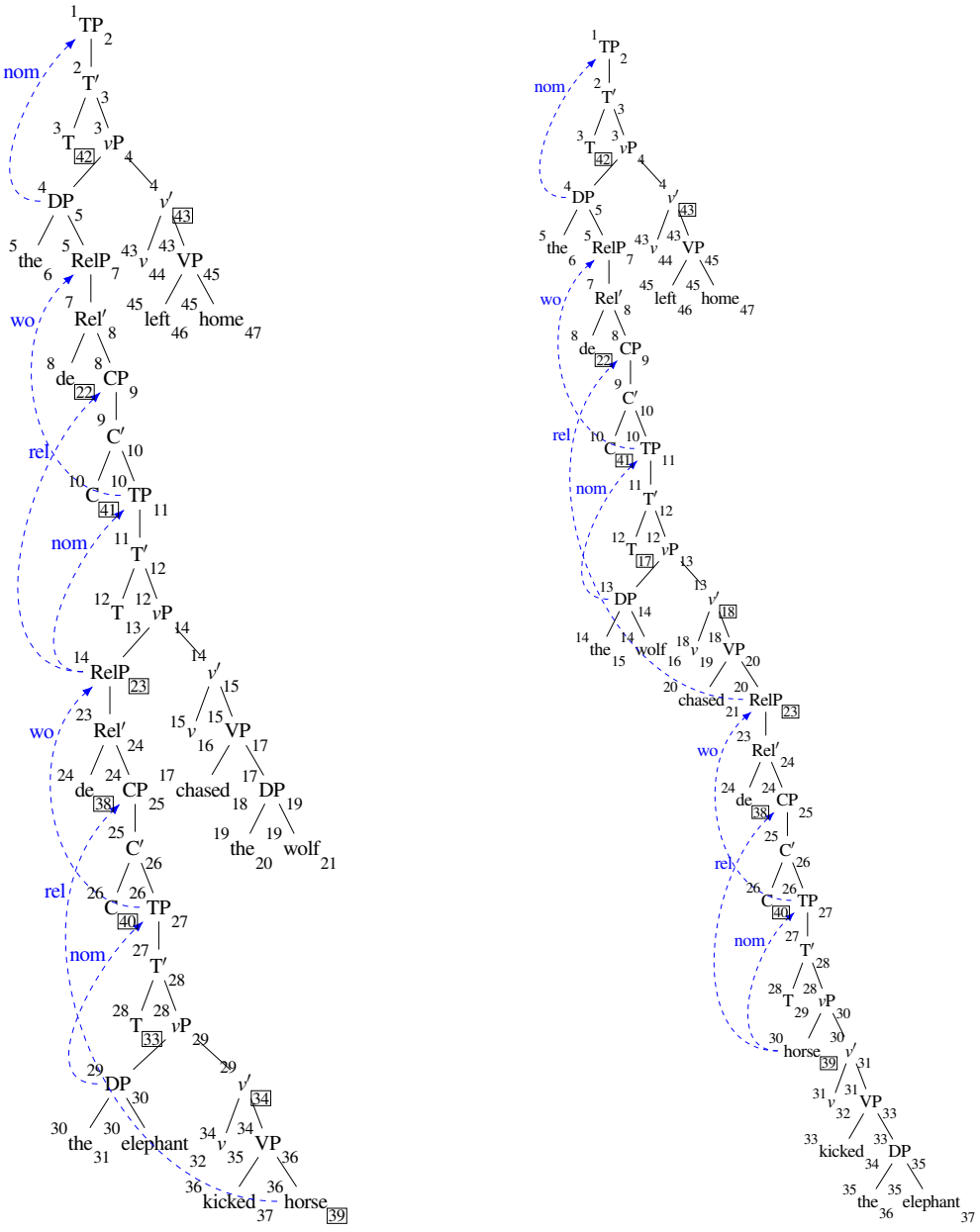
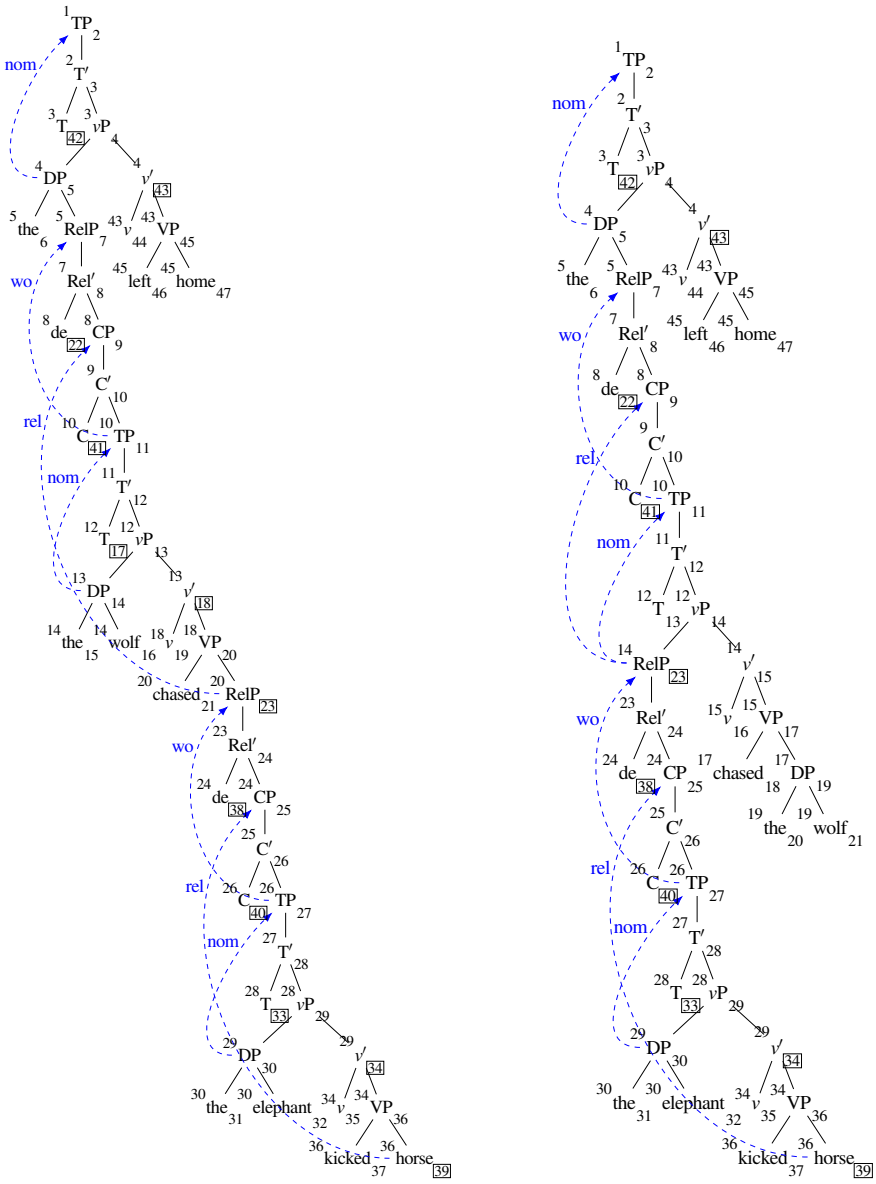
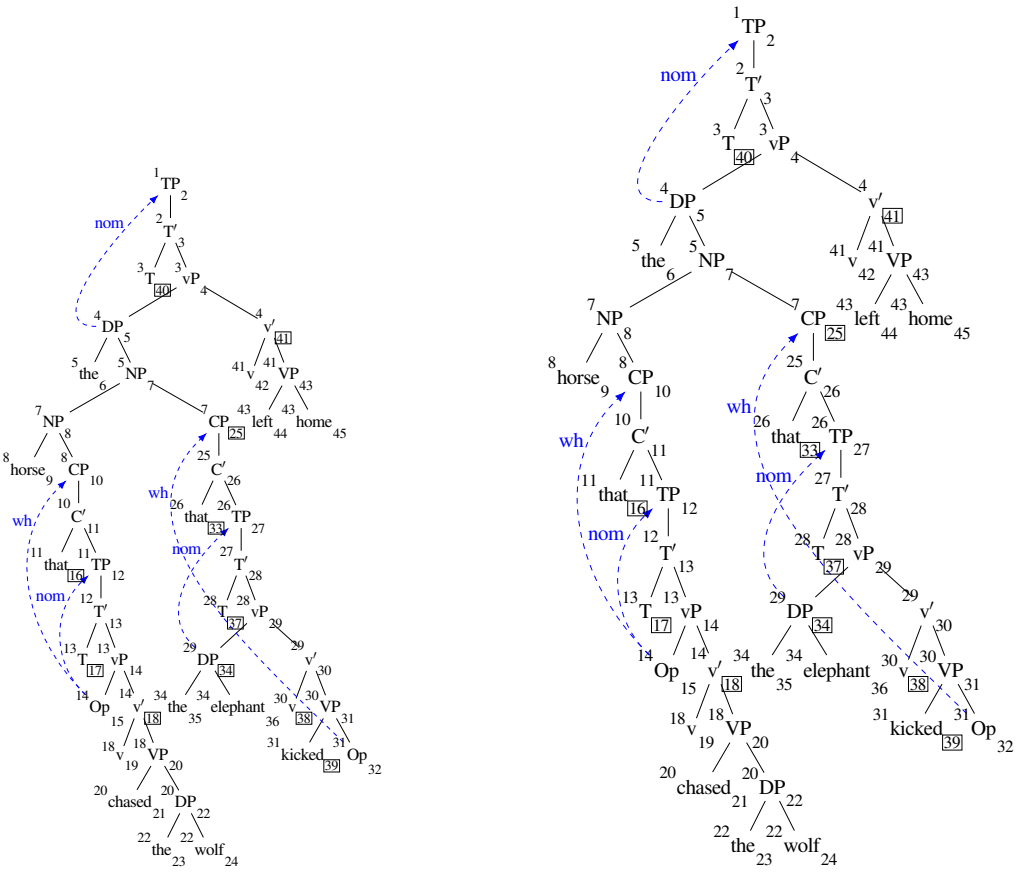


Figure 6: Annotated Mandarin stacked RC (SS vs OS), built following the promotion analysis.



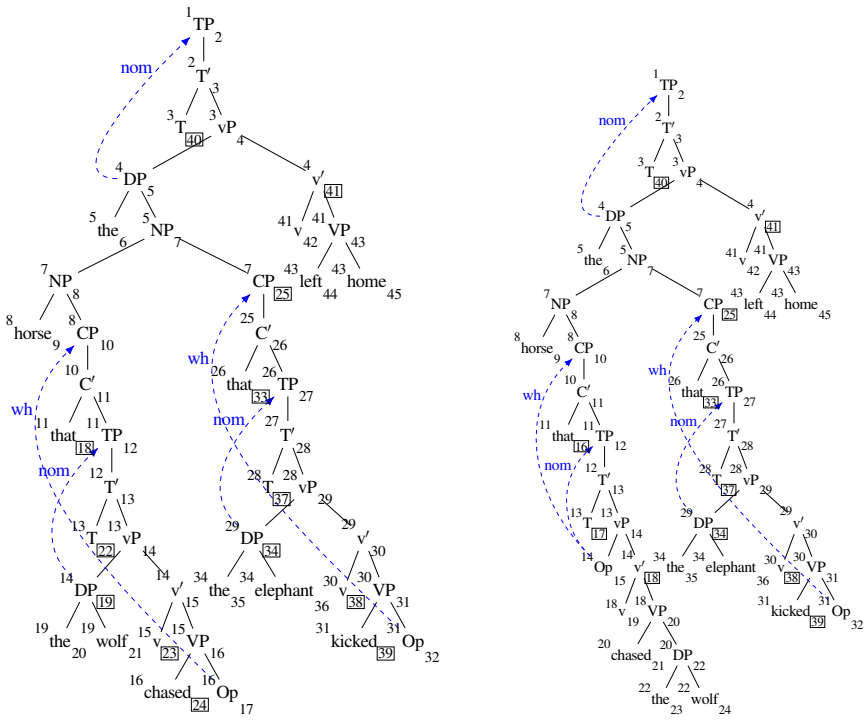
(a) OO (b) SO

Figure 7: Annotated Mandarin stacked RC (OO vs SO), built following the promotion analysis.



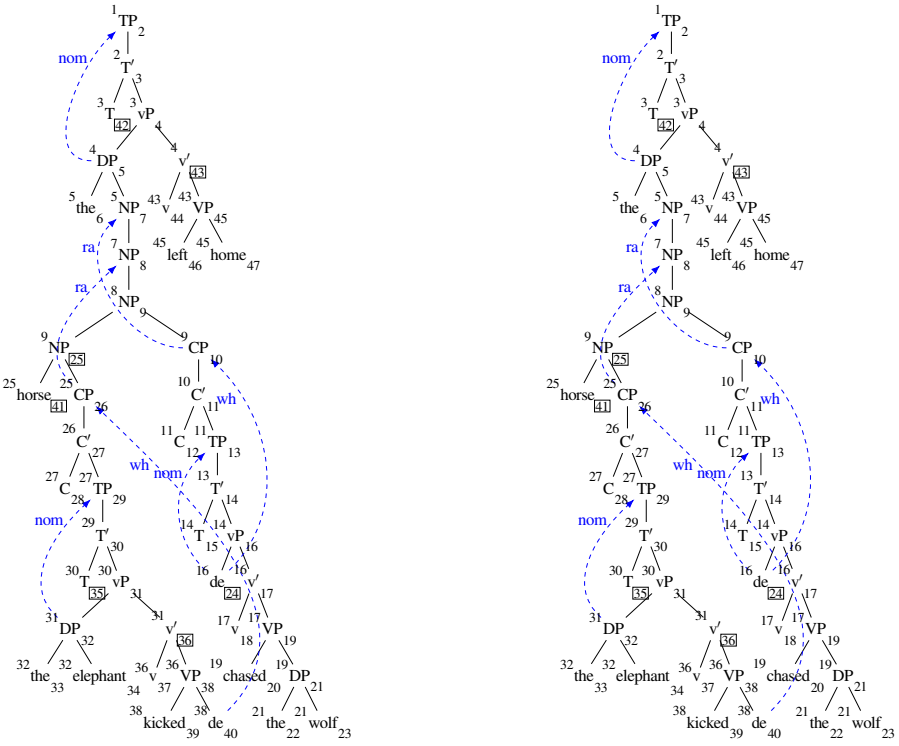
(a) SS (b) OS

Figure 8: Annotated English stacked RC (SS vs OS), built following the wh-movement analysis.



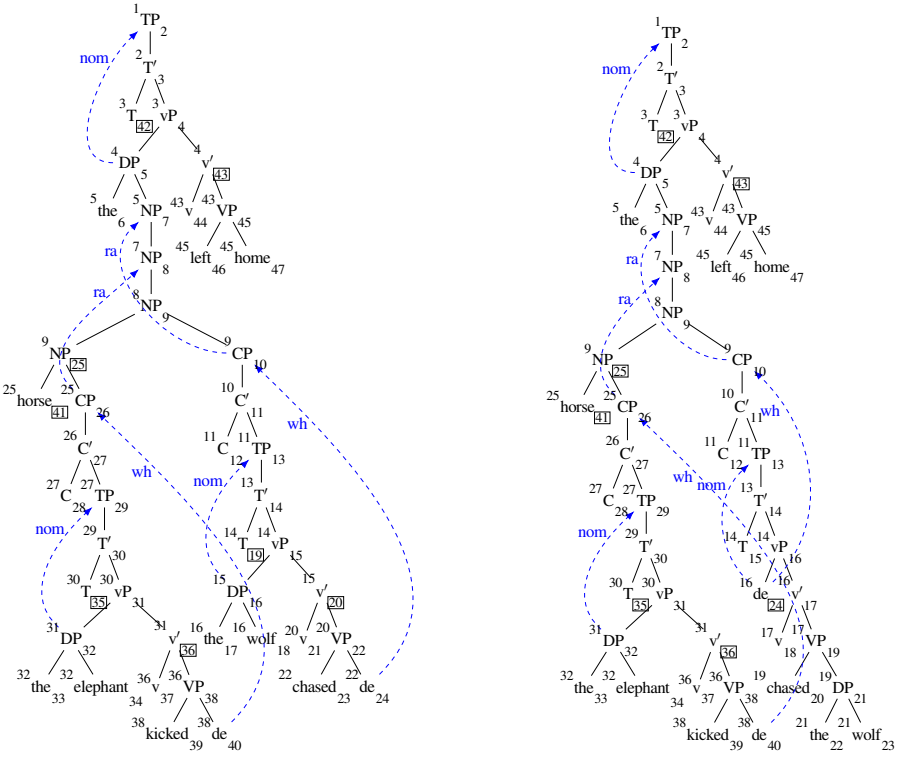
(a) OO (b) SO

Figure 9: Annotated English stacked RC (OO vs SO), built following the wh-movement analysis.



(a) SS (b) OS

Figure 10: Annotated Mandarin stacked RC (SS vs OS), built following the wh-movement analysis.



(a) SS (b) OS

Figure 11: Annotated Mandarin stacked RC (OO vs SO), built following the wh-movement analysis.