

# A workspace-based theory of adjuncts

Daniel Milway (dan.milway@gmail.com)

## 1 The problem of adjuncts in Generative Grammar

The Generative Grammar enterprise has been remarkably successful at reducing a wide variety of grammatical relations to a handful of simple ones. The adjunct relation, though, is among a few which have stubbornly refused reduction.<sup>1</sup> So, while relations such as *subject-of*, *object-of*, and *possessum-of* can now be defined in terms of Merge, the adjunct relation cannot. Instead their exceptional nature is stipulated, or an attempt is made to show that it is illusory. The former approach, of course, is undesirable, but the latter approach, if successful, would yield a very desirable result.

The latter approach, I will argue in section 2, does not seem likely to succeed, because the evidence for the exceptionality of adjuncts is fairly strong. I will further argue that the nature of this exceptionality—that is, the very nature of adjuncts—makes reduction to Merge logically impossible.

I will then argue, in section 3 that attempts to account for adjuncts by either redefining Merge or adding a novel operation, while logically feasible, are undesirable when we consider the broader context of linguistic theory. Specifically I argue that the optionality of adjuncts negates any claims of necessity for introducing theoretical complication to account for them. This conclusion, however, seems to contradict the conclusion of section 2, creating an apparent paradox which I address in section 4.

---

<sup>1</sup>Also in this group is conjunction.

20 In the remainder of the paper, I present a theory of adjuncts according to which, adjuncts, like argu-  
21 ments, are derived in separate workspaces from their host but, unlike argument, adjunct workspaces  
22 are never incorporated into their host’s workspace. I introduce the theoretical background for this  
23 proposal in section 5, make my proposal in section 6, and offer corroborating evidence in section 7.

## 24 **2 The essence of adjuncts**

25 Adjuncts are generally distinguishable from predicates and arguments on the basis of three  
26 properties—optionality, stackability, and reorderability.<sup>2</sup> They are optional in the sense that (1)—  
27 without any adjuncts—and (2)—with an adjunct—are both grammatical. They are stackable in  
28 the sense that an expression like (2), with one adjunct, is as grammatical as one like (3) with a  
29 second adjunct added, and so on. Finally, they are reorderable in the sense that (3) and (4) are both  
30 grammatical despite the fact that their adjuncts are ordered differently.

31 (1) [X The police silenced the workers].

32 (2) [Y [X The police silenced the workers], [against their wills]].

33 (3) [Z [Y [X The police silenced the workers], [against their wills]], [following the demonstration]].

34 (4) [U [W [X The police silenced the workers] [following the demonstration]], [against their  
35 wills]].

36 A theoretical approach like cartography (Cinque and Rizzi 2010), though, argues that adjuncts do  
37 not form a class of their own. Rather, “adjuncts” are actually specifiers of particular functional  
38 heads. The argument for this claim begins with the observation that “adjuncts” are not as reorderable  
39 as (3) and (4) would suggest. For instance, there are well-known restrictions on the ordering of  
40 adjectives—an ordering of size adjectives before shape adjectives, as in (5), is preferred to the  
41 reverse order, as in (6).<sup>3</sup>

---

<sup>2</sup>These properties are generalizations to which, no doubt, there are exceptions. I ask the reader to attach “all else being equal” to such generalizations.

<sup>3</sup>See Sproat and Shih (1991) for further discussion of the adjective ordering restriction

42 (5) a small square table

43 (6) ?\* a square small table

44 By hypothesis, such an ordering restriction reflects a fixed ordering of a set of functional heads  
45 (SIZE>SHAPE) which select adjectives as specifiers. Since this set of functional heads is innate, it  
46 must be finite. It follows, —then, that adjectives—and, by extension, adjuncts in general—cannot be  
47 stacked indefinitely. That is, there is a fixed upper bound of the number of adjuncts in an expression.  
48 As for optionality, one need only point out the plethora of optionally transitive verbs, or pro-drop  
49 languages to see that optionality is not the sole province of adjuncts. Thus a cartographic approach  
50 comes to the conclusion that there is no special class of constituents that answers to the name  
51 “adjuncts”.

52 The cartographic conclusion, however, has certain implications which, when made explicit, cast  
53 doubt on it. The first implication is that adjective ordering restrictions such as the one demonstrated  
54 in (5) and (6) represents only an example case of a larger ordering restriction that includes not  
55 only all other adjective classes but also determiners and nouns. That is, the ordered sequence of  
56 functional heads SIZE > SHAPE is part of a larger sequence given in (7).

57 (7) D > ... > SIZE > ... SHAPE > ... N

58 However, when we test this we see that placing a determiner or noun out of order yields a different  
59 sort of unacceptability compared to that in (6)

60 (8) a. \*square a small table

61 b. \*square small a table

62 c. \*square small table a

63 (9) a. \*a square table small

64 b. \*a table square small

65 c. \*table a square small

66 While (6) is an awkwardly formed nominal phrase, the strings in (8) and (9) are gibberish.

67 Or consider another ordering restriction in English—S>V>O. Disobeying this ordering restriction  
68 usually yields ungrammatical strings (\*VSO, \*VOS), but sometimes yields a distinct grammatical  
69 sentence (OVS). The latter result does not seem to occur when adjunct ordering restrictions are  
70 violated.

71 The task of assimilating adjuncts to specifiers or complements seems difficult at least. As soon as  
72 they are brought under the same umbrella as other types of constituents, they must immediately be  
73 relegated to a special corner of that umbrella. Absent any stronger arguments, I will continue to  
74 assume that adjuncts exist and have the properties listed above.

75 It stands to reason, though, that the three characteristic properties of adjuncts—optionality, stack-  
76 ability, and reorderability—should be reducible to a single essential characteristic, and that the  
77 discovery of that characteristic is the first step towards an explanatory account of adjuncts. The key  
78 to discovering this characteristic, I think, is the fact that the labelled expressions in (1)-(4)—U, W,  
79 X, Y and Z—are all syntactically equivalent, a term that bears explanation.

80 The expressions in question are all equally grammatical, but this is not enough to call them  
81 syntactically equivalent—(10) and (11) are equally grammatical but not equivalent, as they are  
82 distinct categories.

83 (10) The cat ate the fish.

84 (11) The morning star

85 The expressions in question are of the same category, but this is also not enough to call them  
86 syntactically equivalent. According to most standard theories of labelling, the expressions in (12)  
87 are both labelled as V and the expressions in (13) are both labelled as T, but the (a) examples are  
88 not equivalent to the (b) examples.

89 (12) a. hit

90 b. hit the ball

91 (13) a. ate the cake

92           b. Juan ate the cake

93 We can show that the (a) examples are not equivalent to the (b) examples by attempting to substitute  
94 one for the other in a test context. So in (14), the ungrammatical (b) is derived from (a) by  
95 substituting *hit* for *hit the ball*, and in (15), the ungrammatical (b) is derived from (a) by substituting  
96 an incomplete TP for a complete one.

97 (14)   a. The toddler hit the ball.

98           b. \*The toddler hit

99 (15)   a. I believe Juan ate the cake.

100          b. \*I believe ate the cake.

101 Compare this to a case of adjunction—in (16), (b) is derived from (a) by substituting a VP with  
102 an adjunct for one without an adjunct. Both examples are grammatical, because the VP with an  
103 adjunct is equivalent to the VP without the adjunct.

104 (16)   a. Rosie [sang the anthem].

105          b. Rosie [sang the anthem with gusto].

106 The substitution test does not furnish us with a definition of syntactic equivalency, though. Such a  
107 definition requires additional work. The fact of syntactic equivalency, though, leads to an important  
108 conclusion about host-adjunct expressions—they are not formed by Merge.

109 The reasoning to this conclusion is straightforward. By definition, Merge combines two objects  $\alpha$   
110 and  $\beta$  to create a new object  $\gamma$  that is distinct from both  $\alpha$  and  $\beta$ . So, if a host-adjunct expressions  
111  $H \frown A$  were formed by Merging H and A, it would be distinct. All host-adjunct expressions  $H \frown A$   
112 are equivalent to their host H and therefore cannot be formed by Merge.

113 Several researchers have previously reached this conclusion in some form or another. This leads  
114 them to propose an additional way of combining expressions. In the next section, I show that this  
115 step is neither necessary, nor desirable.

### 116 **3 Adjunction is optional**

117 Our current approaches to adjunction—late Merge and Pair-Merge—introduce complications to  
118 the theory of grammar, and any complication should be viewed with some skepticism and only  
119 accepted if they are shown to be absolutely necessary. Proponents of the current approaches, then,  
120 plead necessity. The argument goes as follows: Adjunction is ubiquitous in language but cannot be  
121 reduced to cyclic application of simple Merge, therefore our theory of language cannot be limited to  
122 cyclic application of simple Merge.

123 This can be seen as analogous to one of Chomsky’s argument for transformations in *Syntactic*  
124 *Structures*, which goes roughly as follows: phrase structure rules alone are not sufficient to generate  
125 all the sentences of natural language, therefore we must augment them with transformations.  
126 Although there were other arguments for transformations, I would like to compare their necessity to  
127 the supposed necessity of Pair-Merge or late Merge.

128 To start with, we must come to some description of a valid necessity argument in generative syntax.  
129 A necessity argument takes the form of a syllogism consisting of two premises and a conclusion.  
130 The first premise is an empirical claim about the expressiveness of natural language, where what I  
131 mean by expressiveness is the range of thoughts that are expressible in natural language. the claim  
132 made in the first premise is something like “the expressiveness of natural language includes at least  
133 the thought-classes *A, B, C, D*,” where *A, B, C* and *D* are demonstrated with some data. The second  
134 premise is a claim about the expressiveness of some theoretical grammar—something like, “the  
135 proposed grammar *G* of natural language generates expressions of thought-classes *A, B* and *C*, but  
136 not *D*.” The conclusion, of course, is that *G* is an insufficient theory of natural language.

137 A simplified version of the necessity argument for transformations is as follows. Natural languages  
138 can express both declarative and interrogative forms of the same core proposition as shown in (17)  
139 and (18).

140 (17) Violet wrote the anthem.

141 (18) What did Violet write?

142 Phrase structure rules, however, can generate declaratives, but not interrogatives. Therefore, phrase  
143 structure rules are insufficient as a theory of natural language. Important to note is that both premises  
144 rest on other premises. The first premise assumes that declaratives and interrogatives express distinct  
145 classes of thoughts, while the second premise assumes a particular version of phrase structure rules.  
146 The necessity argument for Pair-Merge or late Merge goes as follows. Natural language can express  
147 thought with or without adjuncts as shown in (19) and (20)

148 (19) Rosie sang the anthem.

149 (20) Rosie sang the anthem with gusto.

150 A Merge-only grammar can generate unadjoined expressions like (19) but not adjoined sentence  
151 like (20). Therefore, a Merge-only grammar is insufficient as a theory of natural language. Since I  
152 have already argued in favour of the the second premise, I will examine the first premise here.

153 The first premise assumes that (19) and (20) express two distinct classes of thought. When we  
154 analyze (20) though, we see that it can actually be expressed as the juxtaposition of two distinct  
155 sentences: (19) and (21).

156 (21) The singing had gusto.

157 If a Merge-only grammar can generate (19), it should also be able to generate (21), and therefore, it  
158 can generate the juxtaposition of the two. Given my definition of expressiveness in terms of the  
159 range of thoughts expressible by a language, then, it seems that the thought expressed by (20) can  
160 be expressed in a Merge-only grammar. So, the necessity argument for Pair-Merge or late Merge  
161 does not go through.

## 162 **4 A paradox?**

163 In section 2 I argued that Merge could not create host-adjunct structures, while in section 3 I argued  
164 that a Merge-only grammar was expressive enough to generate the thoughts behind expressions with  
165 adjuncts. On their face, these seem to contradict each other. This apparent contradiction, though,  
166 can be cleared up by being a bit more precise about what is being claimed in each case.

167 The first claim is a conditional claim. It says that if there exists a computational combinatory  
168 operation—call it Adjoin—that creates host adjunct structures, then Adjoin cannot be identical to  
169 to Merge. The second claim is a modal claim. It says that the human language faculty does not  
170 necessarily contain the computational combinatory operation Adjoin. Stated this way, then, the two  
171 claims are compatible with each other, though they leave in a difficult position in our search for a  
172 theory of adjuncts.

## 173 **5 Towards a theory of Adjuncts**

174 Our theory of adjuncts, then, must account for the fact that adjunction is syntactically vacuous and  
175 it should do so without adding any additional combinatory operation.

### 176 **5.1 Workspaces**

177 In recent years, two distinct conceptions of workspaces have gained currency among generative  
178 theorists. In one conception, formalized by Collins and Stabler (2016), each stage of a derivation  
179 consists of a lexical array, containing lexical item tokens, and a workspace, containing syntactic  
180 objects. In the other conception, described by Nunes (2004), a stage of a derivation consists of  
181 possibly several workspaces. I will be adopting the latter conception.

182 Under this conception, a workspace is a way of formalizing the intuition that arguments are derived  
183 separately from clausal spines. So for instance, the derivation of a simple transitive clause like (22)  
184 involves at least three workspaces—One derivation each for the nominal arguments, *the citizens* and



185 *the masks*, and one derivation for the clausal spine.

186 (22) The citizens wore some masks.

187 Operations like Merge are defined in terms defined in terms of workspaces, which has the effect  
188 of encapsulating the workspaces. In the case of Merge, this means that the workspace defines the  
189 operation's accessibility conditions. Two objects, then, can be Merged only if they are in the same  
190 workspace. So, for instance, in the derivation of (22), the indefinite determiner *some* cannot Merge  
191 with *citizens*, as the two objects are not in the same workspace.

## 192 5.2 Deletion

193 Deletion is a much more intuitive, yet less understood, operation of the language faculty. Every  
194 generative theory of syntax has some mechanism to either not pronounce certain constituents of  
195 a given expression or imbue silence with meaning. In current transformational theories, this is  
196 accomplished by the sensorimotor system rather than the narrow syntax. In these theories, syntactic  
197 derivations tend to create redundant structure. For example the derivation of a *wh*-question like (23)  
198 involves merging the *wh*-expression twice, first as the direct object and then at the left edge of the  
199 sentence. This results in a syntactic object with two instances of that *wh*-expression as in (24).

200 (23) What did the student hear?

201 (24) What did the student hear what

202 Since (23), rather than (24), is pronounced, we know that the rightmost copy of *what* has been  
203 deleted.

204 Our current theory of deletion starts with a single principle given in (25).

205 (25) Given two identical objects X and Y where X asymmetrically c-commands Y, delete Y.

206 This principle accurately predicts (23), but faces several issues. Empirically speaking, there are many  
207 apparent exceptions to (25). Theoretically speaking, the notion of identity is not fully understood

208 and, as I will discuss in section 6.1.2, c-command as the deciding factor is too narrow.

209 Despite these issues, I will assume something like (25), perhaps with exceptions, is active in the  
210 language faculty.

## 211 **6 A workspace-based theory of adjunction**

212 The theory of adjuncts that I propose is best viewed in contrast to the workspace theory of arguments.  
213 According to this theory, outlined in section 5.1, an argument is derived in a separate workspace  
214 from its clausal spine, and the result of that derivation is merged into clausal spine derivation. An  
215 adjunct is also derived in a separate workspace, except that that workspace is never merged into the  
216 clausal spine derivation. So the syntactic representation of (20) is given in (26) with the adjunct-free  
217 sentence derived (19) in WS1, and the adjunct PP *with gusto* derived in WS2.

218 (26)  $\langle [\{ \{ Rosie, \{ T, \dots \{ sing, \{ the, song \} \} \} \} ]_{WS1}, [\{ with, gusto \} ]_{WS2} \rangle$

219 Note this perfectly captures the essential character of adjuncts, namely that they are syntactically  
220 vacuous. The VPs *sing the anthem* and *sing the anthem with gusto* are syntactically equivalent  
221 because there is no narrow syntactic object that corresponds to the latter string. So, it is wrong to  
222 say that a Voice head selects both objects. Rather the relevant part of the derivation of *sing the*  
223 *anthem* proceeds as in (27) while the corresponding derivation part for *sing the anthem with gusto*  
224 proceeds as in (28).

225 (27) Stage N:  $\langle [\{ sing, \{ the, anthem \} \}, Voice ]_{WS1} \rangle$  (Merge(Voice, WS1))

226 Stage N+1:  $\langle [\{ Voice, \{ sing, \{ the, anthem \} \} ]_{WS1} \rangle$

227 (28) Stage N:  $\langle [\{ sing, \{ the, anthem \} \}, Voice ]_{WS1}, [\{ with, gusto \} ]_{WS2} \rangle$  (Merge(Voice, WS1))

228 Stage N+1:  $\langle [\{ Voice, \{ sing, \{ the, anthem \} \} ]_{WS1}, [\{ with, gusto \} ]_{WS2} \rangle$

229 Notice that each stage pair is derived by the same operation.

230 In terms of interpretation, this proposal makes roughly the correct prediction/ That is, the host

231 and adjunct are distinct syntactic objects and, therefore, would be interpreted as such. Recall,  
232 in section 3, I argued that an expression with an adjunct could be expressed as two juxtaposed  
233 expressions. This analysis formalizes that intuition.

234 Turning to pronunciation, it might be suggested that my proposal introduces new complexity  
235 to the already complicated nature of pronunciation—Our best theories suggest that c-command  
236 is vital for linearization, but there can be no c-command relation across workspaces. Such an  
237 objection, however, would mistake the nature of the linearization problem, namely that Merge  
238 creates unordered objects that must be converted to ordered object for pronunciation. A derivation  
239 stage such as (26), though, is already ordered ( $WS1 \prec WS1$ ), so no linearization problem should  
240 occur.

241 In what follows, I will refine this proposal somewhat, but the core claim—that adjuncts are in  
242 separate workspaces from their hosts—will remain the same. I pause here to note that this solution  
243 broadly accounts for adjunct without recourse to novel operations or major modifications to the  
244 architecture of the grammar, and is therefore superior to Pair-Merge and late Merge.

## 245 **6.1 The problem of adjunct scope**

246 The sentence in (29) is ambiguous.

247 (29) Sharon made the error deliberately.

248 It can be interpreted as saying either that Sharon intended to make the error in question, or that she  
249 made the error in a deliberate manner. The conclusion drawn from this sort of ambiguity is that the  
250 adverb *deliberately* has two possible scopes—A high scope resulting in the first interpretation, and  
251 a low scope resulting in the second interpretation. Under an X-bar theory of adjuncts, this can be  
252 easily accounted for by aligning scope with attachment site as in (30) and (31).

253 (30) **The high-scope interpretation of (29) in X-bar theory**



269 described by saying which of the two quantifier phrases scopes over the other.

270 (33) Every student read a book.

271 a.  $\forall s(\exists b(\text{read}(b,s)))$

272 b.  $\exists b(\forall s(\text{read}(b,s)))$

273 The relationship between a modifier and a modified expression, however, is generally considered  
274 to be symmetric, at least in terms of their interpretation.<sup>4</sup> So, in the low-scope interpretation of  
275 (29), the logical predicate expressed by *deliberately* is conjoined with the one expressed by *make an*  
276 *error*, as shown in (34).

277 (34)  $\lambda e(\text{make}(\text{the-error}, e) \& \text{deliberate}(e))$

278 It does not, then, make sense to say that *deliberately* “scopes over” the VP. We can still ask, though,  
279 why does *deliberately* conjoin with the VP and not, say, with AspP, or TP. The answer, at least in  
280 X-bar terms is obvious—the adverb and the VP conjoin because they are in the same position, that  
281 is [Comp, Voice]. In other words, *deliberately* conjoins with the VP, because both scope directly  
282 under Voice, and therefore, indirectly under everything that scopes over Voice.

283 This rethinking of adjunct scope, then suggests a workspace-based analysis of the low scope  
284 interpretation of (29), shown in (35).

285 (35)  $\left\langle \begin{array}{l} [\{\text{Sharon}, \{\text{T}, \dots \{\text{Voice}, \{\text{make}, \{\text{the}, \text{error}\}\}\}\}], \\ [\{\text{Sharon}, \{\text{T}, \dots \{\text{Voice}, \{\text{deliberately}\}\}\}], \end{array} \right\rangle$

286 Here we can say that *deliberately* and the VP are in the same position, as they are both the  
287 complement of Voice in their respective workspaces. Such a representation, however, raises three  
288 obvious questions:

- 289 1. How is (35) interpreted?
- 290 2. How is (35) pronounced?
- 291 3. How is (35) derived?

---

<sup>4</sup>Setting aside cases of non-intersective modification.

292 I address these three questions in turn directly.

### 293 **6.1.1 How is (35) interpreted?**

294 The derivation stage in (35) contains two workspaces, each of which contains a finite clause. I will  
295 assume that the interpretation of each clause contains an event description and a specification of  
296 how the event described relates to the context of utterance. For the sake of clarity, I will consider  
297 only the event-description portion of the meaning.

298 So the event description contained in the first workspace—the one associated with the host— is  
299 given in (36), and the event description contained in the second workspace—the one associated  
300 with the adjunct—is given in (37).

301 (36)  $\lambda e(\text{make}(e) \& \text{AGENT}(e)(\mathbf{sharon}) \& \text{THEME}(e)(\mathbf{the-error}))$

302 (37)  $\lambda e(\text{AGENT}(e)(\mathbf{sharon}) \& \text{deliberately}(e))$

303 If, as we've assumed thus far, juxtaposing (36) and (37) yields the conjunction of the two, and if we  
304 take the further simplifying step of eliminating redundant conjuncts, we get the correct interpretation  
305 in (38).

306 (38)  $\lambda e(\text{make}(e) \& \text{AGENT}(e)(\mathbf{sharon}) \& \text{THEME}(e)(\mathbf{the-error}) \& \text{deliberately}(e))$

307 More could be said, of course, about the interpretation of (35), but I will leave this as a task for  
308 further research and move on to the question of pronunciation

### 309 **6.1.2 How is (35) pronounced?**

310 The problem posed for pronunciation by (35) is that the adjunct workspace contains most of a clause  
311 which is not pronounced. That is, *Sharon*, T, Voice, *etc.* must be deleted somehow. Recall from  
312 section 5.2 that the basic rule of deletion is that if a syntactic object contains two constituents,  $\alpha$   
313 and  $\beta$ , such that  $\alpha = \beta$  and  $\alpha$  asymmetrically c-commands  $\beta$ , then  $\beta$  is deleted.

314 The notion of identity here, must capture copies, but not repetitions, so in order for the various

315 phrases and heads to be deleted from the adjunct we must show that they can be treated as copies of  
316 the corresponding phrases and heads in the host. since the distinction between copies and repetitions  
317 is to follow from the derivational history of an expression, I will postpone the question of identity  
318 until the following section and stipulate, for the moment, that *Sharon*, T, Voice, *etc.* in the adjunct  
319 are considered copies of their counterparts in the host.

320 As for the c-command requirement for deletion, it is quite plain that it cannot apply to the deletion  
321 of copies in different workspaces as in (35). Since the c-command relation is dependant on Merge,  
322 the domain of which is limited to the workspace, it cannot hold across workspaces. However, if we  
323 broaden the c-command requirement on deletion to one of a more general ordering ( $\alpha > \beta$ ) then it  
324 can apply to elements in separate workspaces, since workspaces in a derivation are ordered with  
325 respect to each other.

326 This broadening of the c-command requirement may seem *ad hoc* on its face, but there is a good  
327 reason to think that an operation like deletion is not sensitive specifically to c-command. That  
328 reason is that, as decades of research suggest, the syntactic component is the only component of  
329 the language faculty that is particular to the language faculty. It follows from this that deletion,  
330 an operation of the externalization system, is not particular to language. Since it is not particular  
331 to language, it should not be defined in language-particular terms. Therefore, defining deletion in  
332 terms of ordering as opposed to c-command is theoretically preferred.

333 So, turning back to the task at hand, (35) is pronounced by deleting all the redundant structure in  
334 the adjunct. This occurs because every element of the deleted structure is identical to an element in  
335 the host and ordered with respect to that matching element.

### 336 **6.1.3 How is (35) derived?**

337 The derivation of host-adjunct structures such as (35) can be divided into to parts. In the first  
338 part, the two workspaces—host and adjunct— are derived independently of each other, and in the  
339 second part, the workspaces are derived in lockstep. The first part represents the standardly assumed

340 operation of workspaces, and is, therefore, already understood, at least insofar as workspaces are  
341 understood. The second part—the part involving lockstep derivation—is novel and its explanation  
342 will occupy this section.

343 The result of the first part of the derivation is given in (39) below.

$$344 \quad (39) \quad \left\langle \begin{array}{l} [\{make, \{the, error\}\}, Voice, \dots, T]_{WS1}, \\ [\{deliberately\}, Voice, \dots, T]_{WS2}, [Sharon]_{WS3} \end{array} \right\rangle$$

345 Let's suppose that nothing forces the workspaces to derive in lockstep, but rather they derive freely  
346 and only result in a host adjunct structure if their respective derivations mirror each other. This,  
347 however, would lead to two problems.

348 The first problem this poses has to do with the copy/repetition distinction. The externalization  
349 system, by hypothesis, deletes copies, not repetitions. Recall that T, Voice, the subject, *etc.* of the  
350 adjunct workspace delete in this case. This deletion would only occur if those objects were copies of  
351 their counterparts in the host object and, while the necessary and sufficient conditions on copy-hood  
352 are not well understood, There is good reason to believe that content-identity is not sufficient. That  
353 is, Two instances of, say,  $Voice_{Act}$  are not copies just because they have identical content—it seem  
354 they must have an identical derivational history. This could not possibly hold of Voice, T, *etc* if the  
355 second stage of the derivation under discussion proceeds freely.

356 The second problem has to do with the fact that the subject *Sharon* appears in both the host and  
357 adjunct workspace in (35). If we were to derive the two workspaces independently of each other,  
358 starting with (39), there would be a step in which WS3 would be added to either WS1 or WS2.  
359 Incorporating WS3 into one workspace, however, precludes a later step of incorporating it into the  
360 other workspace. Therefore, (35) does not seem to be derivable if its two workspaces are derived  
361 independently of each other.

362 The lockstep derivation, then, must be “forced”, yet current theory offers no method for this. In  
363 the remainder of this section I will present and discuss a proposal which would allow for lockstep  
364 derivation. First, I will introduce and formally define a workspace-based version of Merge—what



365 Chomsky (2019) refers to as MERGE. Then, I will discuss the higher-order function, map, which  
 366 will allow us do derive in lockstep. Finally, I will discuss how the copy/repetition distinction can be  
 367 made in light of these developments.

### 368 6.1.3.1 MERGE and its formal definition

369 Chomsky (2019) argues that the standard conception of Merge— $\text{Merge}(\alpha, \beta) \rightarrow \{\alpha, \beta\}$ —needs  
 370 to be replaced with a new one, called MERGE, which meets a number of desiderata. One such  
 371 desideratum is that MERGE should be defined in terms of workspaces, rather than syntactic objects.  
 372 In order to do this we must first provide some definitions for workspaces and other derivational  
 373 notions. These definitions are given in (40)-(42).

374 (40) A derivation  $D$  is a finite sequence of stages  $\langle S_1, S_2, \dots, S_n \rangle$ , where  $D(i) = S_i$ .

375 (41) A stage  $S$  is a finite sequence of workspaces  $\langle WS_1, WS_2, \dots, WS_n \rangle$ , where  $S(i) = WS_i$ .

376 (42) A workspace  $WS$  is a finite sequence of syntactic objects  $\langle SO_1, SO_2, \dots, SO_n \rangle$ , where  $WS(i) =$   
 377  $SO_i$ .

378 In addition to the workspace desideratum, MERGE should also “restrict computational resources”  
 379 (Chomsky 2019), by ensuring that when a new object is created by MERGE, its constituent parts  
 380 do not remain accessible in the workspace. That is, MERGE substitutes the new object for the  
 381 old objects. The definition of MERGE in (43), where “+” represents an “append” operation and  
 382 “−” represents a “delete” operation, meets the two desiderata that I have mentioned thus far.<sup>5</sup>

383 (43) Where  $\omega$  is a workspace, and  $\alpha$  and  $\beta$  are syntactic objects,

$$384 \quad \text{MERGE}_3(\omega, \alpha, \beta) \rightarrow \begin{cases} \{\alpha, \beta\} + ((\omega - \alpha) - \beta) & \text{if } \alpha \text{ and } \beta \text{ are in } \omega \\ \{\alpha, \beta\} + (\omega - \alpha) & \text{if } \alpha \text{ is in } \omega \text{ and } \beta \text{ is in } \alpha \\ \text{undefined} & \text{otherwise} \end{cases}$$

---

<sup>5</sup>The astute reader will likely note that my definition of MERGE sacrifices the simplicity of Merge to meet the Chomsky’s desiderata. This, I believe, reflects the fact that we lack a sufficient model of neural computation in which to ground our grammatical theory. Such a model would likely meet the “restrict resources” desideratum automatically.

385 This definition, however, seems to over-generate. Consider the derivation in (44)

386 (44)  $WS = \langle P, Q, X, Y \rangle$  ( $P, Q, X,$  and  $Y$  are lexical item tokens)

387 a.  $merge(WS, P, Q) \rightarrow \langle \{P, Q, X, Y\} (= WS') \rangle$

388 b.  $merge(WS', X, Y) \rightarrow \langle \{P, Q\}, \{X, Y\} \rangle (= WS'')$

389 If such a derivation were possible within a single workspace, then we could derive an entire  
 390 clause—including complex nominal arguments—within a single workspace. This would, at best,  
 391 render workspaces redundant, perhaps making the grammar indeterminate—any sentence would be  
 392 derivable in at least two distinct ways.

393 The situation gets worse when we consider the fact that the definition of merge in (43) stipulates the  
 394 distinction between internal and external merge. By hypothesis, though, the two cases of merge  
 395 should fall out from a single definition of merge. Without the stipulation, it's likely that unrestricted  
 396 parallel merge (Citko 2005) or sideward merge (Nunes 2004) would be derivable in this system.  
 397 As Chomsky (2019) argues, though, once such varieties of merge are allowed, there is virtually  
 398 no restriction on what can be derived. Thus, a definition of merge like that in (43) would likely  
 399 over-generate.

400 This issue can be overcome in a non-stipulative way by eliminating one of the syntactic-object  
 401 arguments from the definition of merge and defining merge as in (45).

402 (45) Where  $\omega$  is a workspace, and  $\alpha$  is a syntactic object,

$$403 \quad \text{MERGE}_2(\omega, \alpha) \rightarrow \begin{cases} \{\alpha, \omega(1)\} + ((\omega - \alpha) - \omega(1)) & \text{if } \alpha \text{ is in } \omega \\ \{\alpha, \omega(1)\} + (\omega - \omega(1)) & \text{if } \alpha \text{ is in } \omega(1) \\ \text{undefined} & \text{otherwise} \end{cases}$$

404 I have restricted merge here by identifying a privileged member of a given workspace—the first  
 405 member  $\omega(1)$ . This is what is sometimes referred to as the root of the tree. This is a justifiable step  
 406 in that the first member of a workspace has a unique property among workspace members—the  
 407 existence of a workspace depends only on the existence of its first member. That is, there are

408 workspaces of length 1, 2, 3, *etc* but no workspaces of length 0. A corollary of this is that the  
409 proposition in (46) is only true for  $i = 1$ .

410 (46) For every workspace  $\omega$ ,  $\omega(i)$  is defined.

411 By restricting merge in this way, we can rule out the derivation in (44). All instances of  $\text{MERGE}_2$   
412 modify  $\text{WS}(1)$ .  $\text{WS}''(1)$  and  $\text{WS}'(1)$  in (44) are identical. Therefore No instance of  $\text{MERGE}_2$  could  
413 derive  $\text{WS}''$  from  $\text{WS}'$ .

414 Being a computational procedure,  $\text{MERGE}$  ought to proceed in steps. Therefore, it should be a  
415 curried (or schönfinked) function. So,  $\text{MERGE}$  would be defined as in (47), with  $\mathcal{M}$  standing in  
416 for the intension of  $\text{MERGE}$  (*i.e.*, the right side of the equals sign in (43)).

417 (47)  $\text{MERGE} = (\lambda \omega. (\lambda \alpha. \mathcal{M}))$

418 Curried functions are a variety of higher-order functions because they have functions as outputs  
419 in contrast first-order functions whose inputs and outputs are strictly non-functional. Under this  
420 version of  $\text{MERGE}$  a step of external merge is divided into two steps as in (48).

421 (48) a.  $\text{MERGE}(\text{W}) \rightarrow \text{MERGE}^{\text{W}}$

422 b.  $\text{MERGE}^{\text{W}}(\text{X}) \rightarrow \text{MERGE}^{\text{W},\text{X}} \rightarrow \{\text{X}, \text{W}(1)\} + ((\text{W} - \text{X}) - \text{W}(1))$

423 Note here that, since lambda abstraction and reduction is sensitive only to the form of the variables,  
424 the order of these steps, dictated by the order of lambda expressions in (47), is arbitrary. We could, in  
425 principle, reorder the lambda expressions in (47) and we would have a different order of operations  
426 in (48) with the same result. This fact will come into play shortly.

### 427 6.1.3.2 The map function

428 In the previous section I noted that curried functions are a class of higher-order functions because  
429 they have functions as outputs. In this section I will introduce a higher-order function that takes  
430 functions as inputs—the map function—which will be key to achieving lockstep parallel derivations.  
431 Informally speaking, map takes a function and applies it to a list of arguments. Formally, map is

432 defined in (49).

$$433 \quad (49) \quad \text{map}(f, \langle x_0, x_1, \dots, x_n \rangle) \rightarrow \langle f(x_0), f(x_1), \dots, f(x_n) \rangle$$

434 Now, let's consider how lockstep parallel derivations would proceed. The stage at which the lockstep  
435 derivation begins was given in (39) and repeated here as (50).

$$436 \quad (50) \quad \left\langle \left[ \left[ \{ \text{make}, \{ \text{the}, \text{error} \} \}, \text{Voice}, \dots, T \right]_{\text{WS1}}, \right. \right. \\ \left. \left. \left[ \{ \text{deliberately} \}, \text{Voice}, \dots, T \right]_{\text{WS2}}, [\text{Sharon}]_{\text{WS3}} \right] \right\rangle$$

437 The next step is to merge Voice in WS1 and WS2 and to do that we start with MERGE curried  
438 in the reverse order of (47), shown in (51), with  $\alpha$  and  $\omega$  ranging over SOs and workspaces,  
439 respectively. Note, though, that R-MERGE is not a newly proposed operation. It has the same  
440 intension as MERGE—represented as  $\mathcal{M}$ —with inverted lambda terms.

$$441 \quad (51) \quad \text{R-MERGE} = (\lambda \alpha. (\lambda \omega. \mathcal{M}))$$

442 Our first step, then, is to apply R-MERGE to Voice as in (52)

$$443 \quad (52) \quad \text{R-MERGE}(\text{Voice}) \rightarrow \text{R-MERGE}^{\text{Voice}}$$

444 Next we map this function to WS1 and WS2 as in (53).

$$445 \quad (53) \quad \text{map}(\text{R-MERGE}^{\text{Voice}}, \langle \text{WS1}, \text{WS2} \rangle) \rightarrow \left\langle \left[ \left[ \text{Voice}, \{ \text{make}, \{ \text{the}, \text{error} \} \}, \dots, T \right], \right. \right. \\ \left. \left. \left[ \left[ \text{Voice} \{ \text{deliberately} \}, \dots, T \right] \right] \right\rangle$$

446 And so on like that for the remainder of the derivation. Thus we can derive (35).

### 447 **6.1.3.3 Identity across workspaces**

448 If (52) and (53) are two steps in the derivation on (35), we still need to explain how the two  
449 instances of Voice can be considered copies of each other in order to explain how one of them  
450 deletes.

451 I mentioned in section 6.1.2 that, under a derivational theory of syntax, copies can be distinguished  
452 from repetitions in that the former share a derivational history, while the latter do not. In order  
453 for two objects to share a derivational history, they must have the same origin. The origin of any

454 syntactic object in a given derivation is a tokening operation (Select in terms of Collins and Stabler  
455 (2016)) in the case of lexical item tokens or a subderivation in the case of derived objects like  
456 complex nominals.

457 In the case of Voice, since it a lexical item token, it's two instances in (35) must be linked by a  
458 single instance of the tokening operation Select, defined in (54).

459 (54)  $\text{Select}(\alpha, \omega) \rightarrow \omega + \alpha$

460 Where  $\alpha$  is a lexical item and  $\omega$  is a workspace

461 Of course, this operation can be curried as in (55) and mapped so that a single instance of Select  
462 can put a single token in two workspaces as in (56)

463 (55)  $(\lambda \alpha. (\lambda \omega. \omega + \alpha))$

464 (56) a.  $\text{Select}(\text{Voice}) \rightarrow \text{Select}^{\text{Voice}}$

465 b.  $\text{Map}(\text{Select}^{\text{Voice}}, \langle \text{WS1}, \text{WS2} \rangle) \rightarrow \langle \text{WS1} + \text{Voice}, \text{WS2} + \text{Voice} \rangle$

466 So, the two instances of Voice share a single tokening operation, and therefore are the same object.<sup>6</sup>

## 467 **7 Problems solved by this theory**

468 In this section, I will outline a few problems related to adjunction that the proposed theory provides  
469 natural solutions to. First, I will address the island-hood of adjuncts. Then, I will discuss parasitic  
470 gaps, whereby adjunct island-effects are ameliorated. Finally, I will discuss a problem to do with  
471 ACC-ing clauses which was, to my knowledge, first revealed by Cinque (1996) and made explicit in  
472 a separate manuscript (Author forthcoming).

---

<sup>6</sup>This also seems to be how we identify individual objects in general: I am the same individual as I was last year because both versions of me share the same birth event—the same origin.

473 **7.1 The Island-hood of adjuncts**

474 A well-known property of adjuncts is that they are islands to movement. Indeed, Bošković (forth-  
 475 coming) points out that, while the island-hood of many other constructions varies across languages,  
 476 adjunct island-hood seems to be constant.<sup>7</sup> So, for instance (57) is an ungrammatical question,  
 477 and (58) is contains an ungrammatical relative clause because they both require an instance of  
 478 *wh*-movement out of an adjunct.

479 (57) \*What<sub>i</sub> did she eat an apple [after washing \_\_\_<sub>i</sub>]?  
 480

(58) \*The student who<sub>i</sub> he invited Barbara [without meeting \_\_\_<sub>i</sub>]

481 To see how the theory of adjuncts I propose here predicts adjunct island-hood consider the stage  
 482 of the derivation of (57) immediately before *wh*-movement occurs. As shown in (59), the *wh*-  
 483 expression *what* is in the adjunct workspace (WS2), which “scopes over” the TP. Note that both  
 484 workspaces contain a C<sub>*wh*</sub> head.

485 (59)  $\left\langle \begin{array}{l} [\{C_{wh}, \{she, \{T, \dots\}\}]_{WS1}, \\ [\{C_{wh}, \{after, \{washing, what\}\}]_{WS2} \end{array} \right\rangle$

486 In order to derive (57), we would need a *wh*-movement operation such as (60).

487 (60) MERGE(WS1)(*what*)

488 The result of this operation, however, is undefined because *what* is neither a member of WS1, nor  
 489 contained in the root object of WS1.

490 The operation in (61), on the other hand, is defined yielding the stage in (62).

491 (61) MERGE(WS2)(*what*)

492 (62)  $\left\langle \begin{array}{l} [\{C_{wh}, \{she, \{T, \dots\}\}]_{WS1}, \\ [\{what\{C_{wh}, \{after, \{washing, what\}\}\}]_{WS2} \end{array} \right\rangle$

493 This stage is problematic for two reasons. First, the C<sub>*wh*</sub> head in WS1 would bear an unsatisfied

---

<sup>7</sup>Bošković notes that, since the Coordinated Structure Constraint is also constant across languages, it should be unified with adjunct island-hood.

494 *wh*-feature which would lead to a crash at the CI interface. Second, (62) would not yield (57) when  
 495 linearized because *what*, being in WS2 would ordered after all of the words in WS1. That is, we  
 496 would expect (62) to be linearized as (63).

497 (63) \*She ate an apple what after washing

498 Thus the island-hood of adjuncts follows naturally from my proposed theor of adjuncts.

## 499 7.2 Parasitic Gaps

500 The island-hood of adjuncts, though constant across languages, is circumvented in so-called parasitic  
 501 gap constructions (Engdahl 1983) as in (64) and (65).<sup>8</sup>

502 (64) What<sub>i</sub> did she eat \_\_\_<sub>i</sub> [after washing *ec*<sub>i</sub>]?

503 (65) The student who he invited \_\_ [without meeting *ec*<sub>i</sub>]

504 Here the parasitic gaps in the adjuncts, represented here as *ecs*, are licensed if there is a parallel  
 505 trace in the host. This required parallelism is both syntactic—the trace and the parasitic gap have the  
 506 same grammatical role (*i.e.* direct object in (64) and (65))—and semantic—the trace and parasitic  
 507 gap co-refer.

508 Here, the mechanism for ensuring lockstep derivation—higher-order functions—allows us to derive  
 509 parasitic gaps. To demonstrate this, consider the penultimate stage in the derivation of (64) shown  
 510 in (66).

511 (66)  $\left\langle \left[ \left[ C_{wh}, \{she, \{T, \{\dots, what_i\}\}\} \right]_{WS1}, \right. \right. \\ \left. \left. \left[ \left[ C_{wh}, \{after, \{washing, what_i\}\}\} \right]_{WS2} \right] \right\rangle$

512 Note that the two instances of *what* here are copies of each other, meaning they share a derivational  
 513 origin. The final stage of (64), given in (68) is derived in two steps given in (67).

514 (67) a. R-MERGE(*what*<sub>i</sub>) → R-MERGE<sup>*what*<sub>i</sub></sup>

---

<sup>8</sup>I represent the gaps within the adjuncts here as *{ec}*s because, depending on the analysis, they are alternately identified as traces of movement or null proforms.

515 b.  $\text{map}(\text{R-MERGE}^{\text{what}_i}, \langle \text{WS1}, \text{WS2} \rangle) \rightarrow (68)$

516 (68)  $\left\langle \left[ \left[ \left[ \text{what}_i \{ C_{wh}, \{ she, \{ T, \{ \dots, \text{what}_i \} \} \} \} \right] \right] \text{WS1}, \right. \right. \\ \left. \left. \left[ \left[ \text{what}_i \{ C_{wh}, \{ after, \{ washing, \text{what}_i \} \} \} \right] \right] \text{WS2} \right] \right\rangle$

517 As discussed in section 6.1.2, all instances of  $\text{what}_i$  except for the highest instance in the first  
518 workspace is deleted, yielding the string (64).

519 Thus parasitic gaps are naturally accounted for in the theory I propose here.

### 520 7.3 Cartography's facts

521 In section 2 I discussed the adjective ordering restriction as an example of the class of facts  
522 which motivate the cartographic approach to adjuncts. I argued that adjective ordering restrictions  
523 (*e.g.*, SIZE > SHAPE) and stronger word orderings (*e.g.*, D > N in English) are different sorts of  
524 phenomena. This conclusion does not, however, mean that adjective ordering restrictions are not  
525 real, and therefore don't need explanation. Rather, it means that they must be explained in a way  
526 different from the stronger word-order restrictions. A workspace-theoretic approach can provide  
527 such a different explanation, given a few auxiliary hypotheses.

528 To begin, I give the derivation of (5)—a nominal phrase with an acceptable adjective sequence—in  
529 (69), followed by the derivation of (6)—a nominal phrase with a deviant adjective sequence—in  
530 (70).<sup>9</sup>

---

<sup>9</sup>I leave out Select operations for the sake of brevity.



(Start)	$\left\langle \begin{array}{l} [\{small\}, SIZE]_{WS1}, \\ [\{square\}, SIZE, SHAPE]_{WS2}, \\ [\sqrt{TABLE}, n, SIZE, SHAPE]_{WS3} \end{array} \right\rangle$	0
MERGE( $n$ )(WS3)	$\rightarrow \left\langle \begin{array}{l} [\{small\}, SIZE]_{WS1}, \\ [\{square\}, SIZE, SHAPE]_{WS2}, \\ [\{\sqrt{TABLE}, n\}, SIZE, SHAPE]_{WS3} \end{array} \right\rangle$	1
Map(MERGE(SHAPE))( $\langle$ WS2,WS3 $\rangle$ )	$\rightarrow \left\langle \begin{array}{l} [\{small\}, SIZE]_{WS1}, \\ [\{SHAPE, square\}, SIZE]_{WS2}, \\ [\{SHAPE, \{n, \sqrt{TABLE}\}\}, SIZE]_{WS3} \end{array} \right\rangle$	2
Map(MERGE(SIZE))( $\langle$ WS1,WS2,WS3 $\rangle$ )	$\rightarrow \left\langle \begin{array}{l} [\{SIZE, small\}]_{WS1}, \\ [\{SIZE, \{SHAPE, square\}\}]_{WS2}, \\ [\{SIZE, \{SHAPE, \{n, \sqrt{TABLE}\}\}\}]_{WS3} \end{array} \right\rangle$	3

(Start)	$\left\langle \begin{array}{l} [\{square\}, SIZE, SHAPE]_{WS1}, \\ [\{small\}, SIZE]_{WS2}, \\ [\sqrt{TABLE}, n, SIZE, SHAPE]_{WS3} \end{array} \right\rangle$	0
MERGE( $n$ )(WS3)	$\rightarrow \left\langle \begin{array}{l} [\{square\}, SIZE, SHAPE]_{WS1}, \\ [\{small\}, SIZE]_{WS2}, \\ [\{\sqrt{TABLE}, n\}, SIZE, SHAPE]_{WS3} \end{array} \right\rangle$	1
Map(MERGE(SHAPE))( $\langle$ WS1,WS3 $\rangle$ )	$\rightarrow \left\langle \begin{array}{l} [\{SHAPE, square\}, SIZE]_{WS1}, \\ [\{small\}, SIZE]_{WS2}, \\ [\{SHAPE, \{n, \sqrt{TABLE}\}\}, SIZE]_{WS3} \end{array} \right\rangle$	2
Map(MERGE(SIZE))( $\langle$ WS1,WS2,WS3 $\rangle$ )	$\rightarrow \left\langle \begin{array}{l} [\{SIZE, \{SHAPE, square\}\}]_{WS1}, \\ [\{SIZE, small\}]_{WS2}, \\ [\{SIZE, \{SHAPE, \{n, \sqrt{TABLE}\}\}\}]_{WS3} \end{array} \right\rangle$	3

533 The key point of comparison here is between respective second steps, in which SHAPE is merged.

534 In (69), this step maps MERGE(SHAPE) to a contiguous sub-sequence of the active workspaces.  
535 In (70), on the other hand, this step maps the same carried function to a non-contiguous sub-  
536 sequence. If we make the auxiliary hypothesis that mapping over a contiguous sequence is more  
537 computationally efficient than mapping over a non-contiguous sequence, then we have a possible  
538 explanation of the deviance of (6) and, by extension, a possible explanation of adjunct ordering  
539 restrictions. That is, violations of adjunct ordering restrictions, rather than being violations of  
540 selection restrictions, are the result of suboptimal derivations.

541 Note, however, that this approach maintains the universal hierarchy proposed by cartographers with  
542 a few alterations. Under the cartographer’s approach, adjuncts merge with their respective functional  
543 heads as specifiers, and a functional head selects its subordinate category as a complement. This  
544 seems to lead to the conclusion that the entire functional sequence of a domain is merged in every  
545 derivation of that domain. Under the present approach, adjuncts still merge with their respective  
546 functional heads, but as complements. That is, the structural relation between functional heads, like  
547 SIZE, and modifiers, like *small*, is the same as the relation between roots and their categorizing  
548 heads. It follows from this that modifiers merged with the interpretive relation between functional  
549 head and modifier should be the same as the one between categorizing heads and roots. This  
550 prediction is borne out in the intuitive understanding of polysemy.<sup>10</sup>

551 Consider, for instance, how one would define the word *work*. Since it is polysemous we would have  
552 to give a list of definitions—we would say “*work* as a noun means . . .” followed by “*work* as a verb  
553 means . . .”, or vice versa. We could formalize these as in (71).

554 (71) a.  $SEM(\{n, \sqrt{WORK}\}) = \dots$

555 b.  $SEM(\{v, \sqrt{WORK}\}) = \dots$

556 Now compare this to the adjective *light* which is many ways polysemous. Our list of definitions  
557 would be as follows—“*light* as a colour adjective means . . .”, “*light* as a weight adjective means  
558 . . .”, “*light* as an evaluative adjective means . . .”, and so on. Again, we can formalize these as in

---

<sup>10</sup>There seems to be no systematic account of grammatical category or polysemy in current semantic theory.

559 (72).

560 (72) a.  $SEM(\{COLOUR, light\}) = \dots$

561 b.  $SEM(\{WEIGHT, light\}) = \dots$

562 c.  $SEM(\{VALUE, light\}) = \dots$

563 In both cases, we replace the *as-a* relation with the head-complement relation. If such a move  
564 were made in isolation, it would be quite innocuous, even trivial. In the current context,  
565 though, the move was a logical result of a substantive hypothesis and should, therefore, be seen as  
566 corroborating evidence in favour of that hypothesis.

## 567 **8 Conclusion**

568 I have argued in this paper that the basic facts about adjuncts only make sense if we assume that  
569 adjuncts are not truly attached to their hosts. While previous theories of grammar have not offered  
570 any way of formalizing this assertion, I proposed that the relatively new notion of workspaces  
571 offers such a possibility. That is, I proposed that adjuncts, like arguments, are derived in their  
572 own workspaces, but, unlike arguments, they are not incorporated into the “main” workspace. I  
573 formalized this proposal and, in the process, proposed a workspace-based formalization of MERGE.  
574 I then applied this formalized proposal to some generalizations related to adjunct—Islands, Parasitic  
575 Gaps, and adjective ordering constraints—showing that those generalizations are either predicted by  
576 my proposal or consistent with it.

## 577 **References**

578 Bošković, Željko. Forthcoming. “On Unifying the Coordinate Structure Constraint and the Adjunct  
579 Condition.” In *Rethinking Grammar (Festschrift for Ian Roberts)*, edited by A. Bárány, T. Biberauer,  
580 J. Douglas, and S. Vikner.

581 Chomsky, Noam. 2019. “Some Puzzling Foundational Issues: The Reading Program.” *Catalan*

582 *Journal of Linguistics*, 263–85.

583 Cinque, Guglielmo. 1996. “The Pseudo-Relative and ACC-Ing Constructions After Verbs of  
584 Perception.” In *Italian Syntax and Universal Grammar*, by Guglielmo Cinque, 244–75. Cambridge:  
585 Cambridge University Press.

586 Cinque, Guglielmo, and Luigi Rizzi. 2010. “The Cartography of Syntactic Structures.” In *Oxford*  
587 *Handbook of Linguistic Analysis*, edited by Bernd Heine and Heiko Narrog. Oxford University  
588 Press.

589 Citko, Barbara. 2005. “On the Nature of Merge: External Merge, Internal Merge, and Parallel  
590 Merge.” *Linguistic Inquiry* 36 (4). MIT Press: 475–96.

591 Collins, Chris, and Edward Stabler. 2016. “A Formalization of Minimalist Syntax.” *Syntax* 19 (1):  
592 43–78. <https://doi.org/10.1111/synt.12117>.

593 Engdahl, Elisabet. 1983. “Parasitic Gaps.” *Linguistics and Philosophy*. JSTOR, 5–34.

594 Nunes, Jairo. 2004. *Linearization of Chains and Sideward Movement*. Vol. 43. MIT press.

595 Sproat, Richard, and Chilin Shih. 1991. “The Cross-Linguistic Distribution of Adjective Ordering  
596 Restrictions.” In *Interdisciplinary Approaches to Language*, 565–93. Springer.