# The Active-Filler Strategy in a Move-Eager Left-Corner Minimalist Grammar Parser

**Tim Hunter**
Department of Linguistics
UCLA
timhunter@ucla.edu

**Miloš Stanojević**
School of Informatics
University of Edinburgh
m.stanojevic@ed.ac.uk

**Edward P. Stabler**
Samung Research America
California, USA
stabler@ucla.edu

## Abstract

Recent psycholinguistic evidence suggests that human parsing of moved elements is 'active', and perhaps even 'hyper-active': it seems that a leftward-moved object is related to a verbal position rapidly, perhaps even before the transitivity information associated with the verb is available to the listener. This paper presents a formal, sound and complete parser for Minimalist Grammars whose search space contains branching points that we can identify as the locus of the decision to perform this kind of active gap-finding. This brings formal models of parsing into closer contact with recent psycholinguistic theorizing than was previously possible.

## 1 Introduction

Minimalist Grammars (MGs) (Stabler, 1997, 2011) provide an explicit formulation of the central ideas of contemporary transformational grammar, deriving from Chomsky (1995). They have allowed formal insights into syntactic theory itself (Kobele, 2010; Kobele and Michaelis, 2011; Hunter, 2011; Graf, 2013), and there has been some work using MGs as the basis for psycholinguistic modeling. But this psycholinguistic work has focused primarily on sentence-processing at a relatively high level of abstraction, considering various measures of the workload imposed by different kinds of sentences — either information-theoretic metrics (Hale, 2003, 2006; Yun et al., 2015), or metrics based on memory load (Kobele et al., 2012; Graf and Marcinek, 2014; Brennan et al., 2016) — rather than the algorithmic-level questions of how derivations are pieced together incrementally.

A significant amount of experimental sentence-processing work aims to investigate exactly these kinds of algorithmic-level questions as they apply to long-distance syntactic dependencies, for example filler-gap dependencies between a moved wh-phrase and its base position. This is the kind of syntactic construction that MGs are particularly well-placed to describe (in contrast to simpler formalisms such as context-free grammars where parsing is well-studied), but it has been difficult to connect the experimental psycholinguistic work with any incremental, algorithmic-level MG parsing algorithms. Most parsing strategies proposed by psycholinguists have not been easy to relate to formal models of parsing.

## 2 Motivation and Background

A significant problem that confronts the human sentence-processor is the treatment of *filler-gap* dependencies. These are dependencies between a pronounced element, the *filler*, and a position in the sentence that is not indicated in any direct way by the pronunciation, the *gap*. A canonical example is the kind of dependency created by wh-movement, for example the one shown in (1).

(1) What did John buy ___ yesterday?

The interesting puzzle posed by such dependencies is that a parser, of course, does not get to "see" the gap: it must somehow determine that there is a gap in the position indicated in (1) on the basis of the properties of the surrounding words, for example the fact that 'what' must be associated with a corresponding gap, the fact that 'buy' takes a direct object, etc.

Experimental psycholinguistic work has uncovered a number of robust generalizations about how the human parsing system decides where to posit gap sites in amongst the pronounced elements as it works through a sentence incrementally. One conceivable strategy would be to posit gaps "only as a last resort, when all other structural hypotheses about that part of the sentence have been tried and



*Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 1–10
Minneapolis, USA, June 7, 2019. 

have failed" (Fodor, 1978, p.433). But the strategy that comprehenders actually employ is essentially to treat gaps as a "first resort", or what has become known as the "active filler" or "active gap-finding" strategy: hypothesize that there is a gap in any position where there might be one, and retract this hypothesis if subsequent input provides bottom-up evidence disconfirming it (Fodor, 1978; Stowe, 1986; Frazier and Clifton, 1989). Specifically, there is reason to believe that the dependency in (1) is constructed before the parser encounters 'yesterday'. A primary piece of evidence for this is the so-called "filled-gap effect": in a sentence like (2), we observe a reading slowdown at 'books' (Stowe, 1986).

(2) What did John buy books about ___ yesterday?

This slowdown is what one might expect if a dependency between 'what' and the object-position of 'buy' is constructed — actively, as a first resort — before the comprehender reads past 'buy', and then has to be retracted when 'books' is read. (What was hypothesized to be a gap position is in fact filled, hence "filled-gap effect".)

This basic generalization prompts a number of questions about the details of when and how this sort of hypothesizing of a gap takes place: in particular, one can ask what counts as a position where there "might be" a gap, and how this strategy interacts with the intricate grammatical constraints upon the relevant long-distance dependencies. See for example Traxler and Pickering (1996), Phillips (2006), Staub (2007), Wagers and Phillips (2009), and Omaki et al. (2015), among many others, for investigations of these issues.

At present it is difficult for the generalizations emerging from this experimental work to be framed in terms of the workings of a parser for contemporary transformational grammars. Consider for comparison the earlier empirical work on attachment preferences and garden path theory (e.g. Frazier and Clifton, 1996): since the focus was on grammatical relationships that were local in phrase-structural terms, the strategies being discovered could be understood as strategies for searching through the hypothesis space induced by the operations of a context-free parser. For example, the garden-path effect in (3) can be interpreted as evidence that given the locally ambiguous prefix 'When Fido scratched the vet', readers pursue the analysis in (4a) rather than the one in (4b). This is an instance of the Late Closure preference.

(3) When Fido scratched the vet (and his new assistant) removed the muzzle.

(4) a. When [$_S$ Fido scratched the vet] [$_S$ . . . ]
b. When [$_S$ Fido scratched] [$_S$ the vet . . . ]

Another way to put this is to say that after the word 'scratched', a bottom-up parser has the choice between performing a reduce step (to analyze this verb as a complete, intransitive VP) or performing a shift step (supposing that other remaining input will also be part of the VP), and it prefers the latter. See Figure 1, where the initial empty sequence of stack elements is indicated by $\epsilon$. If we suppose that the parser first explores the branch of the search space shown on the left in Figure 1, corresponding to the structure in (4a), then the disruption observed at the word 'removed' in (3) can be linked to the idea that this word triggers backtracking to the branching point shown in the diagram, so that the alternative intransitive-verb analysis in (4b) can be constructed by following the other branch.

In principle, it should be possible to give an analogous description of the active filler strategy for positing gaps: we can imagine a description of the parser's search space that allows us to state preferences for one kind of transition (the kind that interrupts "local processing" and posits a gap associated with an earlier filler) over another (the kind that continues working with local material). This is difficult at present, however, because there are relatively few formal models of parsing that treat both long-distance dependencies and local dependencies in a cohesive, integrated manner. Aside from this technical hurdle, however, the active filler strategy can be regarded as having the same form as the Late Closure preference: just as humans' first guess given the prefix 'When Fido scratched the vet' is (4a) rather than (4b), their first guess given the prefix 'What did John buy' is (5a) rather than (5b).

(5) a. What did John buy ___ . . .
b. What did John buy . . .

## 3 Minimalist Grammars

A Minimalist Grammar (Stabler, 1997, 2011) is defined with a tuple $G = \langle \Sigma, B, Lex, C, \{\text{MERGE,MOVE}\} \rangle$, where $\Sigma$ is the **vocabulary**, $B$ is a set of **basic features**, $Lex$

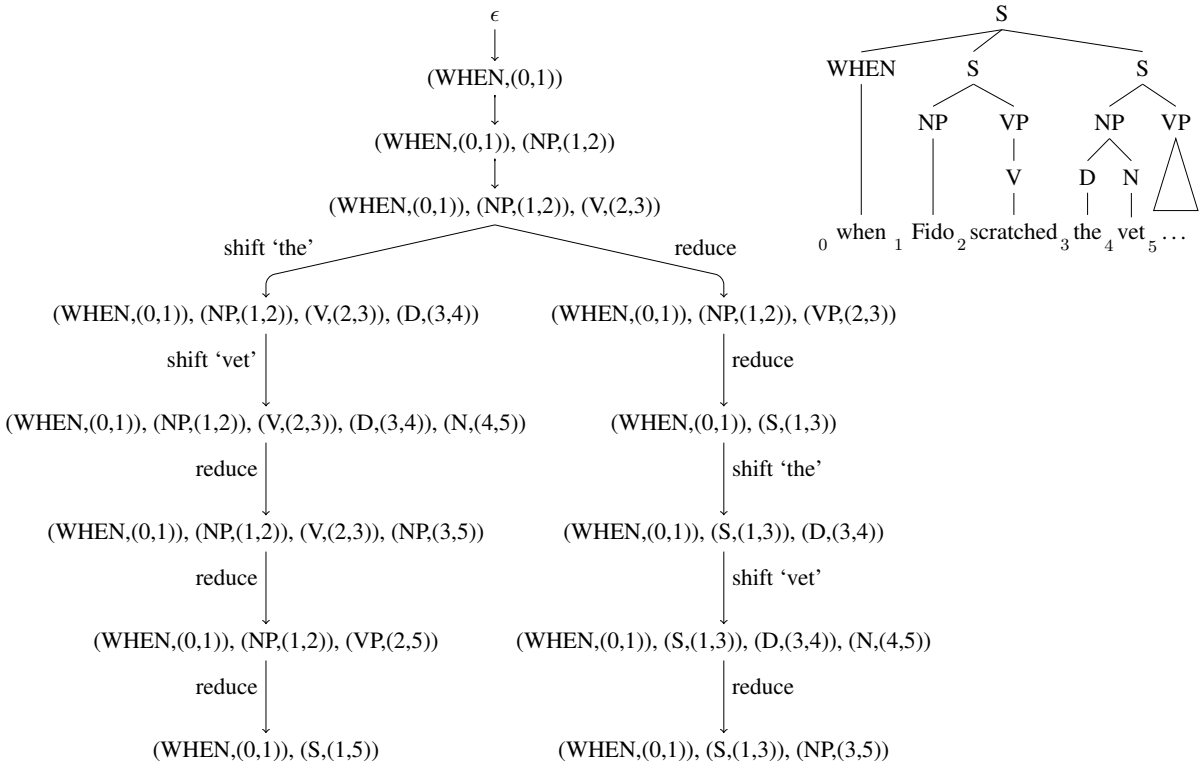


Figure 1: Part of the search space for the locally-ambiguous prefix '$_0$ When $_1$ Fido $_2$ scratched $_3$ the $_4$ vet $_5$' in a bottom-up shift-reduce parser. The left branch is the route favoured by Late Closure (Frazier and Clifton, 1996). The garden-path effect in (3) can be seen as a consequence of the reanalysis required when a parser searches this left branch first.

is a finite **lexicon** (as defined just below), $C \in B$ is the **start category**, and MERGE and MOVE are the generating functions. The basic features of the set $B$ are concatenated with prefix operators to specify their roles, as follows:

**categories, selectees** $= B$
**selectors** $= \{=f \mid f \in B\}$
**licensees** $= \{-f \mid f \in B\}$
**licensors** $= \{+f \mid f \in B\}$

Let $F$ be the set of role-marked **features**, that is, the union of the categories, selectors, licensors and licensees. Let $T = \{::, :\}$ be two **types**, indicating "lexical" and "derived" structures, respectively. Let $\mathbb{C} = \Sigma^* \times T \times F^*$ be the set of **chains**. Let $E = \mathbb{C}^+$ be the set of **expressions**; intuitively, an expression is a chain together with its "moving" sub-chains, if any. Finally, the **lexicon** $Lex \subset \Sigma^* \times \{::\} \times F^*$ is a finite set. The functions MERGE and MOVE are defined in Table 1. Note that each MERGE rule deletes a selection feature $=f$ and a corresponding category feature $f$, so the result on the left side of each rule has two features less than the total number of features on the right. Similarly, each MOVE rule deletes a licensor feature $+f$ and a licensee feature $-f$. The rules (understood as functions from right-to-left, or "bottom-up") have pairwise disjoint domains; that is, an instance of a right side of a rule is not an instance of the right side of any other rule. The set of all **structures** that can be derived from the lexicon is $S(G) = \text{closure}(Lex, \{\text{MERGE}, \text{MOVE}\})$. The set of **sentences** $L(G) = \{s \mid s \cdot C \in S(G) \text{ for some type } \cdot \in \{:, ::\}\}$, where $C$ is the "start" category.

Two simple derivations are shown in Figures 2 and 3. These trees have elements of the grammar's lexicon (not shown separately) at their leaves. At each binary-branching node we write the structure that results from applying MERGE to the structures at the daughter nodes; and at each unary-branching node we write the structure that results from applying MOVE to the structure at the daughter node.

The lowest MERGE step shown in Figure 2, for example, combines (via MERGE3, specifically) the lexical items for 'buy' and 'what'; the `d` category feature on 'what' can satisfy the first of the `=d` se-

**merge** is the union of the following 3 rules, each with 2 elements on the right,
for strings $s, t \in \Sigma^*$, for types $\cdot \in \{:, ::\}$ (lexical and derived, respectively),
for feature sequences $\gamma \in F^*$, $\delta \in F^+$, and for chains $\alpha_1, \ldots, \alpha_k, \iota_1, \ldots, \iota_l$ $(0 \leq k, l)$

(MERGE1) lexical item $s$ selects non-mover $t$ to produce the merged $st$

$$st : \gamma, \alpha_1, \ldots, \alpha_k \quad \rightarrow \quad s :: =f\gamma \qquad t \cdot f, \alpha_1, \ldots, \alpha_k$$

(MERGE2) derived item $s$ selects a non-mover $t$ to produce the merged $ts$

$$ts : \gamma, \alpha_1, \ldots, \alpha_k, \iota_1, \ldots, \iota_l \quad \rightarrow \quad s : =f\gamma, \alpha_1, \ldots, \alpha_k \qquad t \cdot f, \iota_1, \ldots, \iota_l$$

(MERGE3) any item $s$ selects a mover $t$ to produce the merged $s$ with chain $t$

$$s : \gamma, \alpha_1, \ldots, \alpha_k, t : \delta, \iota_1, \ldots, \iota_l \quad \rightarrow \quad s \cdot =f\gamma, \alpha_1, \ldots, \alpha_k \qquad t \cdot f\delta, \iota_1, \ldots, \iota_l$$

**move** is the union of the following 2 rules, each with 1 element on the right,
for $\delta \in F^+$, such that none of the chains $\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k$ has $-f$ as its first feature:

(MOVE1) final move of $t$, so its $-f$ chain is eliminated on the left

$$ts : \gamma, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k \quad \rightarrow \quad s : +f\gamma, \alpha_1, \ldots, \alpha_{i-1}, t : -f, \alpha_{i+1}, \ldots, \alpha_k$$

(MOVE2) nonfinal move of $t$, so its chain continues with features $\delta$

$$s : \gamma, \alpha_1, \ldots, \alpha_{i-1}, t : \delta, \alpha_{i+1}, \ldots, \alpha_k \quad \rightarrow \quad s : +f\gamma, \alpha_1, \ldots, \alpha_{i-1}, t : -f\delta, \alpha_{i+1}, \ldots, \alpha_k$$

Table 1: **Rules for minimalist grammars** from Stabler 2011, §A.1.

lectors on 'buy', and these two features are deleted in the resulting structure. This resulting structure, like the two above it, consists of two chains: as well as the chain[1] that participates "as usual" in the structure-building steps of combining with the subject and silent complementizer, there is the chain 'what : −wh' representing the wh-element that is "in transit" throughout these steps of the derivation. Given this separation of a structure into its component chains, movement amounts to bringing together two chains. The (formally redundant) dashed line in the figure links the MOVE step at the root of the derivation to the structure that gave rise to the 'what' chain that this MOVE step acts on. The last two steps of the derivation effectively *wrap* (Bach, 1979) the components 'John buys' and 'what' around the (as it happens, silent) complementizer.

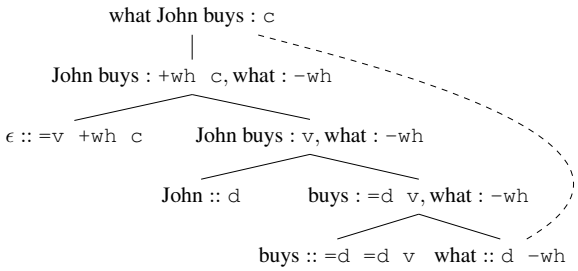


Figure 2: Example derivation for: what John buys

## 4 Previous MG Parsers

Stabler (2013) presented the first systematic generalization of incremental/transition-based CFG parsing methods to MGs, specifically a top-down MG parser. This requires a complete root-to-leaf path to a lexical item before it can be scanned, and therefore only allows a filler (e.g. a wh-phrase) to be consumed once we commit to a particular position for the corresponding gap (e.g. matrix subject position, matrix object position, embedded subject position, etc.). In terms of the tree diagrams like Figures 2 and 3, both the solid-line connection from the root down to a wh-phrase and the dashed-line connection are established before the wh-phrase can be consumed. The ambiguity-resolution question raised by filler-gap dependencies therefore amounts to a choice between competing analyses that diverged before the filler was consumed, rather than a choice of how to extend a particular analysis like in Figure 1. See Hunter (in press) for more detailed discussion.

Stanojević and Stabler (2018) adapt the idea of left-corner parsing from CFGs to MGs. This parser can consume a wh-filler without committing to a particular gap site for it, and therefore — unlike the Stabler (2013) parser — there is a single sequence of steps that it can take to parse a prefix such as 'What does John think' which can be extended with either a subject-gap or object-

[1] In traditional terminology, this first chain happens to be a trivial or one-membered chain, i.e. one that does not undergo any movement.

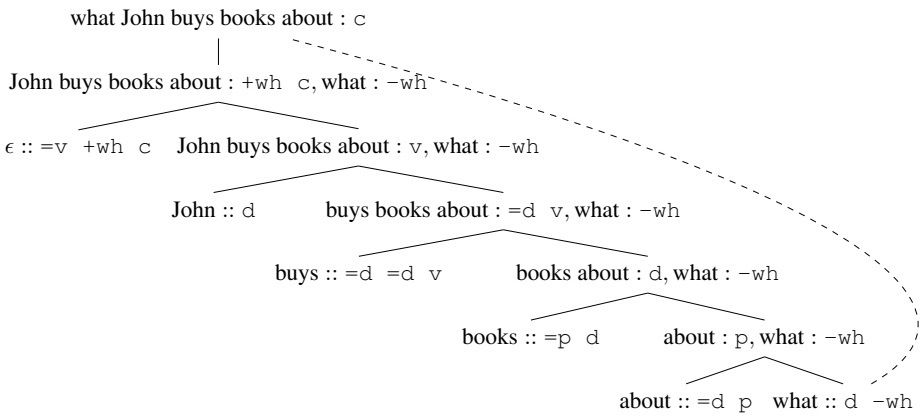


Figure 3: Example derivation for: what John buys books about

gap structure.[2] But it does this without identifying a "filler site" for the wh-phrase either: the wh-phrase, in effect, remains entirely disconnected from the rest of the structure until its gap site is encountered, then the rest of the clause(s) out of which the wh-phrase moves is assembled, and only then is the wh-phrase slotted into its surface position as part of the linking of this clause into its surroundings. In terms of the tree diagrams: while this parser does allow the solid-line connection from the root down to a wh-phrase to be unknown when the wh-phrase is scanned, it constructs the dashed-line connection only after this solid-line connection is eventually established.

The goal here is to adjust the parsing mechanisms of Stanojević and Stabler (2018) so that they produce a search space where the choice points are more in line with the psycholinguistic literature's framing of the choices that confront the human sentence-processing mechanism regarding filler-gap dependencies. With respect to the tree diagrams, we would like a parser that can establish the dashed-line connection down to a wh-phrase at the point where the wh-phrase is consumed, and delay the solid-line connection until later.

## 5 Move-Eager Left-Corner MG Parsing

We maintain an input buffer and a store. Each item in the store is either an element of the form ((start index, end index) · category), or an implication (written with ⇒) from one such element to another. There is a distinguished "top" item in the store; additional items are unordered. We begin with an implication $((0, n) \cdot \texttt{c}) \Rightarrow ((0, n)\ \text{ROOT})$ in the store, where c is the starting category of the grammar and 'ROOT' is a distinguished grammar-external symbol.

A SHIFT transition consumes a word from the buffer and puts a corresponding element $((i, i+1) :: X)$ into the top position in the store, or $((i, i) :: X)$ in the case of shifting an empty string.

We define the other parsing transitions in terms of the five MG grammatical rules in Table 1.

If $R$ is a binary grammatical rule $A \rightarrow B\ C$ and we have $B$ at the top of our store, then the transition relation $\text{LC}(R)$ allows us to replace this $B$ with the implication $C \Rightarrow A$; or, if we have $C$ at the top of our store, then $\text{LC}(R)$ allows us to replace $C$ with the implication $B \Rightarrow A$. The idea in the latter case is that, since we have already found a $C$, finding a $B$ in the future is now all that we need to do to establish an $A$. This is familiar from left-corner CFG parsing, and forms the core of how MERGE steps are parsed (since the MERGE rules are the binary rules). For example, if we have found a preposition spanning from position $i$ to position $j$, i.e. $((i, j) :: \texttt{=d p})$, then LC(MERGE1) allows us to replace this with an implication $((j, k) \cdot \texttt{d}) \Rightarrow ((i, k) : \texttt{p})$. The right side of this implication has type ':', since it is necessarily non-lexical; the type of the left side is unspecified ($\cdot \in \{:, ::\}$).

Given an implication $X \Rightarrow Y$ somewhere in our store, a central idea from (arc-eager) left-corner parsing is that parsing steps that produce an $X$ can be *connected*, or chained together, with this stored implication to instead produce a $Y$ (and in this case we remove the implication from the store). We can think of $X \Rightarrow Y$ as a fragment of tree

[2] Leaving aside questions of how movement dependencies are treated, left-corner parsing is also generally regarded as more psychologically plausible for reasons relating to the memory demands imposed by different kinds of embedding configurations in basic, movement-free structures (Resnik, 1992).

structure that has $Y$ at the root and has an "unfilled" $X$ somewhere along its frontier (or a *context*, a $Y$ tree with an $X$ hole); if there is a step we can take that can produce an $X$, that $X$ can be plugged in to the tree fragment.

For any parsing transition $T$, there are four variants $\textsc{c}0(T)$, $\textsc{c}1(T)$, $\textsc{c}2(T)$ and $\textsc{c}3(T)$ that connect, in slightly varying configurations, the items produced by $T$ itself with implications already in the store.

(6) a. If $T$ produces $B$ and we already have $B \Rightarrow A$, then $\textsc{c}0(T)$ produces $A$.

b. If $T$ produces $B \Rightarrow A$ and we already have $C \Rightarrow B$, then $\textsc{c}1(T)$ produces $C \Rightarrow A$.

c. If $T$ produces $C \Rightarrow B$ and we already have $B \Rightarrow A$, then $\textsc{c}2(T)$ produces $C \Rightarrow A$.

d. If $T$ produces $C \Rightarrow B$ and we already have $B \Rightarrow A$ and $D \Rightarrow C$, then $\textsc{c}3(T)$ produces $D \Rightarrow A$.

In all cases the relevant pre-existing implications are removed from the store. C0 connects a shifted lexical item with the antecedent of an implication, i.e. the "unfilled" slot at the bottom of some tree fragment. Rules $\textsc{c}1(T)$ and $\textsc{c}2(T)$ are similar to function composition, or the **B** combinatory rule of CCG (Steedman, 2000).[3] $\textsc{c}1(T)$ and $\textsc{c}2(T)$ differ from each other in whether it is the top or bottom of the fragment newly created by $T$ that connects with a pre-existing fragment; $\textsc{c}3(T)$ is for the more complicated cases where connections are made at *both* ends of the fragment created by $T$. See Figure 4.

The place where the parser presented here differs from that of Stanojević and Stabler (2018) is in the treatment of MOVE rules. These are treated as ways to "extend" the other parsing transitions. Given a grammar rule MOVE$n$ of the form $A \rightarrow B$, if a parsing transition $T$ produces an implication $C \Rightarrow B$, then $\textsc{mv}n(T)$ produces $C \Rightarrow A$.[4] (The parser of Stanojević and Stabler (2018), in contrast, would wait until $C$ is completed and we simply have $B$, at which point a standalone MOVE-transition would replace this with $A$.)

[3] Resnik (1992, p.197) emphasizes this relationship between arc-eager parsing's connect rules and function composition, and the analogy to CCG's function composition operation specifically.

[4] Note that whereas $\Rightarrow$ "points upwards" in the tree, $\rightarrow$ points downwards (cf. Table 1).

With these rules, we obtain a search space that better allows us to precisely express the active/greedy gap-finding strategies that the psycholinguistic evidence supports. This is illustrated by the traces shown in Figures 5-6.[5]

The first interesting step in Figure 5 is Step 2, which builds the MERGE3 step (i.e. the bottom application of MERGE in Figure 2 discussed earlier) on top of 'what' to produce an implication. Given the actual surroundings of 'what' in Figure 2, (the feature parts of) this implication would be `=d =d v` $\Rightarrow$ `=d v, -wh`.[6] But it could also be `=d`$\gamma \Rightarrow \gamma$, `-wh` for any other feature-sequence $\gamma$ (cf. Table 1), so the parser creates an implication where these additional features are left as variables to be resolved by unification later. This is the new store item shown in Step 2, where the start and end positions of the selector of 'what' are likewise unknown and left as variables $n_0$ and $n_1$, $\alpha_3$ is the first feature of $\gamma$ (which we actually know cannot be a licensor) and $\alpha_4$ is the rest of $\gamma$.[7]

The next step shifts the empty complementizer into the store. The resulting item has no variables, and spans from position 1 to position 1.

Step 4 is perhaps the most complex and interesting step. At its core is the fact that LC(MERGE1) constructs, from the silent complementizer whose features are `=v +wh c`, an implication from the its complement (features `v`, plus possible movers) to its parent (features `+wh c`, plus possible movers). But the right-hand side of this implication is something that MOVE1 can apply to; specifically, MOVE1 applied to `+wh c`, `-wh` produces `c`. So putting these together, MV1(LC(MERGE1)) produces an implication from `v,-wh` to `c`. And C2 can chain this together with the initial implication from `c` to ROOT, to produce an implication from `v,-wh` to ROOT as the end result. This new store

[5] An implementation using depth-first backtracking search is available at https://github.com/stanojevic/Move-Eager-Left-Corner-MG-Parser

[6] In these sequences of feature-sequences, spaces bind more tightly than commas.

[7] Leaving the other features of the wh-phrase's selector as variables allows us to remain completely agnostic about the base position of the wh-phrase. But a version of this parser that did not do this would still avoid the problem for the top-down MG parser discussed in Section 4: it would commit to the *immediate* surroundings of the wh-phrase's base position (for which there are only finitely many options) before moving on from consuming the filler, but it would remain agnostic about how far this surrounding material is from the root of the tree. Committing to the immediate surroundings of the wh-phrase would not be unnatural in languages with rich case-marking.

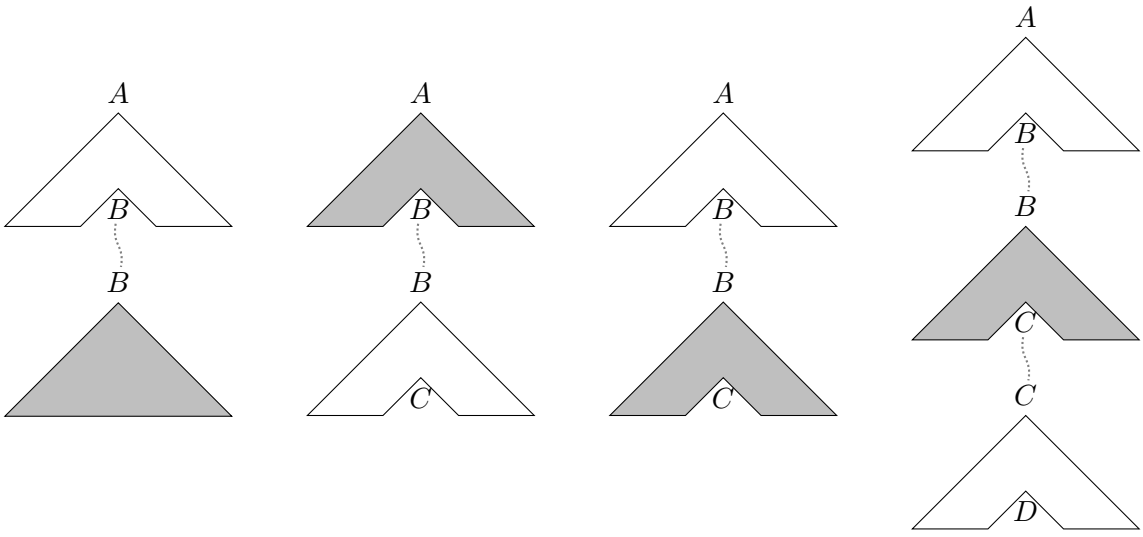


Figure 4: Illustration of connecting operations C0, C1, C2 and C3. The newly created item is shaded in each case.

item in Step 4 says, in effect, that finding a v spanning positions 1 to $n_0$, out of which has moved the −wh element already found spanning positions 0 to 1, will allow us to conclude that the root of a complete tree spans positions 0 to $n_0$. The implication established at Step 2 remains in place, unaffected by this step.

Step 6 places 'John' in the subject position: LC(MERGE2) creates an implication from some selector with features =d$\gamma$ (plus movers, if any) to the parent node with features $\gamma$ (plus the same movers, if any). By taking $\gamma$ to be v and taking the relevant movers to be −wh, the right-hand side of this new implication can be unified with the left-hand side of the one established at Step 4; and furthermore, the left-hand side of the new implication can be unified with the right-hand side of the one established at Step 2 (i.e. the parent of the wh-phrase). The new implication is therefore chained together with two existing ones, by C3, to produce an implication simply from =d =d v to ROOT. The left-hand side of this implication is plugged in when we shift the next word, 'buys'.

Particularly important for the goals outlined above is that instead of the C3(LC(MERGE2)) transition in Step 6, the parser also has the option of taking the C2(LC(MERGE2)) transition shown in Step 6' in Figure 6. This transition involves the same MERGE step putting 'John' in the subject position, and connects the resulting structure "upwards" to the sought-after v, −wh in the same way, but *does not* connect the bottom of the resulting structure to the surroundings of the wh-phrase that were constructed at Step 2. Instead, the sister node of the subject (features =d v, −wh) is left open as the left-hand side of the implication to ROOT, and the implication constructed at Step 2 remains. This is exactly what is required in the sentence being parsed in Figure 6, where the gap is further embedded inside the direct object. But the first five steps are the same in both cases.[8]

The choice between whether to take Step 6 or 6' therefore reflects exactly the choice between whether to follow the active filler strategy or not, just as the the choice between a shift step and a reduce step in Figure 1 reflects the choice between whether to follow the Late Closure strategy or not; recall the discussion of (4) and (5) above. The observed human preference for active gap-finding might therefore be formulated as a preference for C3 transitions over C2 transitions, just as Late Closure effects can be formulated as the result of a preference for shift transitions over reduce transitions. On this view, the filled-gap effect in (2) (i.e. the disruption at 'books') is the result of backtracking out of an area of the search space that a C3 transition led into, corresponding to the analysis in (5a), back to a branching point from which

[8]The same can be said of the Stanojević and Stabler (2018) parser. But that parser would establish the connection between the 'know' clause and the 'eat' clause only after reaching the gap site in 'John knows what Mary ate', in contrast to the way the two clauses would be connected immediately upon entering the 'eat' clause in 'John knows that Mary ate'.

| | | |
|---|---|---|
| 0 init | $((0, n_0) \cdot_1 \texttt{c}) \Rightarrow ((0, n_0)$ ROOT$)$ | |
| 1 SHIFT 'what' | $((0, 1) :: \texttt{d -wh})$ | |
| | $((0, n_0) \cdot_1 \texttt{c}) \Rightarrow ((0, n_0)$ ROOT$)$ | |
| 2 LC(MERGE3) | $((n_0, n_1) \cdot_2 =\texttt{d}\alpha_3\alpha_4) \Rightarrow ((n_0, n_1) : \alpha_3\alpha_4), ((0, 1) : \texttt{-wh})$ | $\alpha_3 \neq +f_9$ |
| | $((0, n_7) \cdot_8 \texttt{c}) \Rightarrow ((0, n_7)$ ROOT$)$ | |
| 3 SHIFT $\epsilon$ | $((1, 1) :: \texttt{=v +wh c})$ | |
| | $((n_4, n_5) \cdot_6 =\texttt{d}\alpha_7\alpha_8) \Rightarrow ((n_4, n_5) : \alpha_7\alpha_8), ((0, 1) : \texttt{-wh})$ | $\alpha_7 \neq +f_{13}$ |
| | $((0, n_{11}) \cdot_{12} \texttt{c}) \Rightarrow ((0, n_{11})$ ROOT$)$ | |
| 4 C2(MV1(LC(MERGE1))) | $((1, n_0) \cdot_1 \texttt{v}), ((0, 1), \texttt{-wh}) \Rightarrow ((0, n_0)$ ROOT$)$ | |
| | $((n_3, n_4) \cdot_5 =\texttt{d}\alpha_6\alpha_7) \Rightarrow ((n_3, n_4) : \alpha_6\alpha_7), ((0, 1) : \texttt{-wh})$ | $\alpha_6 \neq +f_8$ |
| 5 SHIFT 'John' | $((1, 2) :: \texttt{d})$ | |
| | $((1, n_0) \cdot_1 \texttt{v}), ((0, 1), \texttt{-wh}) \Rightarrow ((0, n_0)$ ROOT$)$ | |
| | $((n_3, n_4) \cdot_5 =\texttt{d}\alpha_6\alpha_7) \Rightarrow ((n_3, n_4) : \alpha_6\alpha_7), ((0, 1) : \texttt{-wh})$ | $\alpha_6 \neq +f_8$ |
| 6 C3(LC(MERGE2)) | $((2, n_0) \cdot_1 \texttt{=d =d v}) \Rightarrow ((0, n_0)$ ROOT$)$ | |
| 7 C0(SHIFT) 'buys' | $((0, 3)$ ROOT$)$ | |

Figure 5: Trace of the parser's progress on 'What John buys', with a gap in object position. Variables are subscripted, and unification of variables when the rules apply is restricted by the indicated inequalities. Note that the (derived, lexical) type indicators are variables when they are introduced before the type is specified.

| | | |
|---|---|---|
| 6' C2(LC(MERGE2)) | $((2, n_0) : \texttt{=d v}, ((0, 1), \texttt{-wh}) \Rightarrow ((0, n_0)$ ROOT$)$ | |
| | $((n_2, n_3) \cdot_4 =\texttt{d}\alpha_5\alpha_6) \Rightarrow ((n_2, n_3) : \alpha_5\alpha_6), ((0, 1) : \texttt{-wh})$ | $\alpha_6 \neq +f_8$ |
| 7' SHIFT 'buys' | $((2, 3) :: \texttt{=d =d v}$ | |
| | $((2, n_4) : \texttt{=d v}, ((0, 1), \texttt{-wh}) \Rightarrow ((0, n_4)$ ROOT$)$ | |
| | $((n_6, n_7) \cdot_8 =\texttt{d}\alpha_9\alpha_{10}) \Rightarrow ((n_6, n_7) : \alpha_9\alpha_{10}), ((0, 1) : \texttt{-wh})$ | $\alpha_{10} \neq +f_{11}$ |
| 8' C2(LC(MERGE1)) | $((3, n_0) \cdot_1 \texttt{d}, ((0, 1), \texttt{-wh}) \Rightarrow ((0, n_0)$ ROOT$)$ | |
| | $((n_3, n_4) \cdot_5 =\texttt{d}\alpha_6\alpha_7) \Rightarrow ((n_3, n_4) : \alpha_6\alpha_7), ((0, 1) : \texttt{-wh})$ | $\alpha_7 \neq +f_8$ |
| 9' SHIFT 'books' | $((3, 4) :: \texttt{=p d}$ | |
| | $((3, n_2) \cdot_3 \texttt{d}, ((0, 1), \texttt{-wh}) \Rightarrow ((0, n_2)$ ROOT$)$ | |
| | $((n_5, n_6) \cdot_7 =\texttt{d}\alpha_8\alpha_9) \Rightarrow ((n_3, n_4) : \alpha_6\alpha_7), ((0, 1) : \texttt{-wh})$ | $\alpha_9 \neq +f_9$ |
| 10' C3(LC(MERGE1)) | $((4, n_0) \cdot_1 \texttt{=d p}\alpha_8\alpha_9) \Rightarrow ((0, n_0)$ ROOT$)$ | |
| 11' C0(SHIFT) 'about' | $((0, 5)$ ROOT$)$ | |

Figure 6: Trace of the parser's progress on 'What John buys books about', with gap inside a PP inside an object. As anticipated in the discussion of example (5) above, the first five steps, up to and including the shift step that consumes the subject 'John', are the same as in Figure 5, and so we do not repeat them again, showing only how the remaining steps differ.

we can take a C2 transition instead to construct the analysis in (5b).

This general, formal hypothesis of a preference for C3 transitions over C2 transitions has the potential to make predictions about human parsing preferences in domains beyond those that directly prompted the active gap-finding generalization.

## 6 Conclusion

The main contribution we would like to highlight is that this parser's search space, for sentences containing a filler-gap dependency, is shaped in such a way that it contains branching points corresponding to the choice of whether to (a) posit a gap actively as a first-resort when the opportunity arises, or (b) explore other analyses of the local material first before resorting to positing a gap. This makes it possible to at least *state*, in a precise and general way, the widely-accepted generalization that the human parsing mechanism takes the former option (i.e. adopts the active-filler strategy), and formulate a theory that includes a stipulation to this effect. But if the facts had turned out differently it would have been just as easy to stipulate that the other option is taken instead, and so in this respect we make no claim here to having progressed towards an *explanation* of the observed active-filler generalization. Rather we hope to have pinpointed more precisely what there is to be explained.

This kind of formal instantiation of the active filler idea may also provide a way for variations on the broadly-accepted core idea to be formulated in ways that make precise, distinguishable predictions. For example, looking more closely at Figures 5 and 6, we see that the parser actually posits the gap site before consuming the word that precedes the gap: this happens in Step 6 before shifting 'buys' in Figure 5, and in Step 10' before shifting 'about' in Figure 6. This emerges as a consequence of the fact that, given a binary-branching tree node, a left-corner parser uses one daughter to predict the other (its sister) rather than constructing both independently (as a bottom-up parser would). Since the parser already "knows about" the gap, the way it goes about establishing a structure where the gap and a verb are sisters (if this is what it chooses to do) is by using the gap to predict the verb in its sister position — even though the verb might be usually thought of as appearing to the left of the gap. Although this perhaps diverges from the most natural understanding of the strategies discussed in the psycholinguistics literature, it appears to be similar to the "hyper-active" gap-finding strategy that Omaki et al. (2015) report some evidence for.

A second way in which the details of Figures 5 and 6 may differ from the usual conception of the active filler strategy is that the filler wh-phrase is integrated into its surface position *after* the complementizer is shifted in Step 3. In a sense it is the `+wh` feature on the complementizer that really triggers the construction of the MOVE step of the derivation, rather than the filler, and the filler is identified as the moved `-wh` element only indirectly by virtue of the fact that it covers the required span, from position 0 to position 1. In these sentences with a null complementizer this difference is not really meaningful, but it may be in languages that allow an overt complementizer to co-occur with a fronted wh-phrase.

Finally, one of the most well-known properties of active gap-finding is that it is island-sensitive: humans do not posit gaps in positions which are separated from the filler position by an island boundary (e.g. Traxler and Pickering, 1996; Wagers and Phillips, 2009). In future work we intend to investigate whether this effect might fall out as a natural consequence of certain grammatical encodings of the relevant island constraints.

## Acknowledgments

The second author was supported by ERC H2020 Advanced Fellowship GA 742137 SEMANTAX grant. The second and third authors devised and implemented the parser described here; the first author contributed the psycholinguistic motivations and interpretations.

## References

Emmon Bach. 1979. Control in Montague Grammar. *Linguistic Inquiry*, 10(4):515–531.

J.R. Brennan, E.P. Stabler, S.E. VanWagenen, W.-M. Luh, and J.T. Hale. 2016. Abstract linguistic structure correlates with temporal activity during naturalistic comprehension. *Brain and Language*, 157-158:81–94.

Noam Chomsky. 1995. *The Minimalist Program*. MIT Press, Cambridge, Massachusetts.

Janet Dean Fodor. 1978. Parsing strategies and constraints on transformations. *Linguistic Inquiry*, 9(3):427–473.

Lyn Frazier and Charles Clifton. 1989. Successive cyclicity in the grammar and the parser. *Languages and Cognitive Processes*, 2(4):93–126.

Lyn Frazier and Charles Clifton. 1996. *Construal*. MIT Press, Cambridge, MA.

Thomas Graf. 2013. *Local and transderivational constraints in syntax and semantics*. Ph.D. thesis, UCLA.

Thomas Graf and Bradley Marcinek. 2014. Evaluating evaluation metrics for minimalist parsing. In *Procs. 2014 ACL Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*, page 2836.

John T. Hale. 2003. *Grammar, uncertainty and sentence processing*. Ph.D. thesis, Johns Hopkins University.

John T. Hale. 2006. Uncertainty about the rest of the sentence. *Cognitive Science*, 30:643–672.

Tim Hunter. 2011. *Syntactic Effects of Conjunctivist Semantics: Unifying Movement and Adjunction*. John Benjamins, Philadelphia.

Tim Hunter. in press. Left-corner parsing of minimalist grammars. In R.C. Berwick and E.P. Stabler, editors, *Minimalist Parsing*. Oxford University Press.

Gregory M. Kobele. 2010. Without remnant movement, MGs are context-free. In *Mathematics of Language 10/11*, LNCS 6149, pages 160–173, NY. Springer.

Gregory M. Kobele, Sabrina Gerth, and John T. Hale. 2012. Memory resource allocation in top-down minimalist parsing. In *Procs. Formal Grammar 2012*, Opole, Poland.

Gregory M. Kobele and Jens Michaelis. 2011. Disentangling notions of specifier impenetrability. In M. Kanazawa, A. Kornai, M. Kracht, and H. Seki, editors, *The Mathematics of Language*, pages 126–142. Springer, Berlin.

Akira Omaki, Ellen F. Lau, Imogen Davidson White, Myles L. Dakan, Aaron Apple, and Colin Phillips. 2015. Hyper-active gap filling. *Frontiers in Psychology*, 6(384).

Colin Phillips. 2006. The real-time status of island phenomena. *Language*, 82:795–823.

Philip Resnik. 1992. Left-corner parsing and psychological plausibility. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING '92)*, pages 191–197.

Edward P. Stabler. 1997. Derivational minimalism. In C. Retoré, editor, *Logical Aspects of Computational Linguistics*, LNCS 1328, pages 68–95. Springer-Verlag, NY.

Edward P. Stabler. 2011. Computational perspectives on minimalism. In Cedric Boeckx, editor, *Oxford Handbook of Linguistic Minimalism*, pages 617–641. Oxford University Press, Oxford.

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

Miloš Stanojević and Edward Stabler. 2018. A sound and complete left-corner parser for Minimalist Grammars. In *Procs. Eighth Workshop on Cognitive Aspects of Computational Language Learning and Processing*, pages 65–74.

Adrian Staub. 2007. The parser doesn't ignore transitivity, after all. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 33(3):550–569.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Laurie A. Stowe. 1986. Parsing wh-constructions: Evidence for on-line gap location. *Language and Cognitive Processes*, 1(3):227–245.

Matthew J. Traxler and Martin J. Pickering. 1996. Plausibility and the processing of unbounded dependencies: An eye-tracking study. *Journal of Memory and Language*, 35:454–475.

Matthew W. Wagers and Colin Phillips. 2009. Multiple dependencies and the role of grammar in real-time comprehension. *Journal of Linguistics*, 45:395–433.

Jiwon Yun, Zhong Chen, Tim Hunter, John Whitman, and John Hale. 2015. Uncertainty in the processing of relative clauses in East Asian languages. *Journal of East Asian Linguistics*, 24(2).