Final version to appear in *Structured Sentences and Computational Theory of Mind* edited by Maša Bešlin; Katherine Howitt; Alexandra Krauska; Luisa Seguin; Juan Uriagereka (Cambridge University Press)

The Cycle in Syntax in Context: rereading the past to understand the future

Robert Freidin
Princeton University

1. From the beginning

The cyclic application of rules in generative grammar predates the definition of *syntax* in *Syntactic Structures* (1957), which in a single sentence encapsulates the generative revolution. The first sentence of the first chapter defines *syntax* as "the study of the principles and processes by which sentences are constructed in particular languages." The definition is a striking departure from the usual definitions that preceded it, or in fact that have followed. Consider for example, the definition in *LSLT* (1955): "syntax is the study of linguistic form" (p. 59). The shift in focus from *linguistic form* to the *principles and processes* that construct linguistic forms captures the essence of the generative revolution that began in the 1950s and continues today. Moreover, it automatically connects generative linguistics with the physical sciences, which are concerned with understanding the underlying principles and processes that animate the natural world. From this single sentence the generative enterprise unfolds.[1]

The processes that animate human language consist of the computational mechanisms that construct structured linguistic expressions (e.g. "sentences") from the elements in the lexicon of a language. The operation of these mechanisms is determined first by their formulation and also by general principles that restrict their application (conditions on derivations) or the structured expressions they produce (conditions on representations). From the beginning, computational operations have been designated as *rules* of a formal grammar.

The cyclic application of grammatical rules was first proposed in Chomsky, Halle, Lukoff (1956) for the processes that determine the assignment of stress to English words and phrases, the first two parts of which state (p. 75):

Rule 4: Given a phonemic clause,

> *(i) assign the value 1 to all accented vowels;*
> *(ii) then apply each rule pertaining to accented vowels no more than once to each constituent, applying a rule to a constituent of order n only after having applied it to all constituents of order n + 1; i.e. beginning with the smallest constituents and proceeding to larger and larger constituents …*

The essence of the cycle is stated in (ii) whereby rules of grammar apply to constituent structure beginning with the smallest constituents, which would be lexical elements, and proceeding to successively larger constituents composed of multiple lexical elements

I would like to take this opportunity to acknowledge my conversations and collaboration with Howard Lasnik over almost half a century, which have immeasurably enriched my life in linguistics. My thanks to Howard, an anonymous reviewer, and the editors of this volume for suggestions that have significantly improved the original manuscript.

(i.e. phrases)—what is in essence bottom-up processing of phrase structure (in contrast to the top-down generation of phrase structure by phrase structure rewrite rules). In this initial formulation, the cycle is neither named nor identified as a general principle governing rule application. Moreover, it is stated as part of language-specific rule for English phonology.

The term *cycle* first appears in Halle and Chomsky (1960; p. 276) in the following context (with the terminology highlighted in boldface).

> *This stress assignment on the morphemes, however, does not remain fixed; it may be modified by the constituent structure of the utterance in which the morpheme is found, where, again, morphological and phonetic factors play a role. … The modifications are introduced in a stepwise fashion, successive steps reflecting the influence of successively higher constituents. Note also that the same modifications apply to all constituents regardless of their place in the constituent hierarchy; the same rules are reapplied to each constituent in* ***a repeating cycle*** *until the highest constituent is reached. The final result of such a* ***cyclical reapplication*** *of the same rules reflects to a certain extent the stress distribution of the morphemes as parts of lower constituents.*

This formulation differs from Chomsky, Halle, and Lukoff (1956) in explicitly referring to the reapplication of the same rules. But neither formulation of the cycle is identified as a general principle of grammar until Chomsky and Halle's *Sound Pattern of English* (1968—henceforth *SPE*)[2] in chapter 2, titled *A Sketch of English Phonology and Phonological Theory*, on page 15:

> *It is well known that English has complex prosodic contours involving many levels of stress and pitch*[1] *and intricate processes of vowel reduction. It is clear even from a superficial examination that these contours are determined in some manner by the surface structure of the utterance. Furthermore, it is natural to suppose that in general the phonetic shape of a complex unit (a phrase) will be determined by the inherent properties of its parts and the manner in which these parts are combined, and that similar rules will apply to units of different levels of complexity. These observations suggest a general principle for the application of rules of the phonological component, namely, what we shall call the principle of the "transformational cycle".*[2]

The second footnote references Chomsky, Halle, and Lukoff (1956). *SPE* makes the further clarification:

> *Observe that no rules at all are needed beyond those required for the most elementary phrases. The interplay of these rules in more complex phrases is determined by the principle of the transformational cycle, which is, it should be noted, not a rule of English grammar but rather a general principle governing the applicability of phonological rules in any grammar.* (p.23)

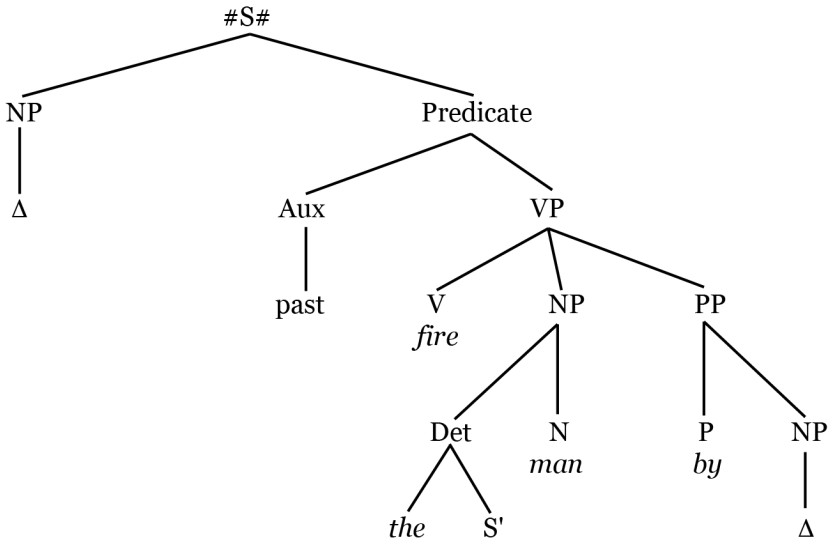
Note that this formulation focuses on phonological rules and does not consider whether this principle generalizes to syntactic rules. See Freidin (2017) for discussion.

The principle of the transformational cycle is extended to syntactic rules in Chomsky (1965) as part of a reformulation that simplifies the previous syntactic theory of *LSLT*. The extension and resulting simplification is demonstrated using the derivation of (4).

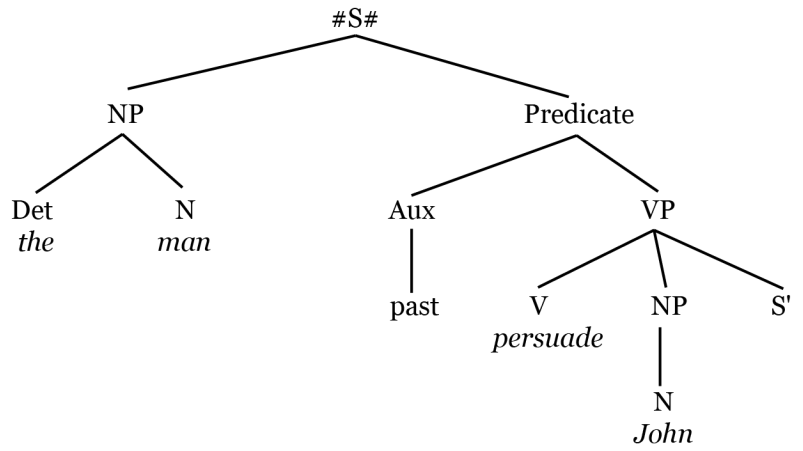
(1) The man who persuaded John to be examined by a specialist was fired.

(1) contains two embedded clausal structures: 1) the relative clause *who persuaded John to be examined by a specialist* modifying the *man* and 2) the infinitival clause *to be examined by a specialist*, where *John*, the object of *persuade*, is interpreted as the subject of the clause. In the syntactic theory of *LSLT*, clausal embedding results from the application of a transformation which combines two independent phrase-markers constructed by the phrase structure rules. Such operations were designated as *generalized transformations*, in contrast to a transformation that applied to a single phrase-marker, a *singulary transformation*. The derivation of (1) would therefore involve three separate base phrase markers (2-4).[3]

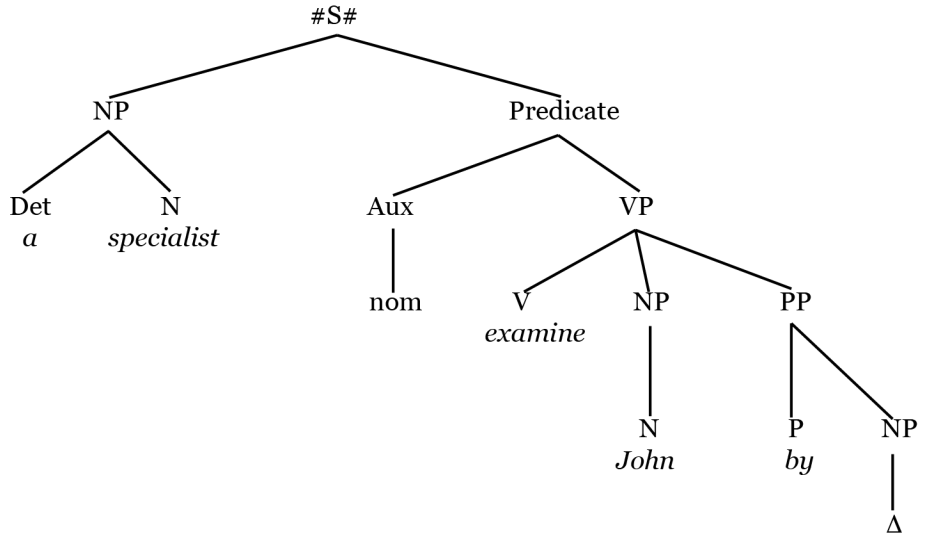
(2)



(3)

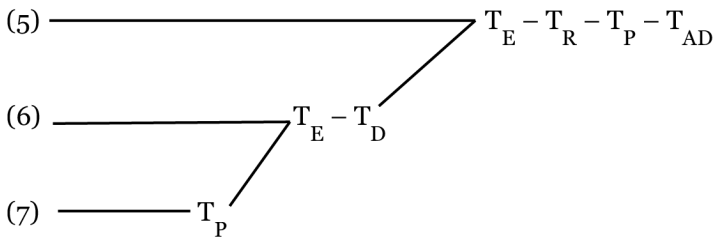


(4)



How (1) is derived from (2-4) is informally represented in the diagram (5), which represents (incompletely) the transformational history of (1).

(5)



(5) constitutes a *T-marker* for (1), which specifies the order in which transformations apply in a derivation, identified in *LSLT* as "a new level of syntactic analysis" (p. 306) and therefore on a par with the three phrase markers (P-markers) in (2-4).

As represented in (5), the transformational history of (1) begins with the

application of the passive transformation (T_P) to the phrase-marker (4). This involves the substitution of the NP *a specialist* for the empty NP dominating Δ with, postposing the agentive subject into the passive *by*-phrase. This operation presumably leaves behind an empty NP which allows the substitution of the object NP *John* into the vacated subject position. Both operations are governed by a non-distinctness condition where the NP replaced is non-distinct from the NP that replaces it.[4] The next step in the derivation involves the embedding of the transformed (4) into the phrase marker (3). This too would result from a substitution operation that substitutes the transformed (4), designated as #S#, for the S' in (3).[5] But this requires that the terminal S' is non-distinct from the nonterminal #S# in the same way that the $[_{NP}\ \Delta\ ]$ in (4) is non-distinct from $[_{NP}$ *a specialist*].

In effect, the non-distinctness condition on substitution operations requires that the position of embedded clauses be generated by phrase structure rules. Given this, to maintain the generalized transformation account of clausal embedding would require an ad hoc stipulation that phrase structure rules cannot reapply to any subordinate S in the construction of a phrase-marker. Alternatively, both generalized transformations and the T-markers they require can be eliminated, thereby simplifying syntactic theory.

The elimination of generalized transformations addresses an apparent weakness in the previous theory which permitted ordering among these transformations for which there was no empirical evidence. The theory of generalized transformations also allowed for the unsubstantiated possibility of a singulary transformation that must apply to a matrix clause before a subordinate clause could be embedded in it.[6] Without generalized transformations, the first weakness doesn't arise. But the second remains a possibility with respect to the interaction of singulary transformations (see Lasnik (2006) for discussion). In fact, the second possibility could apply in the derivation of (1) if the passive transformation applied first to the main clause (2) and then to the subordinate clause (4). The result would be the same under either ordering. However, T_P must apply to (4) so that the deletion transformation T_D can erase the derived subject *John* in (4) once (4) has been embedded in (3) by the generalized transformation T_E.[7]

What rules out the possibility of the necessary application of a singulary transformation to a matrix clause before another singulary transformation applies to a subordinate clause contained in it is the cyclic application of transformations. This is touched on in chapter 3 of *Aspects* in the following:

> *In addition to the rules of the base, so modified, the grammar contains a linear sequence of singulary transformations. These apply to generalized Phrase-markers cyclically, in the following manner. First, the sequence of transformational rules applies to the most deeply embedded base Phrase-marker. …*
>
> *That is, singulary transformations are applied to constituent sentences before they are embedded, and to matrix sentences after embedding has taken place. The embedding itself is now provided by the branching rules of the base rather*

> *than by generalized transformations. We have, in effect, converted the specific properties of the Transformation-marker (5)* [see (5) above—RF] *into general properties of any possible transformational derivation.*

These properties follow from a general principle of the computational system, the cycle—in effect, the same principle that governs the application of the phonological rules that assign stress.

3. Refining the cyclic principle

Chomsky (1973) presents a more restrictive formulation of the syntactic cycle and examines its consequences.

> *To further sharpen the notion "transformational cycle," suppose that we impose the general condition (51):*[21]
>
> *(51) No rule can apply to a domain dominated by a cyclic node A in such a way as to affect solely a proper subdomain of A dominated by a node B which is also a cyclic node.*
>
> *In other words, rules cannot in effect return to earlier stages of the cycle after the derivation has moved to larger, more inclusive domains. We will refer to (51) as the "Strict Cycle Condition."*

One consequence of the Strict Cycle Condition (henceforth SCC) for the derivation of (1), taking #S# as a cyclic node, is that the passive transformation on (4) must apply to that #S# and cannot apply to that #S# when the next higher #S# (in (3)) is reached. The SCC rules out the possibility of applying the passive transformation to the matrix clause (2) before applying the same transformation to the subordinate clause in (4), and also delaying the application of the deletion operation until the matrix clause is reached—even though with either or both the outcome would be the same.

The SCC by itself enforces the cyclic application of transformations and therefore replaces what had been identified as the principle of the transformational cycle in Chomsky and Halle (1968). It actually subsumes the notion of cycle. As stated in Lasnik and Kupin (1976), *what we suggest is that the "strict cycle condition" is not merely part of the cyclic principle, but rather that it exhausts that principle* (p. 188). See also Freidin (1978) for discussion.

Unlike the earlier discussions of the cycle, which were concerned with the derivation of grammatical sentences (e.g. (4)), the SCC is motivated as a general principle that prevents the misgeneration of deviant utterances which could result from the free application of grammatical transformations. Whether misgeneration is possible depends on the formulation of *the processes by which sentences are constructed in particular languages*, a major focus of the generative enterprise. In the early version of transformational grammar, the formulation of transformational operations is both

complex and somewhat opaque, with the result that misgeneration was not a major concern.

Consider for example, the statement of the passive transformation in Chomsky (1957).

Passive — optional:

Structural analysis: *NP – Aux – V – NP*

Structural change: $X_1 - X_2 - X_3 - X_4 \rightarrow X_4 - X_2 + be + en - X_3 - by + X_1$

This formulation performs several distinct operations. In addition to permuting the subject and object NPs, it inserts both the passive auxiliary *be + en*, adjoining it to *Aux* and the passive *by*, adjoining it to the permuted subject NP.[8] What isn't clear from the formulation is that while the object NP in the structural analysis comes to occupy the vacated subject position, the postposed subject NP is not simply switching positions with the object NP in that structural analysis. This becomes clear in the derivation of the passive configurations in (1) above. Moreover, as noted above, the derivation discussed in *Aspects* demonstrates that the elementary transformational operation involved is substitution, which is governed by a general non-distinctness condition. And in Chomsky (1970), what was called "the passive transformation" is split into two parts, one involving the postposing of the subject into a pre-existing passive *by*-phrase and the other involving to preposing of the object NP into the vacated subject position. Furthermore, these operations generalize to nominalizations (e.g. *the proposal's unanimous approval by the committee*), so that the structural analysis for NP preposing would be formulated with a variable X, replacing the string *Aux – V*, thereby extending its application to nominals.[9] In effect, the operation reduces to the elementary operation of substitution applied to an NP in a phrase marker, so substitute NP—or Move NP as it was formulated in Chomsky (1976), an analysis which is extended to *wh*-phrases in questions and relative clauses as Move *wh*-phrase, and then generalized in Chomsky (1980) as Move-α (that is, substitute α for β). These reformulations, while reducing movement transformations to a single elementary operation, no longer stipulate the positions of the the two syntactic objects involved, creating the possibility for generating deviant structures.

Misgeneration with *wh*-movement that would be prohibited under the SCC applies to one derivation of (6).

(6) *Who does John know what saw?

Assuming that *who* in (6) is interpreted as the subject of *saw* and *what* is interpreted as the object of *saw*, the underlying structure of (6) would be (7).[10]

(7) $[_{CP_1}$ e $[$ C_1 $[_{IP_1}$ John knows $[_{CP_2}$ e $[$ C_2 $[_{IP_2}$ who_i saw $what_j$]]]]]]

The first step in the derivation of (6) would substitute *who* for the empty category *e* in CP_2, leaving behind a coindexed empty category t_i, a *trace*.

(8) $[_{CP_1}$ e $[\ C_1\ [_{IP_1}$ John knows $[_{CP_2}$ who_i $[\ C_2\ [_{IP_2}\ t_i$ saw $what_j$]]]]]]

The application of Move *wh*-phrase inside CP establishes CP as a cyclic node (or domain) for *wh*-movement. The next step in the derivation of (6) substitutes *who* in (8) for the empty category *e* in CP_1, leaving behind a second trace of *who*.

(9) $[_{CP_1}$ who_i $[\ C_1\ [_{IP_1}$ John knows $[_{CP_2}$ t_i $[\ C_2\ [_{IP_2}\ t_i$ saw $what_j$]]]]]]

The final step in the derivation of (6) substitutes *what* for the empty category t_i in CP_2, leaving a trace of *what* in IP_2.

(10) $[_{CP_1}$ who_i $[\ C_1\ [_{IP_1}$ John knows $[_{CP_2}$ $what_j$ $[\ C_2\ [_{IP_2}\ t_i$ saw t_j]]]]]]

This last application of Move *wh*-phrase occurs within a cyclic subdomain of the current cycle CP_1 in violation of the SCC.

However the SCC alone is not sufficient to block the derivation of (10) from (7), and thereby the misgeneration of the deviant (6). Another derivational route from (7) to (10) would involve moving *what* to the CP_2 position it occupies in (10) as a first step, and then moving *who* to the position in CP_1 position that it occupies in (10). At no point in this derivation does Move *wh*-phrase apply within a cyclic subdomain of the current cycle. The derivation is ruled out by the Subjacency Condition (see Chomsky (1973, 1977)), a locality condition which prohibits a single movement operation from crossing more than one "bounding domain"—where IP constitutes such a domain. The movement of *who* from IP_2 to CP_1 crosses both IP_2 and IP_1 in violation of the condition.

When Subjacency is interpreted as a constraint on derivational operations, then we require the SCC as well to account for the deviance of (6). The alternative is to consider Subjacency as a constraint on trace binding, a condition on representations, in which case Subjacency subsumes the empirical effects of the SCC for Move *wh*-phrase. This analysis extends to the empirical effects of Move NP by marshalling additional general constraints on representations that are independently motivated like the Subjacency Condition; see Freidin (1978) for details. In this way, cyclicity follows from the formulation of principles without any explicit reference to cyclic domains, in contrast to the SCC.[11]

More generally, there is an interplay between the principles and processes of the computational system for human language (CS_{HL}). Principles are formulated on the basis of the processes assumed, including the linguistic representations they construct. When the SCC was first formulated in Chomsky (1973), the system of computational operations was organized into two separate components: 1) a *base* component

consisting of a set of rewrite rules (e.g. S → NP VP and VP → V NP, as cited in *Aspects*) plus a substitution operation that inserted lexical items into the phrase markers created by the rewrite rules, and 2) a transformational component consisting of a sequence of transformations[12] that maps these phrase markers onto the syntactic representations of the sentences generated. Within the decade that followed, the sequence of transformations is reduced to essentially a single elementary operation of substitution (Move-α) and the existence of rewrite rules as a part of CS_{HL} is called into question. Consider the statement on page 136 of *Lectures on Government and Binding* (1981):

> *Base rules are virtually eliminated*.[13]

So when the minimalist program is first postulated over a decade later, the processes of CS_{HL} and their organization within it have undergone a major reformulation from what they were when the SCC was first proposed.

4. A cyclic principle for a minimalist program

Fast forwarding to Chomsky (1995a), where set-forming Merge is proposed as the structure-building operation that replaces phrase structure rewrite rules, derivations now proceed bottom-up, from lexical items to the phrases that contain them. So, that property of cyclic derivation follows from the formulation of Merge. The question remains whether Merge can apply in a way that would violate "strict cyclicity"—for example, by merging an independent syntactic object with a subpart of an existing phrase.

Initially, the strict cyclicity of Merge was ensured by an independent general principle, the Extension Condition (EC) of Chomsky (1992).[14] The EC extends to "movement operations", yielding displacement, which in Chomsky (2004) are recognized as another mode of the application of Merge, where a constituent of a syntactic object merges with that object thereby creating a new construction. Thus Merge is a single structure-building operation with two modes of application: External Merge (where the two objects merged are separate before the merger) and Internal Merge (where one of the two objects merged is a constituent of the other). The one operation incorporates the effects of both phrase structure rules and movement transformations. One result is that there is no sequence of transformations that applies to successively larger domains, as in *Aspects*. Another is that Merge eliminates substitution as an elementary structure-preserving transformational operation.

The structure-preserving property of substitution is nonetheless incorporated into the Merge-centric theory as the No-Tampering Condition (NTC) of Chomsky (2005), which connects the NTC with cyclicity (as indicated by the addition of boldface to the quotation below).

> *One natural property of efficient computation, with a claim to extralinguistic generality, is that operations forming complex expressions should consist of no more than a rearrangement of the objects to which they apply, not modifying them internally by deletion or insertion of new elements. If tenable, that sharply*

> *reduces computational load: what has once been constructed can be "forgotten" in later computations, in that it will no longer be changed.* ***That is one of the basic intuitions behind the notion of cyclic computation.*** *The EST/Y-model and other approaches violate this condition extensively, resorting to bar levels, traces, indices, and other devices, which both modify given objects and add new elements. A second question, then, is whether all of this technology is eliminable, and the empirical facts susceptible to principled explanation in accord with the "no-tampering" condition of efficient computation.* (p. 11)

From this perspective, cyclicity becomes a consequence of a *natural property of efficient computation*, which renders *the empirical facts susceptible to principled explanation*. Unlike the SCC or the EC that replaces it, the NTC governs both linguistic and nonlinguistic computation, and therefore would not be stipulated as a unique principle of CS_{HL}.[15]

Given the extralinguistic status of the NTC, there remains a question of how it is instantiated in the principles and processes that constitute CS_{HL}. If the processes by themselves allow for violations of the NTC, then one possibility is that the operation of these processes or their structural output would be restricted by one or more principles of CS_{HL} that conform to the NTC. Alternatively, the formulation of processes must by itself prevent such violations. In effect, the formulation of an operation would embody the NTC, as will be demonstrated in the next section.

§4.1. The cyclicity of Merge in context

In its most recent formulations (see Chomsky (2020) and Chomsky et. al. (2023)), Merge is an operation that functions within a workspace WS, the computational space in which a linguistic expression is generated. The initial state of WS for any linguistic expression will contain inscriptions of the lexical items that compose the expression being constructed.[16] Consider for example the simple sentence in (11)

(11) *The manuscript should be revised.*

The initial state of WS (given in square brackets) for this single clause would be roughly (12).[17]

(12) WS = [*should*, *be*, *revised*, *the*, *manuscript*]

The operation Merge combines two syntactic elements α and β in a binary set {α, β} that represents their compositional structure. Thus one function of Merge is binary set formation, where [α, β] → [{α, β}]. The operation also changes WS, mapping it onto WS', where (12) would map to (13).

(13) WS' = [*should*, *be*, *revised*, {*the*, *manuscript*}]

In this way, Merge also functions as a mapping between workspaces. While a set constitutes a syntactic object, a WS (in square brackets) is not a set and therefore not itself a syntactic object. In (12), WS is a collection of five lexical inscriptions, the *members* of the workspace, whereas WS' in (13) contains four members (three lexical inscriptions and a set containing two others). In the mapping between (12) and (13), (external) Merge reduces the number of members of WS by one. In (13), *the* and *manuscript* become *terms* in WS'. Three successive applications of external Merge to (13) yield (14) with a single member.

(14) WS" = [{*should*, {*be*, {*revised*, {*the*, *manuscript*} } } }]

While it is obvious how each application of external Merge in the derivation yielding (14) satisfies the no-tampering and extension conditions, it remains to be demonstrated exactly how this follows from the formulation of external Merge.

Merge, in its simplest formulation, would be binary set formation as (15).

(15) Merge (α, β) ⇒ {α, β}

With external Merge, α and β are members of WS. Thus external Merge targets two members of WS, where by definition neither contains the other, and forms a binary set. In the derivation of *the manuscript should be revised*, we reach a stage where WS will be (16) with two members.

(16) [*should*, { *be* { *revised*, {*the*, *manuscript*} } } }]

If Merge targets the two members of (16), the result is (14). However, if Merge can target *should* (a member of WS) and *be* (a term in another member of WS) the result would be (17), where presumably the two inscriptions of *be* are copies.

(17) [{*should*, *be*}, { *be* { *revised*, {*the*, *manuscript*} } }]

(17) does not violate the NTC, but would if *be* were to disappear from the second member. Whether (17) violates the EC is open to interpretation. One of the members is extended, but the other member involved is not. The operation has the effect of external Merge, where neither target is contained in the other, and not the standard effect of internal Merge where a term inside a member is merged with that member. This would be impossible if there were a binary target restriction (BTR) on Merge where it can only target either two members of WS (external Merge) or one member and a term contained in that member (internal Merge).[18] (17) would require targeting three elements, two members and a term contained in one of them.

Note that BTR[19] also prohibits merger where the two elements are both terms—whether or not they are terms of a different member or the same member. In order to merge two terms of WS would require targeting the member that contains each of them as well, violating the binary restriction. The BTR thus prohibits the extension of Merge as Parallel Merge, Late Merge, or sidewards movement.[20] In effect, Merge is a binary

operation with respect to its output and also its input as defined by the WS in which and to which it applies.

In the case of internal Merge, one element is a member of WS and the second is a term contained in that member. For example, internal Merge maps (14) onto (18), where the operation targets the single member of (14) and the term {*the*, *manuscript*} contained in it.

(18) WS^ = [{{*the*, *manuscript*}, {*should*, {*be*, {*revised*, {*the*, *manuscript*} } } } }]

This application constitutes another instance of binary Merge; but unlike external Merge, internal Merge does not affect the number of members in the workspace. Rather, it assigns a second context to a term contained in a member, which is allowed under the formulation of Merge that incorporates the BTR. Thus the NTC on composition follows from the formulation of Merge for both external and internal Merge.[21]

It should be clear at this point that Merge does not apply in a vacuum. Rather it interacts with content and structure of the workspace in which and to which it applies. And given the formulation of Merge discussed above, the concept of a workspace is essential to the formulation of the operation that manifests *the Basic Property of language: that a language is a finite computational system yielding an infinity of expressions, each of which has a definite interpretation in semantic-pragmatic and sensorimotor systems (informally, thought and sound)* [Berwick and Chomsky (2016, p. 1)].

The BTR constraint on Merge has an interesting consequence for the analysis of passives. It prohibits the construction of the passive *by*-phrase via internal Merge (cf. the derivation for (1) proposed in *Aspects*), where the object of *by* has been displaced from another structural position. Under such analyses, both the passive *by* and its object would be independent terms of a single member in WS before they are merged, and therefore could not be merged under the BTR. The BTR would also prohibit a derivation in which the passive *by* as a member of WS merges with its object when that object is a term contained in another member of WS. In effect, the *by*-phrase can only be legitimately constructed by external Merge when both *by* and its object are members of WS. The analysis generalizes to the derivation of nominalizations (e.g. *the proposal's unanimous approval by the committee* and *the unanimous approval of the proposal by the committee*). Thus the assignment of the external argument function in passives must result from the transmission of that function from the passive participle or the derived nominalization via the passive *by* to its object.[22]

The analysis of the passive *by*-phrase illustrates how the BTR formulation of Merge might constitute the simplest formulation of the operation that creates the compositional structure of human language from the lexicon of a language. As such, Merge provides a basis for a strong minimalist thesis about CS_{HL} as discussed most recently in Chomsky et. al. (2023, p. 12):

> *A guiding principle in this quest for explanation is the Strong Minimalist Thesis (SMT). Over the years, SMT has taken a number of forms,*[27] *but in the present context we understand it in two ways. First, conformity to SMT requires that the structures of I-language are generated by the simplest operations. Hence, Merge must take the simplest possible form, and the number of additional structure-building operations should be minimized (ideally, entirely eliminated). As much as possible, then, the form and the function of necessary operations like Merge are reduced to third-factor principles such as computational efficiency, understood in this context as natural law.*[28]

This discussion identifies two foci of the SMT (simplest operations and computational efficiency) and links them by suggesting that the form and function of these operations might reduce to general principles of computational efficiency.[23] Exactly how such reductions would work is unclear. Moreover, there is perhaps a better way to link these two foci of the SMT whereby the simplest operations of CS_{HL} embody general principles of computational efficiency. The BTR formulation of Merge is one example.

Consider again the cyclicity of Merge, as discussed above. The BTR subsumes the empirical effects of the EC for Merge, the main (perhaps sole) operation that creates compositional structure. Moreover, the BTR subsumes the empirical effects of imposing an extralinguistic NTC as general principle of efficient computation on Merge. As a result, the BTR ensures the strict cyclicity of Merge and thus replaces an independent cyclic principle that governs its operation. The result is a shift in focus from principles that constrain operations to the operations themselves, a shift from principles to the formulation of processes that embody general principles of efficient computation. To the extent that computational operations embody principles of efficient computation we have a stronger version of the SMT.

As illustrated above, the history of cyclicity in generative grammar provides a context for understanding the interaction of principles and processes that constitute the core of the generative enterprise and how this has evolved to our current view of CS_{HL}. This exploration has led to a new and potentially stronger interpretation of the SMT, which attempts to identify an explanatory core for linguistic theory.

References:

Berwick, Robert and Noam Chomsky. 2016. *Why Only Us?: language and evolution*. Cambridge, MA: MIT Press.

Chomsky, Noam. 1955. *The Logical Structure of Linguistic Theory*. Ms., Harvard University, Cambridge, MA. Revised version published in part by Plenum, New York, 1975.

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

Chomsky, Noam. 1964 *Current Issues in Linguistic Theory*. The Hague: Mouton.

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press.

Chomsky, Noam. 1970. Remarks on nominalization. In Roderick A. Jacobs and Peter S. Rosenbaum (eds), *Readings in English Transformational Grammar*. Waltham, MA: Ginn, 184–221.

Chomsky, Noam. 1973. Conditions on transformations. In Stephen R. Anderson and Paul Kiparsky (eds.) *A Festschrift for Morris Halle*. New York: Holt, Rinehart and Winston. 232-286.

Chomsky, Noam. 1976. Conditions on rules of grammar. *Linguistic Analysis* 2: 303–51.

Chomsky, Noam. 1977. On wh-movement. In Peter Culicover, Thomas Wasow and Adrian Akmajian (eds.), *Formal Syntax*. New York: Academic Press, 71-132.

Chomsky, Noam. 1980. On binding. *Linguistic Inquiry* 11: 1-46.

Chomsky, Noam. 1986. *Knowledge of Language: Its nature, origin, and use*. New York: Praeger.

Chomsky, Noam. 1993. A minimalist program for linguistic theory. In Kenneth Hale and Samuel J. Keyser (eds.), *The View from Building 20: Essays in Linguistics in Honor of Sylvain Bromberger*. Cambridge, MA: MIT Press, 1–52. [Reprinted in Chomsky (1995b).]

Chomsky, Noam. 1995a. Bare phrase structure. In Héctor Campos and Paula Kempchinsky (eds.), *Evolution and Revolution in Linguistic Theory: studies in honor of Carlos P. Otero*. Washington, DC: Georgetown University Press, 51–109.

Chomsky, Noam. 1995b. *The Minimalist Program*. Cambridge, MA: MIT Press.

Chomsky, Noam. 2000. Minimalist inquiries: the framework. In Roger Martin, David Michaels, and Juan Uriagereka (eds.), *Step by Step: Essays on minimalist syntax in honor of Howard Lasnik*. Cambridge, MA: MIT Press, 89–155.

Chomsky, Noam. 2001. Derivation by phase. In Michael Kenstowicz (ed.), *Ken Hale: A Life in Language*, Cambridge, MA: MIT Press, 1–52.

Chomsky, Noam. 2004. Beyond explanatory adequacy. In Adriana Belletti (ed.), *Structures and Beyond: The cartography of syntactic structures*. New York: Oxford University Press, 104–131.

Chomsky, Noam. 2005. Three factors in language design. *Linguistic Inquiry* 36:1, 1–22.

Chomsky, Noam. 2008. On phases. In Robert Freidin, Carlos Otero, and Maria Luisa Zubizarreta (eds.), *Foundational Issues in Linguistic Theory: essays in honor of Jean-Roger Vergnaud*, Cambridge, MA: MIT Press, 133–166.

Chomsky, Noam. 2020. The UCLA lectures: With an introduction by Robert Freidin. LingBuzz.(https://ling.auf.net/lingbuzz/005485)

Chomsky, Noam. 2021. Minimalism: Where are we now, and where can we hope to go. *Gengo Kenkyu* 160, 1–41.

Chomsky, Noam and Morris Halle. 1968. *The Sound Pattern of English*. New York: Harper and Row.

Chomsky, Noam, Morris Halle, and Fred Lukoff. 1956. On accent and juncture in English. In Morris Halle, H. Lunt, & H. MacLean (eds.), *For Roman Jakobson*. The Hague: Mouton, 65–80.

Chomsky, Noam and Howard Lasnik. 1993. The theory of principles and parameters. In J. Jacobs, A. von Stechow, W. Sternefeld, and T. Vennemann (eds.), *Syntax: an international handbook of contemporary research* (vol. 1, 506–569). Berlin: Walter de Gruyter. [Reprinted in Chomsky (1995b).]

Chomsky, Noam, Daniel Seely, Robert Berwick, Sandiway Fong, M.A.C. Huybregts, Hisatsugu Kitahara, Andrew McInnerney and Yushi Sugimoto. 2023. *Merge and the Strong Minimalist Thesis*. Elements in Generative Syntax. Cambridge, UK: Cambridge University Press.

Fillmore, Charles J. 1963. The position of embedding transformations in a grammar. *Word*, 19, 208–231.

Freidin, Robert. 1978. Cyclicity and the theory of grammar. *Linguistic Inquiry*, 9, 519–549.

Freidin, Robert. 1994. Conceptual shifts in the science of grammar 1951–1992. In C. P. Otero (ed.) *Noam Chomsky: Critical Assessments*, vol. 1, tome 2, 653–90. London: Routledge. Reprinted in Robert Freidin. 2007. *Generative Grammar: Theory and its History*. London: Routledge, 266–298.

Freidin, Robert. 1999. Cyclicity and minimalism. In Samuel D. Epstein & Norbert Hornstein (Eds.), *Working minimalism*. Cambridge, MA: MIT Press, 95–126.

Freidin, Robert. 2016. Chomsky's linguistics: The goals of the generative enterprise. *Language*, 92, 671–723.

Freidin, Robert. 2017. Cyclicity in syntax. In Mark Aronoff (ed.), *The Oxford Research Encyclopedia of Linguistics*, https://oxfordre.com/linguistics.

Freidin, Robert. 2020. *Syntactic Structures*: a radical appreciation. LingBuzz (https://ling.auf.net/lingbuzz/004996).

Freidin, Robert. 2021. The Strong Minimalist Thesis. In Peter Kosta (ed.) New Perspectives of Generative Grammar and Minimalism. a special edition of *Philosophies*. (https://doi.org/10.3390/philosophies6040097).

Freidin, Robert and Howard Lasnik. 1981. Disjoint reference and wh-trace. *Linguistic Inquiry* 12, 39-53.

Halle, Morris and Noam Chomsky. 1960. The morphophonemics of English. MIT RLE Quarterly Progress Report, 58, 275–281.

Landau, Idan. 2024. *Control*. Elements in Generative Syntax. Cambridge, UK: Cambridge University Press.

Lasnik, Howard. 1988. Subjects and the θ-Criterion. *Natural Language and Linguistic Theory* 6, 1-17.

Lasnik, Howard. 2006. Conceptions of the cycle. In Lisa Cheng and Norbert Corver (eds.), *Wh-movement: moving on*. 197–216. Cambridge, MA: MIT Press.

Lasnik, Howard and Joseph Kupin. 1976. A restrictive theory of transformational grammar. *Theoretical Linguistics* 4, 173-196.

Katz, Jerrold J. and Paul Postal. 1964. *An Integrated Theory of Linguistic Descriptions*. Cambridge, MA: MIT Press.

Rosenbaum, Peter. 1967. *The Grammar of English Predicate Complement Constructions*. Cambridge, MA: MIT Press.

---

[1] See Freidin (2020) for discussion. The processes in *Syntactic Structures* include phrase structure rewrite rules and transformations.

[2] The book had been written over a period of "about ten years" (p. x). However, even in Chomsky (1964), which refers to the transformational cycle in phonology, it is not explicitly identified as a principle.

[3] Although Chomsky (1965) gives a more complicated phrase marker for (3) and represents Δ as *passive*, these details do not affect the main points of this discussion.

[4] The passive transformation on this analysis would presumably insert the passive auxiliary *be–en*, accounting for the presence of the passive auxiliary and passive participle form of the verb.

[5] *Aspects* credits Katz and Postal (1964) for this analysis.

[6] Both observations are credited to Fillmore (1963).

[7] T_D is named the Identity Erasure Transformation in Rosenbaum (1967, p. 6) and is later designated at Equi-NP Deletion. For decades, this deletion analysis for the derivation of subjectless infinitival clauses has been replaced by assuming a phonetically null anaphoric subject PRO in these constructions (see Landau (2024) for discussion). But see Chomsky, et. al. (2023) for a modern reincarnation of the deletion analysis. T_R

designates the transformation that forms relative clauses; T_{AD}, the transformation that deletes the agentive *by*-phrase in passives.

[8] The first formulation of a passive transformation, on page 406 of *LSLT*, applies the permutation (designated as *an elementary operation* on page 405) prior to the two insertions of lexical material.

[9] For a more detailed discussion of the history of the passive transformation in generative grammar, see Freidin (1994).

[10] IP stands for "inflection phrase" where the finite inflection on the verb *saw* begins as a separate element that constitutes the head of IP, which is equivalent to S in *Aspects*. CP for "complementizer phrase" is headed by a phonetically null C. CP is rendered as S' in Chomsky (1973).

[11] As noted in Chomsky (1977), *the subjacency condition is a property of cyclic rules, i.e., part of the definition of the cycle* (p. 73). The deviance of (6) above links the SCC and the subjacency condition, where one derivation violates strict cyclicity but not locality as instantiated in the concept of subjacency while the other derivation violates locality but not strict cyclicity. In Chomsky (2000), the subjacency condition is replaced by the Phase Impenetrability Condition:

> *In phase α with head H, the domain of H is not accessible to operations outside α, only H and its edge are accessible to such operations.* (p. 108)

which *yields a strong form of Subjacency* (p. 108). For critical discussion see §3 of Freidin (2016).

[12] Chomsky (1973) discusses *wh*-movement, *each*-movement, Passive, and Equi-NP Deletion.

[13] See also Chomsky and Lasnik (1993—reprinted as chapter one of Chomsky (1995b)), where it is claimed that *the apparent eliminability of phrase structure rules becomes clear by the late 1960s with the separation of the lexicon from the computational system and the development of X-bar theory* (p. 25). They note further that *phrase structure rules are (largely) redundant with subcategorization, hence are (largely) eliminable* (p. 31).

[14] See Freidin (1999) for a discussion of the Extension Condition and various attempts to derive its empirical effects from independent conditions on derivations and representations (cf. Freidin (1978)).

[15] In Chomsky et. al. (2023), the NTC is characterized as "a consequence" of Preservation, characterized as *a general constraint, normal for all computation in formal systems* (p. 22, see §3.3.3 for details).

[16] This would include multiple inscriptions of the same lexical item, as in *the professor praised the students during the final lecture*.

[17] See Chomsky et. al. (2023) for a more detailed account of the derivation of passives, though the details are not crucial for this demonstration.

[18] Note that the terminology employed here distinguishes a member of a workspace from a term in the workspace. In Chomsky et. al. (2023), a member is identified as a term of WS (see p. 19).

[19] In Freidin (2021) BTR is designated as the Access Restriction.

[20] See Chomsky (2020) for further discussion the empirical problems with such extensions of Merge.

[21] Given the distinction between a member and a term in WS, it is not necessarily obvious that external Merge (which targets two members of WS) and internal Merge (which targets one member and a term contained in that member) are exactly the same operation such that the existence of the former entails the existence of the latter (or perhaps less plausibly, the converse)—contra Chomsky (2020) and elsewhere. In both cases, the first target must always be a member of WS. But targeting a term as the second step extends the range of potential targets, especially considering the multiple terms a single member of WS may contain. And if targeting a member in the first step involves distinguishing the members of the WS, then targeting a term in the second would be a complication for the search process.

[22] See Lasnik (1988) for discussion.

[23] It doesn't mention a third focus dating back to the advent of the minimalist program in Chomsky (1993), legibility conditions imposed at the interface between language and other cognitive systems. See Freidin (2021) for discussion of problems with formulating such conditions.