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.[1] 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 ague 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.

---

[1] Also in this group is conjunction.

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

2 The essence of adjuncts

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

(1) [$_{\text{X}}$ The police silenced the workers].

(2) [$_{\text{Y}}$ [$_{\text{X}}$ The police silenced the workers], [against their wills]].

(3) [$_{\text{Z}}$ [$_{\text{Y}}$ [$_{\text{X}}$ The police silenced the workers], [against their wills]], [following the demonstration]].

(4) [$_{\text{U}}$ [$_{\text{W}}$ [$_{\text{X}}$ The police silenced the workers] [following the demonstration]], [against their wills]].

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

---

[2] These properties are generalizations to which, no doubt, there are exceptions. I ask the reader to attach "all else being equal" to such generalizations.

[3] See Sproat and Shih (1991) for further discussion of the adjective ordering restriction

(5) a small square table

(6) $^{?*}$a square small table

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

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

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

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

(8) a. *square a small table
b. *square small a table
c. *square small table a

(9) a. *a square table small
b. *a table square small
c. *table a square small

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

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

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

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

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

(10) The cat ate the fish.

(11) The morning star

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

(12) a. hit
b. hit the ball

(13) a. ate the cake

b. Juan ate the cake

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

(14) a. The toddler hit the ball.
b. *The toddler hit

(15) a. I believe Juan ate the cake.
b. *I believe ate the cake.

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

(16) a. Rosie [sang the anthem].
b. Rosie [sang the anthem with gusto].

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

The reasoning to this conclusion is straightforward. By definition, Merge combines two objects α and β to create a new object γ that is distinct from both α and β. So, if a host-adjunct expressions $\mathrm{H}^\frown\mathrm{A}$ were formed by Merging H and A, it would be distinct. All host-adjunct expressions $\mathrm{H}^\frown\mathrm{A}$ are equivalent to their host H and therefore cannot be formed by Merge.

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

3 Adjunction is optional

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

This can be seen as analogous to one of Chomsky's argument for transformations in *Syntactic Structures*, which goes roughly as follows: phrase structure rules alone are not sufficient to generate all the sentences of natural language, therefore we must augment them with transformations.[4] Although there were other arguments for transformations, I would like to compare their necessity to the supposed necessity of Pair-Merge or late Merge.

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

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

---

[4] This is, of course, an oversimplification of the argument, but it will do for our current purposes. As Chomsky (1965) points out, PSRs are powerful enough to capture the *weak generative capacity* of language, but crucially not the *strong generative capacity*. Furthermore, the addition of transformations to our theory of grammar allowed us to explain the fact that expressions can be related to each other in a number of ways.

and (18).

(17) Violet wrote the anthem.

(18) What did Violet write?

Phrase structure rules, however, can generate declaratives, but not interrogatives. Therefore, phrase structure rules are insufficient as a theory of natural language. Important to note is that both premises rest on other premises. The first premise assumes that declaratives and interrogatives express distinct classes of thoughts, while the second premise assumes a particular version of phrase structure rules.

The necessity argument for Pair-Merge or late Merge goes as follows. Natural language can express thought with or without adjuncts as shown in (19) and (20)

(19) Rosie sang the anthem.

(20) Rosie sang the anthem with gusto.

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

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

(21) The singing had gusto.

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

4 A paradox?

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

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

5 Towards a theory of Adjuncts

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

5.1 Workspaces

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

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

(22) The citizens wore some masks.

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

5.2 Deletion

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

(23) What did the student hear?

(24) What did the student hear what

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

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

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

This principle accurately predicts (23), but faces several issues. Empirically speaking, there are many apparent exceptions to (25). Theoretically speaking, the notion of identity is not fully understood and, as I will discuss in section 6.1.2, c-command as the deciding factor is too narrow.

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

6 A workspace-based theory of adjunction

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

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

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

(27) Stage N: $\langle[\{sing,\{the,anthem\}\},\mathrm{Voice}]_{\mathrm{WS1}}\rangle$ (Merge(Voice, WS1))
Stage N+1: $\langle[\{\mathrm{Voice},\{sing,\{the,anthem\}\}]_{\mathrm{WS1}}\rangle$

(28) Stage N: $\langle[\{sing,\{the,anthem\}\},\mathrm{Voice}]_{\mathrm{WS1}},[\{with,gusto\}]_{\mathrm{WS2}}\rangle$ (Merge(Voice, WS1))
Stage N+1: $\langle[\{\mathrm{Voice},\{sing,\{the,anthem\}\}]_{\mathrm{WS1}},[\{with,gusto\}]_{\mathrm{WS2}}\rangle$

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

In terms of interpretation, this proposal makes roughly the correct prediction/ That is, the host and adjunct are distinct syntactic objects and, therefore, would be interpreted as such. Recall, in section 3, I argued that an expression with an adjunct could be expressed as two juxtaposed expressions. This analysis formalizes that intuition.

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

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

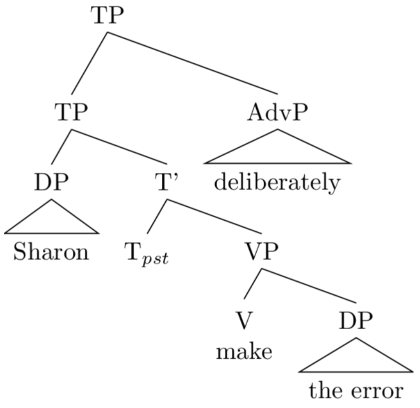
6.1 The problem of adjunct scope

The sentence in (29) is ambiguous.

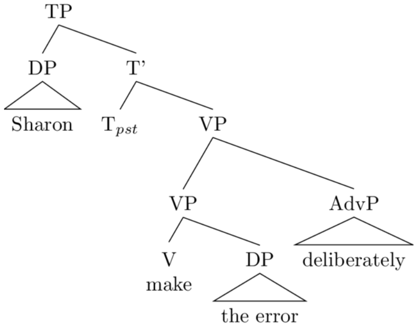
(29) Sharon made the error deliberately.

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

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



(31) **The low-scope interpretation of (29) in X-bar theory**



As it stands, however, the workspace theory of adjuncts cannot account for adjunct scope. Or, to be more precise, it cannot account for the fact that adjuncts can have multiple scope possibilities. This can be seen when we consider how we would represent (29) in a workspace-based analysis—as the juxtaposition of *Sharon made the error* and *deliberately* as shown in (32).

(32) $$\left\langle \begin{array}{c} [\{\mathit{Sharon}, \{\mathrm{T}, \ldots \{\mathrm{Voice}, \{\mathit{make}, \{\mathit{the}, \mathit{error}\}\}\}\}\}], \\ [\mathit{deliberately}] \end{array} \right\rangle$$

If we take a full declarative clause to describe a situation or state of affairs, then, according to (32), (29) would describe a situation *s*, such that in *s* Sharon made the relevant error, and that *s* was brought about by a deliberate choice of the agent of *s*. In other words, the proposed workspace-based theory of adjuncts seems to predict only the high-scope interpretation of (29).

In order to modify our proposal to allow for adjunct scope, we must first realize that adjunct scope-taking is different from other kinds of scope-taking, such as quantifier scope. Usually, when we talk about scope, we have in mind an asymmetric relation. So the two readings of (33) can be described by saying which of the two quantifier phrases scopes over the other.

(33) Every student read a book.

a. $\forall s(\exists b(read(b,s)))$

b. $\exists b(\forall s(read(b,s)))$

The relationship between a modifier and a modified expression, however, is generally considered to be symmetric, at least in terms of their interpretation.[5] So, in the low-scope interpretation of (29), the logical predicate expressed by *deliberately* is conjoined with the one expressed by *make an error*, as shown in (34).

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

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

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

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

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

1. How is (35) interpreted?
2. How is (35) pronounced?
3. How is (35) derived?

---

[5] Setting aside cases of non-intersective modification.

I address these three questions in turn directly.

6.1.1 How is (35) interpreted?

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

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

(36) $\lambda e(make(e)\,\&\,\textsc{Agent}(e)(\textbf{sharon})\,\&\,\textsc{Theme}(e)(\textbf{the-error}))$

(37) $\lambda e(\textsc{Agent}(e)(\textbf{sharon})\,\&\,deliberately(e))$

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

(38) $\lambda e(make(e)\,\&\,\textsc{Agent}(e)(\textbf{sharon})\,\&\,\textsc{Theme}(e)(\textbf{the-error})\,\&\,deliberately(e))$

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

6.1.2 How is (35) pronounced?

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

The notion of identity here, must capture copies, but not repetitions, so in order for the various phrases and heads to be deleted from the adjunct we must show that they can be treated as copies of the corresponding phrases and heads in the host. since the distinction between copies and repetitions is to follow from the derivational history of an expression, I will postpone the question of identity until the following section and stipulate, for the moment, that *Sharon*, T, Voice, *etc*. in the adjunct are considered copies of their counterparts in the host.

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

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

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

6.1.3 How is (35) derived?

The derivation of host-adjunct structures such as (35) can be divided into to parts. In the first part, the two workspaces—host and adjunct— are derived independently of each other, and in the second part, the workspaces are derived in lockstep. The first part represents the standardly assumed operation of workspaces, and is, therefore, already understood, at least insofar as workspaces are understood. The second part—the part involving lockstep derivation—is novel and its explanation will occupy this section.

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

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

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

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

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

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

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

6.1.3.1 MERGE and its formal definition

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

(40) A derivation D is a finite sequence of stages $\langle \mathrm{S}_1, \mathrm{S}_2,\ldots,\mathrm{S}_\mathrm{n}\rangle$, where D(i) = S_i.

(41) A stage S is a finite sequence of workspaces $\langle \mathrm{WS}_1, \mathrm{WS}_2,\ldots \mathrm{WS}_\mathrm{n}\rangle$, where S(i) = WS_i.

(42) A workspace WS is a finite sequence of syntactic objects $\langle \mathrm{SO}_1, \mathrm{SO}_2,\ldots \mathrm{SO}_\mathrm{n}\rangle$, where WS(i) = SO_i.

In addition to the workspace desideratum, MERGE should also "restrict computational resources" (Chomsky 2019), by ensuring that when a new object is created bye MERGE, its constituent parts do not remain in accessible in the workspace. That is, MERGE substitutes the new object for the old objects. The definition of MERGE in (43), where "+" represents an "append" operation and "−" represents a "delete" operation, meets the two desiderata that I have mentioned thus far.[6]

(43) Where ω is a workspace, and α and β are syntactic objects,

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

---

[6] 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.

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

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

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

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

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

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

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

(45) Where ω is a workspace, and α is a syntactic object,

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

I have restricted merge here by identifying a privileged member of a given workspace—the first member $\omega(1)$. This is what is sometimes referred to as the root of the tree. This is a justifiable step in that the first member of a workspace has a unique property among workspace members—the existence of a workspace depends only on the existence of its first member. That is, there are workspaces of length 1, 2, 3, *etc* but no workspaces of length 0. A corollary of this is that the proposition in (46) is only true for $i = 1$.

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

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

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

(47) $\mathrm{MERGE} = (\lambda\omega.(\lambda\alpha..\mathscr{M}))$

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

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

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

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

6.1.3.2 The `map` function

In the previous section I noted that curried functions are a class of higher-order functions because they have functions as outputs. In this section I will introduce a higher-order function that takes functions as inputs—the `map` function—which will be key to achieving lockstep parallel derivations. Informally speaking, `map` takes a function and applies it to a list of arguments. Formally, `map` is defined in (49).

(49) $\texttt{map}(f,\ \langle x_0, x_1, \ldots x_n\rangle) \rightarrow \langle f(x_0), f(x_1), \ldots f(x_n)\rangle$

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

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

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

(51) R-MERGE = $(\lambda\alpha.(\lambda\omega.\mathscr{M}))$

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

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

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

(53) $$\texttt{map}(\text{R-MERGE}^{\text{Voice}},\ \langle \text{WS1, WS2}\rangle) \rightarrow \left\langle \begin{array}{l} [\{\text{Voice}, \{make, \{the, error\}\}\}, \ldots, T], \\ [\{\text{Voice}\{deliberately\}\}, \ldots, T] \end{array} \right\rangle$$

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

6.1.3.3 Identity across workspaces

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

I mentioned in section 6.1.2 that, under a derivational theory of syntax, copies can be distinguished from repetitions in that the former share a derivational history, while the latter do not. In order for two objects to share a derivational history, they must have the same origin. The origin of any syntactic object in a given derivation is a tokening operation (Select in terms of Collins and Stabler (2016)) in the case of lexical item tokens or a subderivation in the case of derived objects like complex nominals.

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

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

Where α is a lexical item and ω is a workspace

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

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

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

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

So, the two instances of Voice share a single tokening operation, and therefore are the same object.[7]

7 Problems solved by this theory

In this section, I will outline a few problems related to adjunction that the proposed theory provides natural solutions to. First, I will address the island-hood of adjuncts. Then, I will discuss parasitic gaps, whereby adjunct island-effects are ameliorated. Finally, I will discuss a class of facts commonly associated with Cartographic/Nanosyntactic approaches to syntax—adjunct ordering constraints.

---

[7] 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.

7.1 The Island-hood of adjuncts

A well-known property of adjuncts is that they are islands to movement. Indeed, Bošković (forthcoming) points out that, while the island-hood of many other constructions varies across languages, adjunct island-hood seems to be constant.[8]So, for instance (57) is an ungrammatical question, and (58) is contains an ungrammatical relative clause because they both require an instance of *wh*-movement out of an adjunct.

(57) *What$_{\text{i}}$ did she eat an apple [after washing __$_{\text{i}}$]?

(58) *The student who$_{\text{i}}$ he invited Barbara [without meeting __$_{\text{i}}$]

To see how the theory of adjuncts I propose here predicts adjunct island-hood consider the stage of the derivation of (57) immediately before *wh*-movement occurs. As shown in (59), the *wh*-expression *what* is in the adjunct workspace (WS2), which "scopes over" the TP. Note that both workspaces contain a C$_{wh}$ head.

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

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

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

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

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

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

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

This stage is problematic for two reasons. First, the C$_{wh}$ head in WS1 would bear an unsatisfied

---

[8] Bošković notes that, since the Coordinated Structure Constraint is also constant across languages, it should be unified with adjunct island-hood.

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

(63) *She ate an apple what after washing

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

7.2 Parasitic Gaps

The island-hood of adjuncts, though constant across languages, is circumvented in so-called parasitic gap constructions (Engdahl 1983) as in (64) and (65).[9]

(64) What$_{\mathrm{i}}$ did she eat __$_{\mathrm{i}}$ [after washing *ec*$_{\mathrm{i}}$]?

(65) The student who he invited __ [without meeting *ec*$_{\mathrm{i}}$]

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

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

(66)
$$\left\langle \begin{array}{l} [\{\mathrm{C}_{wh}, \{she, \{\mathrm{T}, \{\ldots, what_i\}\}\}\}]_{\mathrm{WS1}}, \\ [\{\mathrm{C}_{wh}, \{after, \{washing, what_i\}\}\}]_{\mathrm{WS2}} \end{array} \right\rangle$$

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

(67) a. R-MERGE($what_{\mathrm{i}}$) → R-MERGE$^{what_{\mathrm{i}}}$

---

[9] 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.

b. $\texttt{map}(\text{R-MERGE}^{what_{\mathrm{i}}}, \langle \text{WS1}, \text{WS2} \rangle) \rightarrow (68)$

(68) $$\left\langle \begin{array}{l} [\{what_i\{\mathrm{C}_{wh}, \{she, \{\mathrm{T}, \{\ldots, what_i\}\}\}\}\}]_{\mathrm{WS1}}, \\ [\{what_i\{\mathrm{C}_{wh}, \{after, \{washing, what_i\}\}\}\}]_{\mathrm{WS2}} \end{array} \right\rangle$$

As discussed in section 6.1.2, all instances of $what_{\mathrm{i}}$ except for the highest instance in the first workspace is deleted, yielding the string (64).

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

7.3 Cartography's facts

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

To begin, I give the derivation of (5)—a nominal phrase with an acceptable adjective sequence—in (69), followed by the derivation of (6)—a nominal phrase with a deviant adjective sequence— in (70).[10]

---

[10] I leave out Select operations for the sake of brevity.

(69)

(Start)		$\left\langle \begin{array}{l} [\{\mathit{small}\}, \textsc{Size}]_{WS1}, \\ [\{\mathit{square}\}, \textsc{Size}, \textsc{Shape}]_{WS2}, \\ [\sqrt{\text{TABLE}}, n, \textsc{Size}, \textsc{Shape}]_{WS3} \end{array} \right\rangle$	0
MERGE(n)(WS3)	→	$\left\langle \begin{array}{l} [\{\mathit{small}\}, \textsc{Size}]_{WS1}, \\ [\{\mathit{square}\}, \textsc{Size}, \textsc{Shape}]_{WS2}, \\ [\{\sqrt{\text{TABLE}}, n\}, \textsc{Size}, \textsc{Shape}]_{WS3} \end{array} \right\rangle$	1
Map(MERGE($\textsc{Shape}$))($\langle$WS2,WS3$\rangle$)	→	$\left\langle \begin{array}{l} [\{\mathit{small}\}, \textsc{Size}]_{WS1}, \\ [\{\textsc{Shape}, \mathit{square}\}, \textsc{Size}]_{WS2}, \\ [\{\textsc{Shape}, \{n, \sqrt{\text{TABLE}}\}\}, \textsc{Size}]_{WS3} \end{array} \right\rangle$	2
Map(MERGE($\textsc{Size}$))($\langle$WS1,WS2,WS3$\rangle$)	→	$\left\langle \begin{array}{l} [\{\textsc{Size}, \mathit{small}\}]_{WS1}, \\ [\{\textsc{Size}, \{\textsc{Shape}, \mathit{square}\}\}]_{WS2}, \\ [\{\textsc{Size}, \{\textsc{Shape}, \{n, \sqrt{\text{TABLE}}\}\}\}]_{WS3} \end{array} \right\rangle$	3

(70)

(Start)		$\left\langle \begin{array}{l} [\{\mathit{square}\}, \textsc{Size}, \textsc{Shape}]_{WS1}, \\ [\{\mathit{small}\}, \textsc{Size}]_{WS2}, \\ [\sqrt{\text{TABLE}}, n, \textsc{Size}, \textsc{Shape}]_{WS3} \end{array} \right\rangle$	0
MERGE(n)(WS3)	→	$\left\langle \begin{array}{l} [\{\mathit{square}\}, \textsc{Size}, \textsc{Shape}]_{WS1}, \\ [\{\mathit{small}\}, \textsc{Size}]_{WS2}, \\ [\{\sqrt{\text{TABLE}}, n\}, \textsc{Size}, \textsc{Shape}]_{WS3} \end{array} \right\rangle$	1
Map(MERGE($\textsc{Shape}$))($\langle$WS1,WS3$\rangle$)	→	$\left\langle \begin{array}{l} [\{\textsc{Shape}, \mathit{square}\}, \textsc{Size}]_{WS1}, \\ [\{\mathit{small}\}, \textsc{Size}]_{WS2}, \\ [\{\textsc{Shape}, \{n, \sqrt{\text{TABLE}}\}\}, \textsc{Size}]_{WS3} \end{array} \right\rangle$	2
Map(MERGE($\textsc{Size}$))($\langle$WS1,WS2,WS3$\rangle$)	→	$\left\langle \begin{array}{l} [\{\textsc{Size}, \{\textsc{Shape}, \mathit{square}\}\}]_{WS1}, \\ [\{\textsc{Size}, \mathit{small}\}]_{WS2}, \\ [\{\textsc{Size}, \{\textsc{Shape}, \{n, \sqrt{\text{TABLE}}\}\}\}]_{WS3} \end{array} \right\rangle$	3

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

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

Note, however, that this approach maintains the universal hierarchy proposed by cartographers with a few alterations. Under the cartographer's approach, adjuncts merge with their respective functional heads as specifiers, and a functional head selects its subordinate category as a complement. This seems to lead to the conclusion that the entire functional sequence of a domain is merged in every derivation of that domain. Under the present approach, adjuncts still merge with their respective functional heads, but as complements. That is, the structural relation between functional heads, like SIZE, and modifiers, like *small*, is the same as the relation between roots and their categorizing heads. It follows from this that modifiers merged with the interpretive relation between functional head and modifier should be the same as the one between categorizing heads and roots. This prediction is borne out in the intuitive understanding of polysemy.[11]

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

(71) a. $\text{SEM}(\{n, \sqrt{\text{WORK}}\}) = \ldots$
b. $\text{SEM}(\{v, \sqrt{\text{WORK}}\}) = \ldots$

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

---

[11] There seems to be no systematic account of grammatical category or polysemy in current semantic theory.

(72).

(72) a. $\mathrm{SEM}(\{\textsc{Colour}, \textit{light}\}) = \ldots$
b. $\mathrm{SEM}(\{\textsc{Weight}, \textit{light}\}) = \ldots$
c. $\mathrm{SEM}(\{\textsc{Value}, \textit{light}\}) = \ldots$

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

8 Conclusion

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

Before concluding, though, I would like to discuss some possible implications of some of my proposals—specifically, the introduction of higher-order functions. My proposal makes crucial use of the higher-order function `map`, and this suggests an obvious minimalist criticism—namely that I have introduced unnecessary complexity to the grammar. Put concisely: If adding Pair-Merge to the grammar is illegitimate, then why isn't the addition of `map`? I will propose and discuss two possible answers to this challenge. First, I will discuss the possibility that higher-order functions like `map` are derivable from MERGE—that they "come for free". Second, I will discuss the possibility that it is these higher-order functions, rather than MERGE, which are the fundamental basis of language.

The idea that one could derive higher-order functions from MERGE begins with the suggestion—made frequently by Chomsky[12]—that internal MERGE is sufficient to explain the human faculty of arithmetic. The reasoning is as follows: The simplest case of Merge is vacuous internal Merge ($\mathrm{Merge}(x) \rightarrow \{x\}$), which is identical to the set-theoretic definition of the successor function ($S(n) = n+1$). Since the arithmetic is reducible to a notion of 0 or 1, the successor function and a few other axioms, Merge suffices to generate arithmetic. The process of learning arithmetic, then, is merely the process of setting the axioms of the system.

This result should not be surprising, though, since theoretical models of computation are closely linked to arithmetic. In fact, early models of computation were largely models of arithmetic—where the set of determinable functions that could be represented in model X is the set of X-computable functions on the natural numbers. An assumption generally made, called the Church–Turing thesis, is that a general class of computable functions is identical to the class of functions computable by a Turing machine. So, if we assume that a Merge-based computation system is capable of general computation, then it should be capable of performing every computable function. Since higher-order functions are computable functions, then a Merge-based system should allow for them.

This reasoning hinges on a few hypotheses, but even if it could be done completely deductively, it would still face the serious problem that models of computation and related systems assume a strict distinction between operations and atoms. Take, for instance, the process of deductive reasoning, which derives statements from from statements following rules of inference. In this case our operations are the rules of inference and the atoms are the statements. As Carroll (1895) famously illustrated, it is very easy to blur the lines between a rule of inference—such as *modus ponens*, given in (73)—and the logical statement in (74), but doing so renders the system useless.

(73) $$\frac{P \rightarrow Q, P}{Q}$$

---

[12] See Chomsky (2019, 274) for an instance in writing.

(74) $((P \to Q)\&P) \to Q$

The former is a rule of inference that may or may not be active in a logical system, while the latter is a statement which may or may not be true in a logical system. If a system doesn't explicitly include (73) but can effectively perform it, we can say that the system in question can *simulate* (73). IF a system can prove (74) without it being an axiom, then we can say that the system *generates* (74).

In the grammatical system that I have been assuming, MERGE corresponds to the rules of inference, and the syntactic objects and workspaces correspond to the atoms. In my reasoning above, I concluded that a MERGE-based system could simulate higher-order functions like `map`, but it cannot be concluded from this that `map` could be an integral part of adjunction. The human mind is capable of simulating wide variety of systems. For instance, a skilled Python programmer is effectively able to simulate a Python interpreter, but such a simulation requires learning, practice and considerable mental effort. Adjunction, on the other hand, seems to be fully innate and mostly effortless.

The second possibility is to propose that higher-order functions, or some principle that allows for them, are the basis for language. That is, we accept the minimalist evolutionary proposal that a single mutation separates us from our non-linguistics ancestors, but we propose that instead of MERGE/Merge, the result of that mutation was higher-order functions. There are a number of issues of varying levels surmountability with this proposal which I discuss below.

The first issue is that, while Merge/MERGE is a single operation and, therefore, easily mappable to a single genetic change, higher-order functions are a class of functions, making the task of linking them to a single mutation non-trivial. However, if they do form a (natural) class of functions, then they must share some singular feature, which can be mapped to a single mutation. The definition of a higher-order function as one that takes or gives a function as an input or output, respectively, suggests a such a feature—abstraction.

If abstraction is to be the defining feature of the faculty of language, then it behooves us to give a concrete definition of it. In the mathematico-computational sense, abstraction can be seen as the ability of system to treat functions as data. Applied to our cognitive system, this seems to allow meta-thinking—thinking about thinking, reasoning about reasoning, reflecting upon reflections, and so on, what Hofstadter (1979) calls "jumping out of the system." This kind of meta-thinking, though, is commonly associated with consciousness, which leads to two problems with this approach. The first problem is the hard problem of consciousness—if abstraction and consciousness are the same, then we may never fully understand either. The second problem is more mundane—We are no more conscious of adjunction than we are of MERGE, yet my reasoning here suggests that perhaps we should be conscious of the former.

There is however, a third possibility—a synthesis of the two previous possibilities. The early results of computability theory (Gödel 1931; Turing 1936) made crucial use of abstraction—using, say, number theory to reason about the axioms and operations of number theory. In fact, every simple model of computation allows for abstraction of the sort I am considering here.[13] This seems to suggest that the choice between the two possibilities above is a false one—that MERGE and abstraction cannot truly be disentangled. This does not allow us to avoid the problems that I have raised, though, but it does suggest that they can be combined and perhaps be solved together.

References

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

Carroll, Lewis. 1895. "What the Tortoise Said to Achilles." *Mind* 4 (14). JSTOR: 278–80.

Chomsky, Noam. 1957. *Syntactic Structures*. The Hague/Paris: Mouton.

———. 1965. *Aspects of the Theory of Syntax*. Cambridge: MIT Press.

---

[13] The abstraction feature of simple models of computation seems to allow self-reference, which inevitably leads to paradoxes. Such paradoxes are eliminated by complicating the models with type systems or arbitrary restrictions on abstraction.

———. 2019. “Some Puzzling Foundational Issues: The Reading Program.” *Catalan Journal of Linguistics*, 263–85.

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

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

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

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

Gödel, Kurt. 1931. “Über Formal Unentscheidbare Sätze Der Principia Mathematica Und Verwandter Systeme I.” *Monatshefte Für Mathematik Und Physik* 38 (1). Springer: 173–98.

Hofstadter, Douglas R. 1979. *Gödel, Escher, Bach: An Eternal Golden Braid*. Vol. 13. New York: Basic Books.

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

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

Turing, Alan Mathison. 1936. “On Computable Numbers, with an Application to the Entscheidungsproblem.” *J. Of Math* 58 (345-363): 5.