Article

The Strong Minimalist Thesis

Robert Freidin

Program in Linguistics, Princeton University, Princeton, NJ 08544, USA; freidin@princeton.edu

Abstract: This article reviews and attempts to evaluate the various proposals for a strong minimalist thesis that have been at the core of the minimalist program for linguistic theory since its inception almost three decades ago. These proposals have involved legibility conditions for the interface between language and the cognitive systems that access it, the simplest computational operation Merge (its form and function), and principles of computational efficiency (including inclusiveness, no-tampering, cyclic computation, and the deletion of copies). This evaluation attempts to demonstrate that reliance on interface conditions encounters serious long-standing problems for the analysis of language. It also suggests that the precise formulation of Merge may, in fact, subsume the effects of those principles of efficient computation currently under investigation and might possibly render unnecessary proposals for additional structure building operations (e.g., Pair-Merge and FormSequence).

Keywords: strong minimalist thesis; interface condition; Merge; workspace; efficient computation; inclusiveness; no tampering; strict cyclicity; copy deletion

Citation: Freidin, R. The Strong Minimalist Thesis. *Philosophies* **2021**, *6*, 97. https://doi.org/10.3390/philosophies6040097

Academic Editor: Peter Kosta

Received: 8 October 2021
Accepted: 22 November 2021
Published: 29 November 2021

1. The Earliest Formulation of a Strong Minimalist Thesis

Language is an optimal solution to legibility conditions. So states the first formulation of the Strong Minimalist Thesis (SMT) in Chomsky's "Minimalist Inquiries" 1998 [1] (designated there as *the strongest minimalist thesis*), six years after the original proposal for "a minimalist program for linguistic theory" under that title, first published as the first of the *M.I.T. Occasional Papers in Linguistics* in 1992 [2]. In that interim, *a minimalist program* becomes *The Minimalist Program* [3][1], the title of Chomsky's 1995 collection of papers, which includes the 1992 paper followed by a final chapter that incorporates "Bare Phrase Structure" (*MITOPL* #5, 1994) [6]. Why the SMT does not appear in the 1995 collection is explained in Chomsky's introduction to the 20th anniversary edition of *The Minimalist Program* [7]: "When the first edition of *The Minimalist Program* was published, the thesis seemed too extreme to be seriously proposed."

Interestingly however, the original formulation of the Minimalist Program in the 1992 paper lists among the assumptions that define the program the following (pp. 170–171)[2]:

> *A language, in turn, determines an infinite set of linguistic expressions (SDs), each a pair (p, l) drawn from the interface levels (PF, LF), respectively.*

and

> *Conditions on representations—those of binding theory, Case theory, θ-theory, and so on—hold only at the interface, and are motivated by properties of the interface, perhaps properly understood as modes of interpretation by performance systems.*

which are followed by a final assumption:

> *The linguistic expressions are the optimal realization of the interface conditions, where "optimality" is determined by the economy conditions of UG.*

To get from this final assumption to the formulation of SMT in [1] involves some reasonably straightforward substitutions. Replacing *the linguistic expressions* with *language* is to a large extent motivated by the assumption cited above that a language "determines an infinite set of linguistic expressions" (putting aside the question of the relation between *language* and *a language*[3] plus the fact that *language* "has been studied intensively and productively for 2500 years, but with no clear answer to the question of what language is" [9] (p. 2)). Next, the terms *legibility conditions* and *interface conditions* are virtually synonymous given that the first term is actually interpreted as 'legibility conditions at the interface(s)'. This synonymy is demonstrated in the discussion in [1] of how the SMT might bear on the notion of "linguistic evidence", where the SMT is interpreted as saying that "an optimal solution to legibility conditions satisfies all other empirical tests as well"[4] (p. 97). The equivalence is expressed in the sentence that follows: "The reformulated thesis replaces the obscure notion of "linguistic evidence" by the meaningful notion: satisfaction of interface conditions." The statement also sets up an equivalence between *solution to legibility conditions* and *satisfaction of interface conditions*, where *satisfaction* in this context is again virtually synonymous with *realization*, and thus linking the first formulation of the SMT in [1] to the final assumption that defines the Minimalist Program in [2].

To understand why the 2015 introduction to the 20th anniversary edition of *The Minimalist Program* misses the fact that the SMT has apparently been a part of the minimalist program from the outset, we need to consider the context in which it was written, specifically the formulation of the SMT in 2015, which differs significantly from both that of [1] and, therefore, the virtually synonymous assumption stated in [2].

In the fifteen years that separate the initial formulation of the SMT in [1] and the introduction to the 20th anniversary edition [7] the SMT has been stated in somewhat different though related ways. Although the same formulation appeared the following year in [8], two years later "Beyond Explanatory Adequacy" [10] reframed the discussion in terms of "the initial conditions on language acquisition", which the paper categorizes as

a "unexplained elements of S_0" (the initial state of the language faculty).
b interface conditions ("the principled part of S_0").
c "general properties" (which are external to S_0, e.g., computational efficiency).

In this context, the SMT ("an extremely strong minimalist thesis") would be that there are no unexplained elements of S_0; category (a) is empty. This formulation references interface conditions, but doesn't focus on them, so it is a departure from the previous formulations, as is the inclusion of general properties. This raises several questions for interpretation, first and foremost: what are the elements of S_0? Presumably these would include the principles and processes that generate structured linguistic expressions. Next, what is the distinction between "unexplained" versus "explained" elements? The claim this formulation makes is that interface conditions fall under the latter category, and everything else that cannot be attributed to "general properties" would be unexplained. In effect, elements of S_0 motivated by either interface conditions or general properties have an explanation, and all other elements of S_0 are without explanation and, therefore, basically stipulations. So, this formulation of the SMT suggests that all elements of S_0 are either interface conditions or general properties. Thus, if this version of SMT is valid, then Merge, for example, which is not an interface condition, must somehow fall under general properties. This is not discussed in [10] but does come up in later discussions.

The next formulation of the SMT appears six years later in "Approaching UG from Below" [11] which proposes that the faculty of language FL "is "perfectly" designed" (p. 4). Given this formulation [11] immediately identifies two tasks: (1) "to formulate SMT coherently" and (2) "to determine how close it is to true." As the first task implies, this formulation needs to be spelled out in more detail to be coherent. It differs from previous formulations in that it focuses directly on the larger issue language design[5], suggesting that it may be perfect in some sense to be defined. In effect, this formulation has moved from an optimal solution to interface conditions to perfect design, which is a stronger claim since *optimal* does not entail *perfect*, while the converse entailment holds. However, on the following page, the discussion returns to interface conditions in the earliest formulations of the SMT.

> *An I-language is a computational system that generates infinitely many internal expressions, each of which can be regarded as an array of instructions to the interface systems, sensorimotor (SM) and conceptual-intentional (CI). To the extent that third factor conditions function, the language will be efficiently designed to satisfy conditions imposed at the interface—one can imagine more radical theses, to which I will briefly return. We can regard an account of some linguistic phenomena as principled insofar as it derives them by efficient computation satisfying interface conditions. We can, therefore, formulate SMT as the thesis that all phenomena of language have a principled account in this sense, that language is a perfect solution to interface conditions, the conditions it must at least partially satisfy if it is to be usable at all.*

But here we have moved from *an optimal solution* to *a perfect solution* of interface conditions.[6] Moreover, this formulation is motivated from a consideration of the function of third factor conditions (e.g., computational efficiency) in language design, thereby linking the categories (b) and (c) of [10] mentioned above and, in effect, 'approaching UG from below', which considers question "How little can be attributed to UG while still accounting for the variety of I-languages attained, relying on third factor principles?" The above quoted passage also provides an extended formulation of the SMT such that all phenomena of language have a principled account, whereby they are derived "by efficient computation satisfying interface conditions."

Under this formulation, the barring of "unexplained elements of S_0" in category (a) above follows as a consequence. In this way the SMT guarantees only principled accounts of linguistic phenomena, which would count as explanations[7] rather than mere descriptions because they rely on conditions imposed by cognitive systems external to the faculty of language in concert with general "third factor" properties imposed by the requirements for computational efficiency and the like. The consideration of third factor properties for the distinction between explanation and description significantly raises the bar for linguistic theory, as noted in [11].

> *Insofar as the third factor can be shown to be operative in the design of FL, explanation can proceed "beyond explanatory adequacy" in the technical sense, raising new questions: not only asking what mechanisms suffice to determine I-language from data available, but why these mechanisms should exist, and whether they are real or just dispensable descriptive technology.* (pp. 3–4)

Presumably the answers to these new questions will come from outside the limited domain of UG and the narrow consideration of language data that is used to motivate proposals about it (as has been the focus from the earliest work on generative grammar), where, under the SMT, the focus has shifted to questions of efficient computation satisfying interface conditions.

A year later, "On Phases" [14] returns to the earliest formulation of the SMT, thus "language is an optimal solution to interface conditions that FL must satisfy; that is, language is an optimal way to link sound and meaning, where these notions are given a technical sense in terms of the interface systems that enter into the use and interpretation of expressions generated by an I-language" (p. 135). The retreat from *perfect solution* to *optimal solution* is, perhaps, explained by the cautionary comment that follows: "If SMT held fully, which no one expects, UG would be restricted to properties imposed by interface conditions. A primary task of the MP is to clarify the notions that enter into SMT and to determine how closely the ideal can be approached." Nonetheless, five years later in "Problems of Projection" [15], while the Minimalist Program is characterized as beginning with a question that asks, "what an optimal solution would be to the conditions that must be satisfied by GPs for natural language: namely, satisfying interface conditions" (footnote omitted)[8], the SMT is again formulated as "language is a perfect solution to these conditions" (p. 38). From this perspective the research program should investigate "to what extent our best picture of actual GPs—I-languages—conforms to SMT and; where it does not, seek to show how the discrepancy can be overcome by a deeper analysis of the processes involved." To which a further caution is added about "the gap that resists such efforts includes UG" (footnote omitted)—that is, whatever is postulated as part of UG that is not motivated by interface conditions or general (third factor) properties, the unexplained elements of the initial state of the language faculty as referenced in category (a) in [10].

In Chomsky's "Problems of Projection: Extensions" [16], published the same year as the introduction to the 20th anniversary edition of *The Minimalist Program*, the SMT is presented in the following context:

> *In the best case, phenomena would be explained by interaction of the simplest computational operation Merge, with its two logically possible subcases. Internal Merge IM (automatically yielding "the copy theory of movement") and External Merge EM—interacting with general principles of minimal computation MC. The Strong Minimalist thesis SMT articulates this goal.*

Significantly, there is no mention of interface conditions, nor in the entire paper. Instead, the discussion of the SMT appears to have shifted from interface conditions to the formulation of generative procedures in concert with principles of computational efficiency, now focused on a concept of minimal computation.

It is in this context that the statement in the introduction to the 20th anniversary edition that the SMT was not proposed in the early work on the Minimalist Program because "it seemed too extreme to be seriously proposed" should be interpreted. What follows that statement is a discussion of "the strange property of displacement that is ubiquitous in natural language: phrases are understood both where they are heard and in a position that is not articulated", a property that, Chomsky notes, "had always seemed—to me in particular—a curious imperfection of language" (p. x)—and thus a fundamental contradiction to the idea of language as a perfect system involving the simplest generative procedures. The problem for stating the SMT involved the formulation of the computational system and not interface conditions.

In this introduction the SMT is explicated as follows (p. ix):

> *The basic principle of language (BP) is that each language yields an infinite array of hierarchically structured expressions, each interpreted at two interfaces, conceptual-intentional (C-I) and sensorimotor (SM)—the former yielding a "language of thought" (LOT), perhaps the only such LOT; the latter in large part modality-independent, though there are preferences. The two interfaces provide external conditions that BP must satisfy, subject to crucial qualifications mentioned below. If FL is perfect, then UG should reduce to the simplest possible computational operation satisfying the external conditions, along with principles of minimal computation (MC) that are language independent. The Strong Minimalist Thesis (SMT) proposes that FL is perfect in this sense.*

This explication is then qualified in the following words:

> *SMT is not precisely formulated. MC can be interpreted in various ways, though some of its properties are uncontroversial, and reliance on these carries us a long way, as work stimulated by MP has shown.*

As formulated here, the SMT involves three factors: computational operations, interface conditions, and principles that determine computational efficiency. What follows here will take up each in turn.

2. Interface Conditions

The term *interface* first occurs (in quotations) in Chomsky's writing in the 1977 paper he coauthored with Howard Lasnik [12] (p. 428)[9]:

> *Representations in UP and LF provide the "interface" between linguistic structures and the performance systems. Thus, we assume that all aspects of phonetic form and meaning determined strictly by sentence grammar are indicated in the representation in UP and LF, respectively. The grammar, then, determines a sound-meaning relation; more*

accurately, it determines a pairing of conditions-on-sound and conditions-on-meaning, each of which may be further specified by the interaction of grammar and other systems.

The speculation that conditions on sound and meaning "may be further specified by the interaction of grammar and other systems" (presumably performance systems) is a first step towards the view that the systems that interface with I-language might actually constrain their form and function, which has become a central hypothesis of the Minimalist Program.

In focusing attention on the interfaces, the Minimalist Program along with the SMT that defines its core, constitutes a major shift of focus from the work on generative grammar that precedes it. Thus, starting in 1965 *Aspects of the Theory of Syntax* [13] delineates a crucial distinction between a speaker's "intrinsic tacit knowledge" of the generative system for language (what was designated as linguistic *competence*) and the use of that knowledge, referred to as linguistic *performance*. Prior to the Minimalist Program, the focus of generative grammar was almost exclusively on the system that constitutes a speaker's knowledge of a language. In stark contrast, the SMT suggests that the cognitive systems that interface with the faculty of language FL actually constrain the linguistic structures FL generates.

Although the suggestion is quite plausible, it raises questions of interpretation. Are these constraints formulated within the interface systems themselves or are they internal to UG but motivated by the requirements of these systems? The UG internal interpretation is expressed in the second above-mentioned minimalist assumption that conditions on representations (e.g., those of Case, binding, and θ-theories) are, in effect, interface conditions, i.e., holding "only at the interface" and "motivated by properties of the interface, perhaps properly understood as modes of interpretation by performance systems." This of course raises the question of how the Case filter, binding principles, and the θ-Criterion are each motivated by properties of the cognitive systems that interface with language.

A partial answer to this further question comes from a consideration of the principle of Full Interpretation FI, formulated as a principle of economy such that "there can be no superfluous symbols in representations" and identified as "the "interface condition"" in the first chapter of [3] (p. 27) (Chomsky & Lasnik 1993 [17]). As first formulated in Chomsky 1986 [18] (p. 98), FI "requires that every element of PF and LF, taken to be the interface of syntax (in the broad sense) with systems of language use, must receive an appropriate interpretation—must be licensed in the sense indicated. None can simply be disregarded." Presumably "appropriate interpretation" is determined by the systems that interface with PF and LF representations, with interpretations being assigned by these systems. This requirement of all elements in PF and LF representations guarantees that they cannot contain superfluous symbols, which receive no interpretations, and also that no element (especially one that receives no interpretation) can be "disregarded". At PF, for example, a representation containing a symbol that has no sensorimotor interpretation "does not qualify as a PF representation" [3] (p. 27).

Applied to LF, FI requires that "every element in a representation have a (language independent) interpretation", assumed to bar true expletives and vacuous quantification [3] (p. 27). As formulated, this requirement is imposed by the conceptual-intentional systems CI that interface with language. The question is how it relates to UG internal interface conditions such as the Case filter and the θ-Criterion, or, more precisely, how it might subsume them. Consider the subpart of the latter that requires each argument to bear a θ-role.[10] This would account for the deviance of examples such as **the teacher was praised the student*, where *the teacher* as subject of a passive construction presumably would not be assigned a θ-role, thus violating the constraint. In [18], just before the initial proposal of FI, it is suggested that a θ-role can only be assigned to a Case-marked argument, in which case Case Filter violations could be subsumed under this subpart of the θ-Criterion. Therefore, neither *the teacher* nor *the student* could be assigned a θ-role. The further reduction of this subpart of the θ-Criterion to FI would then require an explanation of how, for example, both *the teacher* and *the student* in the deviant expression fail to receive a language independent interpretation from CI.

Such an explanation would require an understanding of the CI systems that interface with language. However, as noted early on in *The Minimalist Program* [3], "we do not know enough about the "external" systems at the interface to draw firm conclusions about conditions they impose, so the distinction between bare output conditions and others remains speculative in part" (p. 222).[11] The problem at this point is that over a quarter of a century further on, our understanding of these external systems remains basically the same. FI remains the only potential candidate for a bare output condition at the CI interface, and it is still unclear how it operates as such.

This apparent problem with the formulation and interpretation of bare output conditions at the CI interface may, in fact, be a surface reflection of deeper problems. While there can be no question that I-languages interface with performance systems, nonetheless, the SMT formulation in terms of interface conditions is, at best, a promissory note facing three virtually insurmountable obstacles. To demonstrate how it works in any detail requires an understanding of how knowledge is translated into behavior, which we don't possess for any domain of inquiry. That understanding would include the relation between language and thought (in both directions), which remains essentially a mystery. Thoughts—whatever they actually are—come from a rich cognitive stew in the mind of a person encompassing that person's knowledge, beliefs, emotions, and reasoning. When a thought is packed into a linguistic expression and externalized, it loses that cognitive richness it had in the mind of the author and only regains this when it is taken into the mind of another person, with no guarantee that it will be understood in exactly the same way. The problem for understanding the relation between language and thought is that while we have a formal theory of language, there is no equivalent theory of thought (essentially the same situation as the mind-body problem). Furthermore, there is no one-to-one correspondence between sentences and thoughts, as the process of editing our own writing demonstrates, where syntax and lexical choices are altered to express a thought more clearly. This raises a fundamental question for linguistic performance of why a thought is actually expressed in one particular sentence when there are, in fact, numerous other sentences in which it could have been expressed.[12] Ultimately, a theory of linguistic performance will have to account for what Chomsky has called "the creative aspect of language use", which has remained a mystery since Descartes first discussed it close to four hundred years ago (see [21]). How all of this affects the interface condition formulation of the SMT remains an open question.

All of this becomes moot for the formulations of the SMT from 2015 to the present, where the reference to interface conditions no longer appears.[13] Moreover, in Chomsky's 2019 UCLA lectures [22] the existence of interface levels is called into question (p. 11):

> *Derivations yield structures interpreted at two interfaces: conventionally termed SM and CI; postulation of interface levels is in fact superfluous; it is enough to say that extra-linguistic systems access derivations.*

And this position is reiterated in Chomsky's most recent paper [23] (p. 7):

> *It is conventional to speak of two interfaces, the SM and Conceptual-Intentional (CI) interfaces. While the device is convenient for exposition (and I'll adopt it here), there is no need to postulate the interface levels; access can, in principle, take place at any stage of the computation.*

Instead, from 2015 on, the SMT is formulated in terms of the simplest computational operation interacting with principles of minimal computation (computational efficiency).

> *In the best case, phenomena would be explained by interaction of the simplest computational operation—Merge, with its two logically possible subcases. Internal Merge IM (automatically yielding "the copy theory of movement") and External Merge EM—interacting with general principles of minimal computation MC. The Strong Minimalist Thesis SMT articulates this goal.* [16] (p. 4)

The UCLA Lectures places this renewed emphasis on computational operations within the historical context of the Minimalist Program from its inception (p. 20):

> *Well, by the early 1990s it appeared to a number of people, myself included, that enough had been learned so that it might be possible to tackle the whole general problem of language on more principled grounds—that is, to actually seek genuine explanations, to formulate what was called the "Strong Minimalist Thesis", which says that language is basically perfect, what's called "approaching Universal Grammar from below", starting by assuming (let's see how far we can get) a perfect answer that would be a genuine explanation and how much can we explain this way. That's what's been called the Minimalist Program. Now the way to proceed with it is to start with the simplest computational operation and just let's see how far you can go. Where do you run aground?*

The shift in focus from interface conditions (1992–2013) to the simplest computational operation is actually a return to the origins of modern generative grammar, where syntax is defined—for the first time in the study of language—as "the study of the principles and processes by which sentences are constructed in particular languages."[14]

Returning to Chomsky's 2015 remark about the absence of the SMT in the 1995 edition of *The Minimalist Program* that "the thesis seemed too extreme to be seriously proposed," it should be clear from the discussion in this section that this couldn't legitimately apply to the interface condition formulation of the SMT: first, because one assumption stated in the initial proposal for a minimalist program actually states that formulation, and second, because what we know about specific conditions that the interfacing cognitive systems might impose had not changed in any significant way. At best, the only candidate for the status of external interface condition was (and remains) Full Interpretation (first formulated six years before the advent of the Minimalist Program), which as discussed above raises difficult questions of interpretation regarding how I-language and C-I systems actually interface.

3. The Simplest Computational Operation

In 1995, what may have "seemed too extreme to be seriously proposed" was the reduction of UG to "the simplest computational operation"—note the singular. The skepticism about the SMT was, in part, (or perhaps primarily) grounded in the view that the property of displacement was an imperfection of language. As Chomsky explains, "Displacement had always seemed—to me in particular—a curious imperfection of language. Why should languages resort to this device in a very wide range of constructions?" (p. x). From this perspective there are two aspects to the "imperfection" of the displacement property: (1) the formulation of the computational operation required in addition to Merge, and (2) the result that UG could not be reduced to "the simplest computational operation".

This perspective is grounded in the origins of modern generative grammar, where in the 1950s, what have come to be called displacement phenomena were accounted for by a special computational operation originally called a *transformation*. In the theory of the 1950s transformational operations applied to the output of a phrase structure rule component that accounted for the composition and labeling of hierarchically structured linguistic expressions and, in addition, the linear ordering of the elements they contained. Consider, for example, what was designated as the passive transformation in *Syntactic Structures*:

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

As formulated, the passive transformation performs a simultaneous double displacement: the NP following V to a position preceding Aux, and the NP originally preceding Aux to a position following *by* which has been inserted after V. In addition, the transformation inserts the passive auxiliary *be+en* between Aux and V and also the "passive" *by* in front of the displaced (underlying) subject NP. Clearly not a simple computational operation.[15]

Technically, a transformation applies to a phrase-marker generated by phrase structure rules, in some cases modified by the prior application of other transformations. For example, the derivation of the passive sentence *the thieves were arrested by the police* would involve the following phrase structure rules (cf. Appendix II of [24]).

Sentence	→	NP VP
VP	→	Verb NP
NP	→	T N
Verb	→	Aux V
Aux	→	C (M) (have + en) (be + ing)
T	→	the, ...
N	→	thieves, police, ...
V	→	arrest, ...

The output of each rule is a string of symbols; the output of these rules in the derivation of a sentence yields a set of strings, a *phrase-marker*. For the passive sentence under consideration, the phrase-marker before the application of the passive transformation would be

{Sentence, NP VP, NP Verb NP, NP Aux V NP, NP C V NP, T N C V T N, ... }

The ... in the set would be further specified by replacing each lexical category symbol T, N, V with the relevant lexical item. Applying the phrase structure rules in different orders would yield distinct but equivalent phrase-markers, each one yielding the same hierarchical structure.

Applying the passive transformation, as formulated above to this phrase marker converts the designated string NP-Aux-V-NP to a new string NP-Aux-*be+en*-V-*by*-NP—more precisely, the string NP_a-Aux-V-NP_b to NP_b-Aux-*be+en*-V-*by*-NP_a, where the subscripts indicate the double displacement of the NPs *the thieves* and *the police*. Presumably the double displacement would also affect the other strings in the derived phrase marker, so that, for example, the string NP_a-VP would be replaced by the string NP_b-VP and so on, based on the assumption that the phrase-marker expresses the constituent structure of the sentence under analysis. Even so, the passive transformation as formulated leaves open the actual derived constituent structure of passive sentences.

As formulated, the structural analysis of the passive transformation restricts the phrase-markers to which it can apply, and the structural change specifies how it alters the phrase-marker derived. These alterations result from the application of the elementary operations that compose the transformation. In the case of the passive transformation, two movement operations and two insertion operations apply. A formal analysis would have to specify the underlying elementary operation that yields movement.

Both the derived constituent structure of passive sentences and the elementary operation that yields displacement are clarified in the developments in the theory of transformations that followed (1965–1981), where this early formulation of a passive transformation is ultimately subsumed under a more general rule of Move α—in the case of passives, α is a NP. In a passive construction, the underlying verbal object NP is preposed to replace the underlying subject NP via the elementary operation of substitution. Given the non-distinctness condition on substitutions (see *Aspects of the Theory of Syntax* [28]), which intersects with a recoverability condition on deletions, the underlying subject NP replaced would have to be empty—generated by the phrase structure rule for *Sentence* (see above), but where the rule expanding NP does not apply. Under the substitution analysis of passives, the derived constituent structure is essentially identical to the underlying constituent structure generated by the basic phrase structure rules.[16] Furthermore, the analysis no longer makes any reference to a structural analysis of an input string for the transformational operation.

In the first formulation of Merge [6] the fundamental dichotomy between phrase structure rules and transformations is preserved. Merge is introduced as one operation of the computational system for human language C_{HL} that operates on an array of lexical items, technically a *numeration*, to construct linguistic expressions.[17] The first formulation is presented as (p. 62):

> *Given the numeration N, C_{HL} may select an item from N (reducing its index) or perform some permitted operation on the structures it has already formed. One such operation is necessary on conceptual grounds alone: an operation that forms larger units out of those already constructed; call it Merge. Applied to two objects α and β, Merge forms the new object γ. What is γ? γ must be constituted somehow from the two items α and β; the only alternatives are that γ is fixed for all α, β, or that it is randomly selected; neither is worth considering. The simplest object constructed from α and β is the set $\{\alpha, \beta\}$, so we take γ to be at least this set, where α and β are the constituents of γ. Does that suffice? Output conditions dictate otherwise; thus, verbal and nominal elements are interpreted differently at LF and behave differently in the phonological component (see note 9). γ must therefore at least (and, we assume, at most) be of the form $\{\delta, \{\alpha, \beta\}\}$, where δ identifies the relevant properties of γ; call δ the label of γ.*

In this formulation Merge specifies both the composition and labelling of syntactic objects, segmentation and classification (the classical discovery procedures of structuralist linguistics), which also results from the application of phrase structure rules. It differs from phrase structure rules in not specifying a linear order to the structures generated. Where α, β and $\{\alpha, \beta\}$ are syntactic objects, δ is clearly not, and $\{\alpha, \beta\}$ is merely the unlabeled counterpart to $\{\delta, \{\alpha, \beta\}\}$.[18]

In contrast, the operation Move (Move-α) is described as a "second operation that forms phrase markers" (p. 69).[19] Its formulation is given in the following sentence as "Given the phrase marker S with terms K and α, Move targets K, raises α, and merges α with K to form the new category with the constituents α, K." As described, the operation Move involves three, presumably independent, elementary operations: targets, raises and merges. As formulated, K is not necessarily the root of S, nor is a necessarily contained in K. But if Move is constrained in the same way that Merge is, then merger of α must be merger at the root, with K the root—in which case, raising follows as a consequence. Moreover, there would be no need to invoke targeting as separate from merger. In effect the elementary operation that defines Merge would also define Move. Footnote 29 (p. 104) approaches this perspective, but then dismisses it.

> *One could define Merge differently, of course, allowing 'internal Merger'. But there is no conceptual advantage in that; in fact, the definition is more complex.*

As developments nine years later established, what seems to go wrong here is the assumption that to incorporate Move, the definition of Merge would have to become "more complex".[20]

The discussion of Move vs. Merge in the fourth chapter of *The Minimalist Program* proposes a formulation of Move that is clearly incompatible with Merge. While Merge operates on syntactic objects (lexical items and larger structures that contain them), Move is applied to a feature contained in these syntactic objects and not the objects themselves.

> *So far, I have kept to the standard assumption that the operation Move selects α and raises it, targeting K, where α and K are categories constructed from one or more lexical items. But on general minimalist assumptions, that is an unnatural interpretation of the operation. The underlying intuitive idea is that the operation Move is driven by morphological considerations: the requirement that some feature F must be checked. The minimal operation, then, should raise just the feature F: we should restrict α in the operation Move α to lexical features. Let us investigate what happens if we replace the operation Move α by the more principled operation Move F, F a feature.* (p. 262)

So, when *The Minimalist Program* was published in 1995, it seemed clear that UG involved more than the simplest computational operation; and, moreover, that while Merge appeared to be a relatively simple operation, Move was a complicated one.

> *We are now tentatively assuming that if all features of some category α have been checked, then α is inaccessible to movement, whether it is a head or some projection. But if some feature F is, as yet, unchecked, α is free to move. Economy conditions exclude "extra" moves and anything more than the minimal pied-piping required for convergence. In covert movement, features raise alone. Procrastinate expresses the preference for the covert option.* (p. 266)

A Move F analysis of displacement/dislocation required another operation to account for the overt movement of α: "minimal pied-piping". The formulations of Merge vs. Move maintain the dichotomy that existed between their precursors, phrase structure rules and transformations. Therefore, at that point, postulation of a strong minimalist thesis in terms of reducing UG to the simplest computational operation was demonstrably "too extreme to be seriously considered."

Given the formulations of both Merge and Move in *The Minimalist Program*, the path to the simplest computational operation involves three steps. One, the labelling function is eliminated from the formulation of Merge. The work described in [1] explores the possibility that labels are predictable, and [8] proposes that in the best case the label of the syntactic object constructed by Merge is "predictable by general algorithm" (p. 3). Thus, Merge has a single function, the composition of syntactic objects, which is "the indispensable operation of a recursive system" (p.3). Two, recognition that the elementary operation Merge is part of the operation Move. This perspective occurs in the initial discussion of Move vs. Merge in [6] but was suppressed when [6] was incorporated into the fourth chapter of *The Minimalist Program* (see above). In [1], Move is defined as a complex operation consisting of two basic operations: Agree and Merge.

> *First, what operations enter into this component of C_{HL}? One is indispensable in some form for any language-like system: the operation $\underline{\text{Merge}}$, which takes two syntactic objects (α, β) and forms K(α, β) from them. A second is an operation we can call $\underline{\text{Agree}}$, which establishes a relation (agreement, Case checking) between an LI α and a feature F in some restricted search space (its $\underline{\text{domain}}$). Unlike Merge, this operation is language-specific, never built into special-purpose symbolic systems and apparently without significant analogue elsewhere. We are therefore led to speculate that it relates to the design conditions for human language. A third operation is $\underline{\text{Move}}$, combining Merge and Agree. The operation Move establishes agreement between α and F and merges P(F) to αP, where P(F) is a phrase determined by F (perhaps but not necessarily its maximal projection) and αP is a projection headed by α.* (p. 101)

How the phrase determined by F is calculated constitutes an extra step, designated as generalized pied-piping, which feeds Merge.

> *Plainly Move is more complex than its subcomponents Merge and Agree, or even the combination of the two, since it involves the extra step of determining P(F) (generalized "pied-piping").* (p.101)

The third and final step happens with the realization that what had been considered to be two separate computational operations, Merge and Move, were in fact the same elementary operation applied in two distinct ways. As expressed in [10]:

> *N[arrow] S[yntax] is based on the free operation Merge. SMT entails that Merge of a, b is unconstrained, therefore either external or internal. Under external Merge, a and b are separate objects; under internal Merge, one is part of the other, and Merge yields the property of "displacement," which is ubiquitous in language and must be captured in some manner in any theory. It is hard to think of a simpler approach than allowing internal Merge (a grammatical transformation), an operation that is freely available. Accordingly, displacement is not an "imperfection" of language; its absence would be an imperfection.* [footnote omitted] (p. 110)

As the simplest formulation for a recursive operation that yields the structured expressions of language, Merge constitutes the null hypothesis, which requires no explanation, in accord with the formulation of the SMT in this paper whereby there are no unexplained elements of UG. Therefore, it comes for free as well as being unconstrained (i.e., it applies freely).

The application of Merge designated as internal automatically yields the displacement property. For example, in the derivation of the passive sentence *the thieves were arrested*, external Merge would construct the syntactic object {were {arrested {the thieves}}}; then internal Merge would apply to that structure and {the thieves} contained in it to yield {{the thieves} {were {arrested {the thieves}}}}. The result is a single syntactic object that has two contexts: the object of the verb *arrested* and the subject of the passive sentence (though not the subject of the verb *arrested* where the verb assigns an argument function (θ-role) to its subject). In this way, the "copy theory of movement" follows as part of the null hypothesis status of Merge.[21]

In conclusion, for the past twenty years, Merge has remained the simplest computational operation in accord with the 2015 formulation of the SMT in [16].

4. Computational Efficiency

Although *computational efficiency* is briefly mentioned at the conclusion of Chomsky's 1991 paper "Some Notes on Economy of Derivation and Representation" [31] with regard to "the structure of formal systems constructed for computational efficiency", it does not appear again until the third factor of language design is proposed as a topic of investigation in [32]. The abstract for that paper states:

The Principles-and-Parameters approach opened the possibility for serious investigation of the third factor, and the attempt to account for properties of language in terms of general considerations of computational efficiency, eliminating some of the technology postulated as specific to language and providing more principled explanation of linguistic phenomena.

On page 6 "general considerations of computational efficiency" are characterized as "principles of efficient computation", which are language-independent (p. 9). Yet as [10] cautions, "the questions are empirical at every point, including the kinds of computational efficiency that FL selects" (p. 106).

Predating this discussion by a decade, the Inclusiveness Condition, later characterized as "a natural principle of efficient computation" in [11] (p. 6) and a "desideratum of efficient computation" in [14] (p. 138), is first proposed as a "natural condition" in the fourth chapter of *The Minimalist Program* [3] (p.225). The first formulation states that outputs of C_{HL} "consist of nothing beyond properties of items of the lexicon (lexical features)—in other words, that the interface levels consist of nothing more than arrangements of lexical features." This is restated on the next page somewhat more generally: "Insofar as the condition of inclusiveness holds, the syntactic objects are rearrangements of properties of the lexical items of which they are ultimately constituted." One consequence is the elimination of indices (hence traces) and syntactic categories distinguished in terms of bar levels as in early X-bar theory. Inclusiveness also eliminates nonlexical category labels like *S* and *S′* for clause. The elimination of traces leads directly to the "copy theory of movement" (about which see below).

Although *inclusiveness* is not specifically mentioned in [32], it is incorporated in the introduction on page 11 of a second condition of efficient computation, the No-Tampering Condition (NTC).

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.*

In effect, the first sentence merges the inclusiveness condition "operations forming complex expressions should consist of no more than a rearrangement of the objects to which they apply" and the NTC "not modifying them internally by deletion or (the) insertion of new elements." As noted on page 13, the NTC "also entails the so-called copy theory of movement, which leaves unmodified the objects to which it applies, forming an extended object." The underlying assumption is that the NTC holds for the elementary operation Merge, as [11] explains:

> *NTC has always been assumed without comment for EM: there is, for example, no proposal that if V and NP are merged to form VP, then V is merged inside NP. Under SMT, it should hold for IM as well.* (p. 10)

Thus, the copy theory would follow from either the Inclusiveness Condition or the NTC.

The initial formulation of the NTC above also links it to "the notion of cyclic computation", whereby cyclic computation constitutes another property of computational efficiency. As noted in [11], "optimal computation requires some version of strict cyclicity" (p. 16), which is explicated on page 5:

> *A Merge-based system will be compositional in general character: the interpretation of larger units at the interfaces will depend on the interpretation of their parts, a familiar observation in the study of every aspect of language. If the system is computationally efficient, once the interpretation of small units is determined it will not be modified by later operations—the general property of strict cyclicity that has repeatedly been found.*

The question that arises is whether the NTC and a strict cycle condition are separate conditions of efficient computation. Depending on how the latter is formulated, both could bar the modification of syntactic objects created at earlier stages in a derivation; however, only the NTC prohibits modification via deletion.

To address the question, we might start with the original formulation of the strict cycle condition in [33] (p. 243):

> *Strict Cycle Condition* (SSC)
>
> 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.

This formulation predates Move α and, therefore, assumes a formulation of (transformational) rules that is closer to that of the passive transformation cited above. Thus, in the case of this transformation, *S* would be a cyclic node.[22] Basically, the idea is that operations cannot modify a subdomain of any syntactic object to which they are applying. In the first formulation of the Minimalist Program [2] strict cyclicity is recast as an extension condition such that "substitution operations always extend their target" (p. 23).[23] Substitution operations here refer to the precursors of EM and IM, so in essence Merge always extends the syntactic objects to which it applies. This precludes inserting one syntactic object inside the other, in which case EM "always applies in the simplest possible form: at the root" [3] (p. 248), a conclusion that extends to overt IM (p. 254). This prohibition is also a consequence of the NTC, as noted in [11] where *root* is replaced by *edge* (p. 11):

> *Unless an element Z is an isolated element (an interjection, or frozen expression), hence of no interest here, its label W must have a feature indicating that Z can be merged. Under NTC, merge will always be to the edge of Z, so we can call this an edge feature EF of W.*

Thus, it would seem that strict cyclicity follows directly from the NTC and, therefore, is not by itself an independent principle of efficient computation.[24]

Whether the NTC is itself an independent principle of efficient computation depends on the precise formulation of the computational operation to which it applies, Merge. To this end, we now consider the most recent formulation of Merge as a mapping between workspaces that contain the syntactic elements that are eventually combined into a single hierarchically structured linguistic expression.[25] This mapping occurs because Merge combines two syntactic elements in a workspace to form a new syntactic object, the set containing the two elements combined, thereby creating a new workspace containing that new object.

Consider for example the derivation for *the teacher praised a student*, which minimally involves five lexical items. If the initial workspace (indicated with square brackets) contains all five lexical items selected from the Lexicon[26], what is called a *lexical array* in [1], applying Merge to *a* and *student* yields a new workspace containing the syntactic object {*a*, *student*}:

$$\text{[the, a, student, teacher, praised]} \Rightarrow \text{[the, teacher, praised, \{a, student\}]}$$

And the derived workspace would then map onto another,

$$\text{[the, teacher, praised, \{a, student\}]} \Rightarrow \text{[the, teacher, \{praised, \{a, student\}\}]},$$

where in each mapping the application of Merge increases the number of accessible syntactic objects by one. In the first case, we add to the five lexical items the syntactic object {*a*, *student*} for a total of six. In the second case, we add to the six a seventh {*praised*, {*a*, *student*}}.

Both cases are examples of external Merge (EM), where neither of the two elements merged is contained within the other. The simplest operation would combine the two elements in the workspace into a single syntactic object where the resulting workspace no longer contains the two elements combined as also separate terms. If it did (as in what [22] calls "normal recursion" citing proof theory as an example), then the derived workspace would contain more than one new element.[27] EM renders the hierarchical composition of linguistic expressions.

The other application of Merge involves two syntactic objects where one is contained in the other, hence internal Merge (IM). As an example, consider the derivation of *a student was praised*, which would involve the following step:

$$\text{[\{was, \{praised, \{a, student\}\}\}]} \Rightarrow \text{[\{\{a, student\}, \{\{was, \{praised, \{a, student\}\}\}\}]}$$

The syntactic object [{*was*, {*praised*, {*a*, *student*}}} is merged with the syntactic object {*a*, *student*} which it contains. The two instances of {*a*, *student*} are considered to be copies, where only the higher copy in the structure is pronounced but where the lower copy is interpreted as the object of *praised*. Actually, the higher copy also receives an interpretation as the subject of the sentence (roughly 'what the sentence is about'). In this way IM yields the property of displacement, where the analysis is characterized as "the copy theory of movement".[28]

The creation of copies is assumed to be a consequence of IM, so there is no special additional rule. As stated in [11], "There is no rule of formation of copies or remerge, as has sometimes been supposed; just IM applying in the optimal way, satisfying NTC" (p. 10); see also [15] p. 40. However, the copies analysis raises a potential problem, as discussed in [22], which considers a displaced nominal phrase *Mary's book*, which could be interpreted as either 'the book which Mary owns' or 'the book that Mary wrote'. For the sentence *Mary's book, I want to read*, where *Mary's book* is interpreted as both the topic of the sentence and the object of *read* (and thus would involve two copies, one in each position) it is not possible to interpret one copy one way and the other a different way. Both must have the same interpretation. Furthermore, the two instances of *Mary* must refer to the same person. Thus, copies are governed by a general requirement of "stablility" where they must share the same structure and interpretation.

Stability must also apply to syntactic objects in consecutive workspaces. Thus, in both [. . . , *read*, {*Mary's*, *book*}] and [. . . , {*read*, {*Mary's*, *book*}], {*Mary's*, *book*} must have the same structure and interpretation, what is taken for granted because it is so obvious that it doesn't bear discussion. In this case it follows from the fact that {*Mary's*, *book*} is the same syntactic object.

Stability in displacement tells the same story: what have been called "copies" are actually a single syntactic object, which, when it is linearized at PF, is pronounced in one position as expected—in which case there is no deletion of copies. If this is on the right track, then the copy theory of movement may, in fact, be misleading in that it characterizes displacement as involving multiple copies of a syntactic object in syntactic representations at the CI interface. To understand how this perspective might be correct, let's reconsider what the operation Merge does.

In the derivation of the linguistic expression *the professor praised a student*, when Merge constructs the syntactic object {*praised*, {*a*, *student*}} the item {*a*, *student*} acquires a context, the verb *praised* (and conversely, *praised* acquires the context {*a*, *student*}). In this way, Merge creates an item/context architecture[29] for the syntactic structure of linguistic expressions. This carries over to displacement, as in *a student was praised*. In this case, the syntactic object {*a*, *student*} is assigned a second context, {{*was*, {*praised*, {*a*, *student*}}}; thus, displacement is a phenomenon in which a single syntactic object has multiple contexts. EM is the case where a syntactic object has only one context; IM, the case where a single syntactic object has multiple contexts. There is no need to introduce a notion of "copy" or an operation "form copy" regardless of whether copies are merely a consequence of IM (see above) or formed by an actual operation of C_{HL} (see [13,23]).[30]

The notion 'copy' is inescapable if we are talking about physical objects—including, in particular, representations in black and white on a page. Perhaps this explains why this notion has persisted in the analysis of displacement. But if item/context architecture as discussed above is the correct conceptualization of Merge as the fundamental operation in a computational account of knowledge of language, then that notion appears to play no role here.

As a result, so-called copy deletion would play no role in the determination of computational efficiency—contradicting various claims that it does, beginning with the claim in [15] that it follows from "the third factor principle of minimal computation: pronounce as little as possible" [15] (p. 41).[31] In [16] deleting copies is characterized as "another operation of minimal computation" (p. 5). In [22] this is spelled out as follows:

> *So you have massive reduction in complexity if you just don't pronounce extra copies. So computational efficiency compels the deletion of the copies.* (p. 29)

And in [13], copy deletion is further clarified:

> *In externalization, all but one of a copy set is deleted by a third factor principle of computational efficiency.* (p. 11)

See also [23], footnote 15. However, if the account of Merge in terms of item/context architecture is viable, then in this case it is the formulation of the simplest computational operation (Merge) rather than third factor principles of computational efficiency that accounts for the phenomenon of displacement.

At this point the question arises whether this account might be extended to the other two proposals for principles of efficient computation: the Inclusiveness Condition and the NTC—assuming that cyclic computation reduces to the latter as suggested above. The issue with the Inclusiveness Condition is whether Merge as formulated is in any way capable of creating structure beyond the rearrangement of items from the Lexicon. Given the formulations of Merge that are restricted to composition (including displacement) inclusiveness follows from the elements in the Lexicon, so there is no need to articulate an independent principle to achieve this result.

The case of the NTC is slightly more complicated. Imagine how Merge might apply in violation of the NTC. Suppose the initial workspace contains the lexical elements x, y, and z. The following derivation would violate the NTC:

$$[x, y, z] \Rightarrow [x, \{y, z\}] \Rightarrow [\{\{x, y\}, z\}]$$

The second step would violate the NTC by tampering with the syntactic object $\{y, z\}$ that had been constructed in the first step. It would change the item/context architecture in that z, which had been in the context of y in the first step, is now in the context of $\{x, y\}$. The second step could be generated without violating the NTC if in the first step x is merged with y, but then z would never be in the context of y. Now consider a more complicated case where the initial workspace contains w, x, y, and z. and where w has merged with x and y has merged with z, yielding $[\{w, x\}, \{y, z\}]$. If the next step merges x and y, then the result could be either

$$[\{w, x\}, \{\{x, y\}, z\}] \text{ or } [\{w, \{x, y\}\}, \{y, z\}],$$

thereby eliminating either y in the context z or x in the context w. This step differs from normal EM and IM where Merge accesses two, and only two, syntactic objects. With IM, Merge accesses a syntactic object and then a second that is contained in that object. In all the violations of NTC given above, Merge is accessing more than two syntactic objects in the workspace—three in the first case and four in the second. If the formulation of Merge restricts access to two syntactic objects, then no-tampering follows from the restriction: hence, the formulation of the simplest computational operation as binary Merge in the strictest sense. Alternatively, it could be that this Access Restriction (not to be confused with the resource restriction mentioned in footnote 27)[32] is actually a principle of efficient computation. If not, then that part of SMT that references general principles of minimal computation is, at this point, in want of specific proposals, and what we are left with is the simplest computational operation now understood as Merge, which recursively maps a workspace onto a modified workspace to generate the hierarchical composition of linguistic expressions.[33]

5. Beyond the Simplest Computational Operation?

The most recent formulations of the SMT in [13,23] take a step back from the simplest computational operation, proposing instead "the simplest computational operations" as the focus. In these articles, the SMT is developed in the context of a conundrum involving the apparently contradictory goals of UG [23] (p. 7):

[1] (i) It must be rich enough to overcome the problem of poverty of stimulus (POS), the fact that what is acquired demonstrably lies far beyond the evidence available
(ii) It must be simple enough to have evolved under the conditions of human evolution
(iii) It must be the same for all possible languages, given commonality of UG.

Ref. [13] explicates the conundrum as follows:

> *The core system of language might be largely immune to the learnability condition [1i], which has played a prominent role in the generative enterprise. The variability problem [1iii] will be assigned to an ancillary system, the amalgam of language and some SM system. [footnote omitted] The conundrum will then reduce to satisfying the evolvability condition [1ii]. That problem will be overcome, at least for the core computational system of language–our topic here–to the extent that the structures of I-language are generated by the simplest operations. The Strong Minimalist Thesis SMT sets this outcome as a prime goal of the theory of language.*

The question then is what operations are required in addition to Merge and how do they qualify as "simplest computational operations".

One phenomenon raised in [13,22,23] that is analyzed as beyond the capabilities of Merge (specifically set Merge) is characterized as unbounded unstructured coordination. The following example is cited in [22]

> *I met someone young, happy, eager to go to college, tired of wasting his time, . . .*

Where the string of adjective phrases following *someone* is potentially unbounded (hence the . . .), with no structural relation between them. But each adjective phrase modifies the same noun *someone*. In terms of an item/context architecture, this could be described as multiple items (the adjective phrases) that have the same context (the noun *someone*, which each phrase modifies). Thus, coordination would be a case of a single context that has multiple items, in contrast to displacement where a single item is assigned multiple contexts. But given that these adjective phrases have no structural relation to one another, they could not be constructed by Merge on a single plane (i.e., in the same dimension). Rather, Merge would have to be able to form multidimensional structures where the noun is accessible to each adjective phrase and what appears to be a sequence of adjective phrases is the result of linearization for the SM interface.[34] Whether this analysis can be extended to the wider range of more complicated cases remains to be determined.

What should be stressed in conclusion is how the discussion of the SMT over the past almost thirty years has shifted focus back to the principles and processes by which sentences are constructed, thereby returning to the concerns which launched the generative enterprise over seven decades ago.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Notes

1 See [4] for an overview of the minimalist program as presented in the book. As Chomsky makes clear in the 1992 paper, the minimalist program is formulated in terms of the Principles and Parameters framework, thereby incorporating the perspective and some of the proposals of earlier work in generative grammar. The introduction to the 20th anniversary edition of the 1995 collection describes it as "a seamless continuation of pursuits that trace back to the origins of generative grammar, even before the general biolinguistics program, as it is now often called, began to take shape in the 1950s" (p. vii). See also [5] for a discussion of some fundamental ideas the minimalist program is investigating that can be found in some form in the linguistic theory that precedes it.

2 Page references are to the version that occurs in [3].

3 See [8] for some discussion.

4 These include "acquisition, processing, neurology, language change, and so on" (p. 96).

5 Language design has been a constant focus in discussion of the minimalist program from the outset. It predates the proposal of a minimalist program [2] in Chomsky's "Economy of Derivation and Representation" [12]. It is worth noting that the topic as currently discussed first appears in Chomsky & Lasnik's "Filters and Control" 1977 [12] (p. 434) when they state:

It is reasonable to suppose that the design of language should facilitate the perceptual strategies that are used by the hearer. It is possible, though hardly necessary, that general properties of grammar might be explained, in part at least, in terms of the exigencies of performance.

6 The words *perfect* or *perfection* do not occur in the initial formulation of a minimalist program [2], but the second paper on the minimalist program two years later [6] begins with a speculation about whether "language is something like a "perfect system." This is repeated in the introduction to [3] the following year, which begins:

This work is motivated by two related questions: (1) what are the general conditions that the human language faculty should be expected to satisfy? and (2) to what extent is the language faculty determined by these conditions, without special structure that lies beyond them? The first question, in turn, has two aspects: what conditions are imposed on the language faculty by virtue of (A) its place within the array of cognitive systems of the mind/brain, and (B) general considerations of conceptual naturalness that have some independent plausibility, namely, simplicity, economy, symmetry, nonredundancy, and the like?

Chomsky notes that in contrast to question A, question B "has an exact answer, though only parts of it can be surmised in the light of current understanding about language and related cognitive systems", an appraisal which accurately assesses the current situation. Nonetheless, it is in this context that the speculation arises:

To the extent that the answer to question (2) is positive, language is something like a "perfect system," meeting external constraints as well as can be done, in one of the reasonable ways. The Minimalist Program for linguistic theory seeks to explore these possibilities.

So pretty much from the outset, the Minimalist Program is concerned with exploring the possibility that the system of language not only functions in an optimal way, but may be in fact a perfect system in the way it interfaces with other cognitive systems.

7 Or "genuine explanations" in the terminology of Chomsky 2021a [13].

8 GPs abbreviates "generative procedures".

9 The designation UP for "universal phonetics" later becomes PF for "phonetic form".

10 Cf. The Functional Relatedness constraint in [19].

11 The term "bare output condition" designates conditions external to UG, in contrast to UG internal constraints such as filters and conditions on derivations. Ref. [3] goes on to say: "The problems are nevertheless empirical, and we can hope to resolve

them by learning more about the language faculty and the systems with which it interacts. We proceed in the only possible way: by making tentative assumptions about the external systems and proceeding from there" (pp. 222–223).

12 As a striking demonstration, consider the over five million variants of the first sentence of Jane Austen's *Pride and Prejudice* enumerated in [20] that might have appeared in its place had Austen not been such a brilliant writer.

13 The last reference to interface conditions in statements of the SMT occurs in the introduction to the 20th anniversary edition of *The Minimalist Program* (quoted above), but is not mentioned in the formulation given in "Problems of Projection: Extensions" published the same year (quoted below).

14 On the first page of [24]; see [25] for a discussion of the revolutionary nature of this definition in the history of linguistics.

15 For a detailed review of the successive changes the passive transformation underwent from its initial formulation to its penultimate simplification as the operation Move α (see [26]). In particular, the analysis in which the subject of a corresponding active sentence is displaced (or demoted) in the passive no longer applies. But see Collins 2005 for an attempt to reconstruct the full *Syntactic Structures* analysis within current analyses of clause structure [27].

16 Transformations whose output can be filtered in this way by the phrase structure rules are structure-preserving, as discussed in Emonds's 1970 Ph.D. dissertation [28] and his 1976 book [29], the Structure Preserving Hypothesis (SPH), as it came be called. In the fourth chapter of [3] the SPH is criticized for introducing "an unwanted redundancy" on the grounds that the movement of a constituent creates a position without there having to be an empty position into which the constituent moves. However, Emonds's formulations of the SPH do not depend on this substitution analysis.

17 A numeration consists of a set of lexical items each associated with a numerical index that indicates how many times a particular item has been selected from the lexicon. Thus, in the passive sentence discussed above, the article *the* would have the index 2 while all the other lexical items in the array would have an index of 1. Each time an item in the numeration is affected by an operation of C_{HL}, its index is reduced by one. At the end of a derivation, each item in the numeration should have an index of zero.The device of a numeration disappears from discussion after chapter 4 of *The Minimalist Program*. The perhaps obvious alternative is a lexical array where a single lexical item can occur more than once. The goal of a derivation would then be a lexical array that is empty. If, alternatively, the lexical array constitutes the initial workspace for a derivation, then the goal would be to reduce the workspace to a single syntactic object, the linguistic expression derived from the interaction between C_{HL} and the lexicon. See below for further discussion.

18 IM also specifies the composition of linguistic expressions, so composition and displacement are not disjoint properties.

19 Note that phrase marker here can't refer to a set of strings as originally defined given the elimination of phrase structure rules that produce such sets. Rather, it can only refer to the hierarchically structured syntactic object that is produced by Merge.

20 It is important to note that these references to merger in the description of Move are eliminated in chapter 4 of *The Minimalist Program*, which incorporates [6], much of it verbatim.

21 Ref. [15] raises an issue for the copy theory regarding a distinction between copies and "repetitions". However, as discussed in [30], the definition of copy is straightforward whereas formulating a definition for "repetition" is problematic (see also [23]).

22 For a detailed discussion of this formulation and how its empirical effects can be derived from independently motivated general principles, see [19].

23 For discussion of some empirical effects of the extension condition, see [34], which concludes (p. 96):

If this analysis is on the right track, then the reason neither countercyclic Merge nor countercyclic Move can apply is that the elementary operation itself does not permit this. On minimalist assumptions, this should be the optimal explanation. It makes no appeal to external factors such as economy, constraints on linear ordering, or other UG principles. Certain kinds of derivations are excluded simply on the grounds that the elementary operations of C_{HL} are so constructed that they cannot occur.

It should be noted that this was written when Merge and Move were treated as independent operations.

24 See [35] for further discussion.

25 In recent work (e.g., [22]) this formulation is designated as MERGE, to distinguish it from earlier formulations. In earlier work, a workspace containing syntactic objects constructed by Merge is tacitly assumed, if not mentioned—the exception being [36] where Merge is presented as an operation that maps a workspace onto a modified workspace. The earliest reference to "workspace" in Chomsky's writings occurs in [1] where the term is equated with "active memory" (p. 106).

26 If Merge is formulated as a mapping between workspaces, then presumably Merge does not operate on the Lexicon directly. Placing the Lexicon in the workspace creates a situation where the entire content of the Lexicon would be accessible to Merge at any point in the derivation, undermining efficient computation (which presumably is limited to the lexical items of the linguistic expression under analysis).

27 See [22] for discussion of how for natural language this could lead to any kind of island violation. Ref. [22] proposes a "resource restriction" limiting new syntactic objects in a derived workspace to one (cf. [36] where it is first mentioned as an observation), thereby forcing the elimination of the unmerged copies of the newly merged elements. See [22] for discussion of how this restriction prohibits, in principle, operations such as Parallel Merge, Sidewards Merge, and Late Merge. However this would also follow from a formulation of Merge where elements merged do not remain in the workspace as unconnected terms.

28 IM also specifies the composition of linguistic expressions, so composition and displacement are not disjoint properties.

29 This terminology originates with Jean-Roger Vergnaud (p.c. January 2004), see [37].

30 Refs. [13,31] propose a separate operation Form Copy that applies not only to the output of IM for displacement, but also to a range of other syntactic phenomena, including obligatory control, across-the-board deletion, and parasitic gaps. The analysis is grounded in a new idea that the SMT has an "enabling function" as well as a restrictive function.

SMT provides options and systems for language that would have no reason to exist if language did not abide by SMT. [13] (p.6)

31 How this actually applies to the obvious counterexamples, virtually every case of ellipsis, where deletion of phonetic material is entirely optional, remains to be determined.

32 Given the Access Restriction proposed here, the empirical effects of the Resource Restriction follow as a consequence. However, the Resource Restriction does not subsume the NTC.

33 This leaves open the large question of locality constraints on IM and thus the anlaysis of the phenomenon of syntactic islands. The Phase Impenetrability Condition (see [8,14]) seems at this point to be a UG-specific constraint on derivations, though in limiting the scope of IM it might be thought of as a principle of minimal computation.

34 Refs. [13,22,23] propose this n-dimensional analysis for adjuncts (including unbounded unstructured coordination (of which the term itself is an example)), employing another operation Pair-Merge to form these structures. Why these structures could not be constructed with Set-Merge is not entirely clear. Refs. [13,23] propose another operation FormSequence for generating coordinate structures, described as "a limited departure from SMT". Whether this operation is in addition to or a replacement of Pair-Merge is unclear.

References

1. Chomsky, N. Minimalist inquiries: The framework. In *Step by Step: Essays in Minimalist Syntax in Honor of Howard Lasnik*; Martin, R., Michaels, D., Uriagereka, J., Eds.; M.I.T. Press: Cambridge, MA, USA, 2000; pp. 89–155.
2. Chomsky, N. A minimalist program for linguistic theory. In *The View from Building 20: Essays in Linguistics in Honor of Sylvain Bromberger*; Hale, K., Keyser, S.J., Eds.; M.I.T. Press: Cambridge, MA, USA, 1993; pp. 1–52.
3. Chomsky, N. *The Minimalist Program*; M.I.T. Press: Cambridge, MA, USA, 1995.
4. Freidin, R. Review article on The Minimalist Program by Noam Chomsky. *Language* **1997**, *73*, 571–582. [CrossRef]
5. Robert, F.; Lasnik, H. Some roots of minimalism in generative grammar. In *The Oxford Handbook of Linguistic Minimalism*; Boeckx, C., Ed.; Oxford University Press: New York, NY, USA, 2011; pp. 1–26.
6. Chomsky, N. Bare phrase structure. In *Evolution and Revolution in Linguistic Theory: Essays in Honor of Carlos P. otero*; Campos, H., Kempchinsky, P., Eds.; Georgetown University Press: Washington, DC, USA, 1995; pp. 51–109.
7. Chomsky, N. *The Minimalist Program*, 20th ed.; M.I.T. Press: Cambridge, MA, USA, 2015.
8. Chomsky, N. Derivation by phase. In *Ken Hale: A Life in Language*; Kenstowicz, M., Ed.; M.I.T. Press: Cambridge, MA, USA, 2001; pp. 1–52.
9. Chomsky, N. *What Kind of Creatures Are We?* Columbia University Press: New York, NY, USA, 2016.
10. Chomsky, N. Beyond explanatory adequacy. In *Structures and Beyond*; Belletti, A., Ed.; Oxford University Press: Oxford, UK, 2004; pp. 104–131.
11. Chomsky, N. Approaching UG from below. In *Interfaces + Recursion = Language?: Chomsky's Minimalism and the View from Syntax-Semantics*; Sauerland, U., Gärtner, H.-M., Eds.; Mouton de Gruyter: Berlin, Germany, 2007; pp. 1–30.
12. Chomsky, N.; Lasnik, H. Filters and control. *Linguist. Inq.* **1977**, *11*, 425–504.
13. Chomsky, N. Genuine explanation. In *Issues in Comparative Morpho-Syntax and Language Acquisition*; Bocci, G., Botteri, D., Manetti, C., Moscati, V., Eds.; forthcoming.
14. Chomsky, N. On phases. In *Foundational Issues in Linguistic Theory: Essays in Honor of Jean-Roger Vergnaud*; Freidin, R., Otero, C.P., Zubizarreta, M.-L., Eds.; M.I.T. Press: Cambridge, MA, USA, 2008; pp. 133–166.
15. Chomsky, N. Problems of projection. *Lingua* **2013**, *130*, 33–49. [CrossRef]
16. Chomsky, N. Problems of Projection: Extensions. In *Structures, Strategies and Beyond: Studies in Honour of Adriana Belletti*; Di Domenico, E., Hamann, C., Matteini, S., Eds.; John Benjamins: Amsterdam, The Netherlands, 2015; pp. 1–16.
17. Chomsky, N.; Lasnik, H. The theory of principles and parameters. In *Syntax: An International Handbook of Contemporary Research*; Jacobs, J., van Stechow, A., Sternefeld, W., Vennemann, T., Eds.; Walter de Gruyter: Berlin, Germany, 1993; pp. 506–569.
18. Chomsky, N. *Knowledge of Language: Its Nature, Origin, and Use*; Praeger: New York, NY, USA, 1986.
19. Freidin, R. Cyclicity and the theory of grammar. *Linguist. Inq.* **1978**, *9*, 519–549.
20. Freidin, R. *Adventures in English Syntax*; Cambridge University Press: Cambridge, UK, 2020.
21. Chomsky, N. *Cartesian Linguistics: A Chapter in the History of Rationalist Thought*, 3rd ed.; Cambridge University Press: Cambridge, UK, 2009.
22. Chomsky, N. The UCLA Lectures. LingBuzz. 2020. Available online: https://ling.auf.net/lingbuzz/005485 (accessed on 21 November 2021).
23. Chomsky, N. Minimalism: Where Are We Now, and Where Can We Hope to Go. To Appear in *Genko Kenkyu*. 2021. Available online: https://www.youtube.com/watch?v=X4F9NSVVVuw (accessed on 21 November 2021).
24. Chomsky, N. *Syntactic Structures*; Mouton: The Hague, The Netherlands, 1957.

25. Freidin, R. Syntactic Structures: A Radical Appreciation. LingBuzz. 2020. Available online: https://ling.auf.net/lingbuzz/004996 (accessed on 21 November 2021).
26. Freidin, R. Conceptual shifts in the science of grammar 1951–1992. In *Noam Chomsky: Critical Assessments*; Otero, C.P., Ed.; Routledge: London, UK, 1994; Volume 1, pp. 653–690.
27. Collins, C. A smuggling approach to the passive in English. *Syntax* **2005**, *8*, 81–120. [CrossRef]
28. Emonds, J. Root and Structure Preserving Transformations. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1970.
29. Emonds, J. *A Transformational Approach to English Syntax: Root, Structure-Preserving, and Local Transformations*; Academic Press: New York, NY, USA, 1976.
30. Freidin, R. Chomsky's linguistics: The goals of the generative enterprise. *Language* **2016**, *92*, 671–723. [CrossRef]
31. Chomsky, N. Some notes on economy of derivation and representation. In *Principles and Parameters in Comparative Grammar*; Freidin, R., Ed.; M.I.T. Press: Cambridge, MA, USA, 1991; pp. 417–454.
32. Chomsky, N. Three factors in language design. *Linguist. Inq.* **2005**, *36*, 1–22. [CrossRef]
33. Chomsky, N. Conditions on transformations. In *A Festschrift for Morris Halle*; Anderson, S., Kiparsky, P., Eds.; Holt, Rinehart and Winston: New York, NY, USA, 1973; pp. 232–286.
34. Freidin, R. Cyclicity and minimalism. In *Working Minimalism*; Epstein, S.D., Hornstein, N., Eds.; M.I.T. Press: Cambridge, MA, USA, 1999; pp. 95–126.
35. Freidin, R. Cyclicity in syntax. In *Oxford Research Encyclopedia of Linguistics*; Aronoff, M., Ed.; Oxford University Press: Oxford, UK, 2017; Available online: https://oxfordre.com/linguistics/view/10.1093/acrefore/9780199384655.001.0001/acrefore-9780199384655-e-319?rskey=MlkDK4&result=1 (accessed on 21 November 2021).
36. Bobaljik, J. In terms of Merge: Copy and Head movement. In *Papers in Minimalist Syntax*; Pensalfini, R., Ura, H., Eds.; MITWPL: Cambridge, MA, USA, 1995; Volume 27, pp. 41–64.
37. Vergnaud, J.R. *Primitive Elements of Grammatical Theory: Papers by Jean Roger Vergnaud and His Collaborators*; McKinney-Bock, K., Zubizarreta, M.-L., Eds.; Routledge: New York, NY, USA, 2014.