SIMPLEST MERGE GENERATES SET INTERSECTION: IMPLICATIONS FOR COMPLEMENTIZER ‘TRACE’ EXPLANATION

Samuel D. Epstein¹, Hisatsugu Kitahara², and T. Daniel Seely³
University of Michigan¹, Keio University², and Eastern Michigan University³

1. Introduction

A unique aspect of the analysis presented in this paper is that even though we propose Merge in its simplest, strictly binary form, it is nonetheless the case that a category can have more than one sister (i.e. more than one element that it was merged with). The multiple sisters of a given category arise at different derivational points, generating a “two-peaked” structure, or more formally, intersecting sets, neither of which is a term of the other, in the set-theoretic notation of bare phrase structure (Chomsky 1995a). We assume an analysis with a very general, independently motivated deletion algorithm, which retains only the highest copy of multiple copies of an internally merged element, for sensorimotor SM systems. This is presumably an application of the independently necessary overarching principle of Minimal Computation (categories or features must be retained in an optimal fashion) entailing that copies cannot have additional phonological features above and beyond the single set of lexical features of the mover. Conversely, we also assume the principle of recoverability of deletion (categories or features must not be deleted in a random fashion). We then argue that our proposed simplest formulation of Merge (generating set intersection when applied “countercyclically”) coupled with the very general laws of Minimal Computation (informally forcing features to be interpreted at most once) and Recoverability (informally forcing features to be interpreted at least once) allows us to deduce the core complementizer-trace phenomena with no further mechanisms required. Complementizer-trace effects turn out to be an immediate consequence of these arguably quite natural and general principles, having no ad hoc language-specific, construction-specific or operation-specific motivation regarding complementizer-trace phenomena. We then address the cross-linguistic variation of complementizer-trace effects and the apparent problem of undergeneration that our analysis faces given that it deduces complementizer-trace effects from deep, i.e. unparameterizable principles.

2. Simplest Merge and its Inevitable Consequence for the Derivation of “Countercyclic Movement”

In this subsection, we provide a brief overview of the development of the simplest conception of Merge from X-bar theory through bare phrase structure in the sense of Chomsky (1995a). The goal is to factor out of the Merge operation any property that can itself be deduced from deeper principles, leaving the optimal structure building operation, what we refer to below as simplest Merge.

* We would like to thank especially Noam Chomsky for his interest in our work, and also, Miki Obata and Mamoru Saito. All errors are, of course, our own. Part of the material in this paper was presented at Glow in Asia IX, hosted by Mie University (September 5, 2012). We are grateful to the participants for clarifying remarks.
2.1. Brief Overview of X-bar Theory in Early Minimalism

X-bar theory sought to eliminate phrase structure rules, leaving only the general X-bar-format as part of UG. Determining that format, and avoiding construction-specific and language-specific phrase structure rules, was the central research goal of subsequent work. In early minimalism, X-bar theory was still given, with specific stipulated properties. The outputs of the applications of structure-building operations – binary and singulary Generalized Transformation GT – were then assumed to be constrained by X-bar theory, by definition (see Chomsky 1993):

(1) a. binary GT
   (i) takes K and K’, where K’ is not a phrase marker within K
   (ii) adds \( \Delta \) external to K
   (iii) substitutes K’ for \( \Delta \), forming K*, which must satisfy X-bar theory

b. singulary GT
   (i) takes K and K’, where K’ is a phrase marker within K
   (ii) adds \( \Delta \) external to K
   (iii) substitutes K’ for \( \Delta \), forming K*, which must satisfy X-bar theory

Basically, binary GT takes two separate syntactic objects and combines them into a single object, which as we’ll see in a moment is the ‘ancestor’ of External Merge. Singulary GT is the precursor of the more recent Internal Merge, where one of the objects being joined together is contained within the other. In effect, X-bar theory, together with its stipulated (or axiomatic) properties (endocentricity, head-to-complement, and spec-head relation) was taken to be a UG filter on transformational output representations.

2.2. Subjecting X-bar Theory to a Minimalist Critique

Under the strong minimalist thesis, however, X-bar theory was not exempt from explanatory scrutiny; it was asked why X-bar theory seems to hold, as opposed to an infinite number of formally definable alternative phrase structure systems. By adherence to Minimalist method, this question prompts the following question: How “should” phrase structures be generated under minimalist (ideal, simplest) assumptions? Chomsky’s (1995a:396) answer was:

*Given the numeration \( N \), \( C_{HL} \) may select an item from \( N \) (reducing its index) or perform some permitted operation on the structure 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 \( \alpha \) and \( \beta \), Merge forms the new object \( \gamma \). What is \( \gamma \)? \( \gamma \) must be constituted somehow from the two items \( \alpha \) and \( \beta \); ... The simplest object constructed from \( \alpha \) and \( \beta \) is the set \( \{\alpha, \beta\} \), so we take \( \gamma \) to be at least this set, where \( \alpha \) and \( \beta \) are constituents of \( \gamma \). Does that suffice? Output conditions dictate otherwise; thus verbal and nominal elements are interpreted differently at LF and behave differently in the phonological component ... \( \gamma \) must therefore at least (and we assume at most) be of the form \( \{\delta, \{\alpha, \beta\}\} \), where \( \delta \) identifies the relevant properties of \( \gamma \), call \( \delta \) the label of \( \gamma \).*

Merge was introduced as an operation (the central structure building operation of the narrow syntax NS), necessary on conceptual grounds alone, and the simplest object \( \gamma \) constructed from \( \alpha \) and \( \beta \) by Merge was taken to be the set \( \{\alpha, \beta\} \). Chomsky (1995a) assumed the set \( \{\alpha,
β} was too simple; it was assumed that empirical adequacy demanded some departure from the simplest assumption (the standard scientific tension between explanation and “empirical coverage”); that is, the set must be labeled as in e.g. \{δ, {α, β}\}, where δ identifies the relevant properties of γ.\(^1\)

Given that an output of Merge is a labeled set γ={δ, {α, β}}, Chomsky (1995a:397-398) asked what exactly the label of γ is:

If constituents α, β of γ have been formed in the course of computation, one of the two must project, say α. At the LF interface, γ (if maximal) is interpreted as a phrase of the type α (e.g. a nominal phrase if its head κ is nominal), and it behaves in the same manner in the course of computation. It is natural, then, to take the label of γ to be not α itself but rather κ, the head of the constituent that projects, a decision that also leads to technical simplification. Assuming so, we take γ={κ, {α, β}}, where κ is the head of α and its label as well.

Under this definition, the label of γ is the head of one of its constituents. If α projects, then the object γ constructed from α and β by Merge is \{H(α), {α, β}\}, where H(α) is the head of α (see also Chomsky 1995b). Additionally, the notion “term” is defined as follows: (i) K is a term of K; and (ii) if L is a term of K, then the members of the members of L are terms of K (Chomsky 1995a:399).

Chomsky (1995a,b) did not discuss exactly how Merge operates to form such labeled sets, but one way is to formulate Merge as an operation consisting of the following two steps:\(^2\)

(2) a. Applied to α and β, where neither α nor β is a term of the other, Merge
   (i) takes α and β, forming \{α, β\}, and
   (ii) takes H(α) and {α, β}, forming \{H(α), {α, β}\}.

b. Applied to α and β, where α is a term of β, Merge
   (i) takes α and β, forming \{α, β\}, and
   (ii) takes H(β) and {α, β}, forming \{H(β), {α, β}\}.

In (2a,b), the second step in effect labels the simplest object \{α, β\} constructed by the first step. A question is whether we can eliminate or derive the empirically desirable aspects of this second step – i.e. whether we can predict or explain by general principles what the label of any α, β pair will be (see Chomsky 2000:133). If the answer is positive, Merge can be formulated in the simplest form: Merge(α, β)⇒{α, β}, with the label eliminated from the representational notation and instead simply identified as H(α) or H(β), as in Chomsky (2012).

2.3. On the Complexity of “Countercyclic” Covert Movement

In addition to the labeling algorithm, Chomsky (1995b:254) noted an additional complexity, one concerning covert movement:

The computational system C_{HL} is based on two operations, Merge and Move. We have

\(^1\) See Collins (2002) and Seely (2006) for discussion of the idea that such labels (and label projection) can be eliminated entirely from the grammar.

\(^2\) Chomsky (1995a,b) assumes that either α or β may project (in principle), but if the wrong choice is made, deviance would result.
assumed further that Merge always applies in the simplest possible form: at the root. What about Move? The simplest case again is application at the root: if the derivation has reached the stage \( \Sigma \), then Move selects \( \alpha \) and target \( \Sigma \), forming \( \{ \gamma, \{ \alpha, \Sigma \} \} \). But covert movement typically embeds \( \alpha \) and therefore takes a more complex form: given \( \Sigma \), select \( K \) within \( \Sigma \) and raise \( \alpha \) to target \( K \), forming \( \{ \gamma, \{ \alpha, K \} \} \), which substitutes for \( K \) in \( \Sigma \).

We can formally represent this additional complexity by adding a third step to the formulation of Merge:\(^3\)

\[
\begin{align*}
(3) \quad & \text{Applied to } \alpha \text{ and } \beta \text{ within } \Sigma, \text{ where } \alpha \text{ is a term of } \beta, \text{ Merge} \\
& (i) \text{ takes } \alpha \text{ and } \beta, \text{ forming } \{ \alpha, \beta \}, \text{ and } \\
& (ii) \text{ takes } H(\beta) \text{ and } \{ \alpha, \beta \}, \text{ forming } \{ H(\beta), \{ \alpha, \beta \} \}, \text{ and } \\
& (iii) \text{ replaces } \beta \text{ in } \Sigma \text{ by } \{ H(\beta), \{ \alpha, \beta \} \} \\
\end{align*}
\]

Again, a question is whether we can eliminate this third step (countercyclic IM replacement, or ‘substitution’ to use Chomsky’s (1995b:254) terminology), along with the second step (labeling), and thereby derive an empirically adequate, simplest formulation of Merge. If the answer is positive, Merge can be formulated in the simplest form: \( \text{Merge (} \alpha, \beta \text{) } \Rightarrow \{ \alpha, \beta \} \).\(^4\)

### 2.4. An Inevitable Consequence of Simplest Merge

In a series of recent papers, Epstein, Kitahara, and Seely (hereon EKS) have proposed precisely this form of simplest Merge. Unlike Chomsky’s Merge, which allows “countercyclic” replacement as in (3iii), simplest Merge cannot perform replacement. EKS have explored its effects (see EKS 2012, Kitahara 2011) including the inevitable consequences of simplest Merge. Let’s review. Consider the following sentence:

\[
(4) \quad \text{Bill ate rice.}
\]

At some point in the derivation of (4), Merge takes the phase-head \( C \) and merges it with TP, yielding (5):

\[
(5) \quad \{ C, \{ T, \{ \text{Bill, \{v, \{ate, rice\}}\}\}\}\}
\]

At this point, \( C \) transmits unvalued \( \phi i \) to \( T \). \( T \), then, functioning as a \( \phi i \)-probe, locates the goal \( \text{Bill} \) (bearing lexically valued \( \phi i \) and unvalued Case), and Agree between \( T \) and \( \text{Bill} \) applies, valuing \( \phi i \) on \( T \) and Case on \( \text{Bill} \). The standard assumption (see Chomsky 2007, 2008) is that as these features get valued, \( \text{Bill} \) is raised to the so-called Spec(ifier) of \( T \) (such raising required by some residue of EPP), yielding (6) (where we use the notion “Spec(ifier)” only for expository purposes):

\[
(6) \quad \{ C, \{ \text{Bill, \{T, \{Bill, \{v, \{ate, rice\}}\}\}}\}\}
\]

---

\(^3\) See Groat and O’Neil (1996) for an alternative approach to covert movement whereby covert movement applies cyclically in the syntax, but the chain-tail is pronounced, as if, in the eyes of PF, the movement has not applied (= the definition of “covert” movement).

\(^4\) In section 4, we suggest that simplest Merge is not only conceptually desirable, but empirically adequate, and that at least with respect to that-trace phenomena, notoriously resistant to explanation, the empirical coverage/explanatory power of the theory is in fact concomitantly increased.
This “countercyclic” application of Merge (which, recall is not simplest Merge) mapping (5) to (6) executes a form of replacement (= (3iii)) since Merge, not applying at the root, “infixes” Bill into SpecTP. In set-theoretic terms, “countercyclic” Merge of Bill to the then specless $T_1 = \{T, \{\text{Bill}, \{v, \{\text{ate, rice}\}\}\}\}\} yields a new specful syntactic object, namely $T_2 = \{\text{Bill}, \{T, \{\text{Bill}, \{v, \{\text{ate, rice}\}\}\}\}\}\}$. But recall, this creation of SpecTP is not at the root, since C was necessarily already externally merged with the specless $TP = T_1$ (as in (5)). Therefore, $T_2$ (with Bill as ‘newly appointed’ SpecTP) replaces specless $T_1$ in $\{C, T_1\}$, removing $T_1$ from $\{C, T_1\}$ (destroying a relation/set) and merging $T_2$ as the newly appointed “sister” to C. Such replacement – $T_2$ replaces $T_1$, and $T_1$ “vanishes” – yields a new CP (= $\{C, T_2\}$). It is important to note that it is this “new” CP (= $\{C, T_2\}$), and not the “old” CP (= $\{C, T_1\}$), that enters into further derivational processes, i.e. the specless $TP = T_1$ has disappeared from the continuing derivation.

EKS (2012) argue that, given the strong minimalist thesis, NS should contain only the simplest structure-building operation, namely Merge(X, Y)=$\{X, Y\}$. EKS then point out that Merge, defined in this simplest form (= (3i) only) cannot replace existing categories (see Freidin 1999 for important earlier discussion of replacement as a non-primitive operation). Thus, it must be the case that NS equipped only with simplest Merge cannot map (5) (= $\{C, T_1\}$) to (6) (= $\{C, T_2\}$) in the manner just discussed above. But then, what kind of object does “counter-cyclic” application of Merge (e.g. raising to specTP) create?

It is important to note that EKS adopts the idea of Chomsky (2007, 2008) that given a lexical array (which itself must contain a phase head C or v), all instances of EM must take place before any instance of IM. This is motivated on grounds of efficiency, specifically: EM involves no search (there is immediate access to lexical items within an array), while IM (and other operations of the NS such as Agree) involve search (into an already-existing syntactic object). Thus, it follows that EM must be exhausted before any other operation can apply. In (5), then, the NS builds up to CP via EM, and only then can IM take place, creating (6)—if replacement were allowed. Thus, the nature of “countercyclic” IM is crucial in that it is in effect “forced” under efficiency. Since, by hypothesis, C is merged to TP before raising of the External Argument to SpecTP, it follows that when such raising does take place, it is “countercyclic” in the sense specified above.

EKS suggest that the “countercyclic” application of Merge cannot execute the complex operation “replacement” but rather necessarily results – speaking informally now – in a “two-peaked” or “doubly-rooted” representation, informally represented in (7) (where indices appear only for expository purposes, and linear order is irrelevant):

---

5 Chomsky (2007, 2008) and EKS (2012) argue, and in fact attempt to deduce, that all instances of EM must take place before any instance of IM, and hence C must be merged to TP before any instance of IM (involving TP) can occur.
In set-theoretic terms, first we created the set \( CP = \{C, T_1\} \) since \( C \) was merged with specless \( T_1 \). Then, the “countercyclic” application of simplest Merge (incapable of replacement, by our assumptions) merges \( Bill \) to \( T_1 \) forming another set, namely \( T_2 = \{Bill, T_1\} \). This results in two distinct but intersecting set-theoretic syntactic objects SOs, which happen to have \( T_1 (= \{T, vP\}) \) as their “shared” element. That is, as a result of this “countercyclic” Merge (of \( Bill \) to \( T_1 \)), the single “workspace” of NS contains two intersecting set-theoretic SOs, \( CP = \{C, T_1\} \) and \( T_2 = \{Bill, T_1\} \), as shown in (8a,b), where neither is a term of the other.\(^6\)

(8)  
a. \( \{C, \{T, \{Bill, \{v, \{ate, rice\}\}\}\}\}\)  
b. \( \{Bill, \{T, \{v, \{ate, rice\}\}\}\}\)

Interestingly, notice that although Merge is the simplest possible operation, crucially binary and incapable of replacement thus creating only two-membered sets, it is nonetheless the case that \( T_1 (= \{T, \{Bill, \{v, \{ate, rice\}\}\}\}) \) has two different sisters. That is, simplest Merge generates intersecting sets: both \( C \) and (“later”) \( Bill \) were merged with \( T_1 (= \{T, \{Bill, \{v, \{ate, rice\}\}\}\}) \) at different derivational points; hence, \( C \) and \( Bill \) are sisters of \( T_1 \).

Given this result, EKS (2012) further argue that, under the law of semantic composition, these two intersecting set-theoretic SOs, each functioning as a root, would not yield a single semantic value if they were sent to the semantic component together, “as one”. Thus, there must be some way to decompose this “two-peaked” or “doubly-rooted” representation in the “workspace” of NS, prior to the semantic component. One possible way out of this situation, outlined by EKS (2012), is to send the two intersecting SOs to the semantic component separately. Specifically, EKS propose that Transfer dissolves this intersecting situation by removing one intersecting set from the “workspace” of NS. In effect, one “peak” (i.e. one of the two intersecting sets) must be sent to the semantic component by Transfer at this point of the derivation. Under this proposal, cyclic Transfer of TP once CP is built is deduced (given that the phase-edge must be left for the derivation to continue). Suppose that (i) the “workspace” contains (8a) (= \( \{C, T_1\} \)) and (8b) (= \( \{Bill, T_1\} \)), and (ii) cyclic Transfer removes (8b) (= \( \{Bill, T_1\} \)). Then, the phase-edge, namely \( \{C, ---\} \), where “---” is transferred material, in effect is left for subsequent operations. Note, if \( vP \), in addition to CP, is a phase, then cyclic Transfer of VP at \( vP \) also follows.

Summarizing, the “countercyclic” application of Merge (raising Subject to SpecTP after the merger of \( C \) and specless TP) cannot map \( \{C, \{T, vP\}\} \) to \( \{C, \{Subject, \{T, vP\}\}\} \). Instead, it necessarily forms \( \{Subject, \{T, vP\}\} \), which intersects with \( \{C, \{T, vP\}\} \), and neither \( \{Subject, \{T, vP\}\} \) nor \( \{C, \{T, vP\}\} \) is a term of the other. This “two-peaked tree” (or these intersecting sets) thus arises as an inevitable consequence of simplest Merge when

\(^6\) (8a,b) is the single representation of the output of “countercyclic” subject raising. That is, there aren’t two “separate workspaces” denoted by (8a,b), but rather a single representation of intersecting sets, analogous to the single phrase structure representation in (7), which is correspondingly not interpreted as two “separate” trees.
applied “counter-cyclically.” In effect, Merge can create (but not destroy) syntactic relations. It naturally guarantees that all established syntactic relations (among existing categories) remain unaltered in NS.

3. Simplification by Deletion

In NS, all syntactic relations, each established by Merge, remain unaltered, but when transferred to SM, a set-theoretic SO will be simplified by deletion. Although an SO that enters the phonological component will contain all relevant lexical features and all relevant syntactic structures (created in the course of its derivation), by SM what will remain of this SO is just a “pure” phonological representation; all but the phonological features will be deleted from it. What is particularly relevant for present concerns is the case of movement, i.e. cases where there are multiple copies of a mover. Here too, there will be simplification by deletion, specifically, only one copy of a moved element is present at SM, all other copies are deleted. In this section, we review the relevant assumptions concerning deletion in the phonological component, setting the stage for our analysis of that-trace effects.

One of the underlying and enduring assumptions in generative grammar is the principle of recoverability (Recoverability). This principle states that each SM or conceptual-intentional CI interpretable feature borne by a lexical item must be present at the relevant interface. Thus, no CI or SM interpretable feature of a lexical item can simply vanish in the course of a derivation. Thus, the lexical item cot with its phonological features (associated with /kat/) cannot be realized in SM as just /ka/ with no sign of /t/. Likewise if the overt complementizer C-that enters into a derivation, then its phonological features, represented (informally) as /that/, must be present in the phonological representation; these phonological features can’t be lost or vanish in the course of the derivation. So, Recoverability guarantees the presence (at an interface) of relevant CI and SM interpretable features. But, while by hypothesis empirically necessary (given Recoverability), mere presence at the interface is not empirically sufficient.

Another basic assumption, assigned more prominence in recent minimalist literature, is the principle of minimal computation (Minimal Computation). This principle states that an interface-interpretable lexical feature, which must be present (at least once) at the interface given Recoverability, cannot be present more than once. Chomsky (2012) notes that “universally in language, only the structurally prominent copy is pronounced,” which, he suggests “follows from another application of the third factor principle of Minimal Computation: Pronounce as little as possible.” So, for example, if a category bearing phonological features, e.g. Bill, undergoes movement, then there will be two copies of Bill in the output, but only the structurally prominent (universally the highest) copy of Bill will be phonologically realized as /Bill/. Consider (9):

(9) \{Bill, \{T, \{Bill, \{v, left\}\}\}\}

In (9), assuming that Bill moves just from SpecP to SpecTP, there would be two copies of Bill, only the highest of which is “pronounced,” hence at SM it’s /Bill left/ and not /Bill Bill left/. The relative height of such copies is calculated by the positions of their occurrences, where, following Chomsky (1995b), an occurrence of X is defined as a category to which X is merged, i.e. its derivational sister. Thus, in (9) there is one Bill (i.e. just one category, Bill, entered the derivation via a lexical array) with two occurrences determined by Bill’s sisters, namely, the two occurrences \{v, left\} and \{T, \{Bill, \{v, left\}\}\}. The latter occurrence is higher than the former occurrence since the latter occurrence contains the former occurrence as its term, hence Bill in SpecTP is the (only) copy that is phonologically realized. Under this occurrence-based calculation, the position of each copy of X is uniquely determinable by
reference to its merged sister, namely its occurrence. As the derivation proceeds, the merged sisters of the lower occurrences of X are all deleted, and only the highest copy of X will remain for SM.

To summarize, Recoverability requires that each CI and SM interpretable feature of a lexical item is present (hence present at least once) in the relevant interface representation. Minimal Computation requires that interpretable features are present at most once. Finally, the very general and completely independently necessary Full Interpretation guarantees that interface interpretation actually takes place. That is, the features that are (guaranteed to be) present in an interface representation can’t simply be ignored, but must be implemented by the interface. Putting this all together, independently of anything having to do with that-trace, the system determines that although the representation underlying, say, Bill was arrested involves [Bill was arrested Bill], the object Bill is phonologically realized once, but only once.

4. Complementizer-Trace Effects: A New Analysis

Adopting simplest Merge, along with its inevitable consequence of set-intersected representation (“doubly-rooted” tree), we argue that complementizer-trace effects, or at least the core cases discussed in the context of that-trace (see Abe 2011, Browning 1996, Chomsky 1981, 1986, Chomsky and Lasnik 1977, Ishii 2004, Lasnik and Saito 1984, 1992, Perlmuter 1971, Rizzi and Shlonsky 2007, Sobin 1987, 2002, among many others) are deducibly excluded, based on simplest Merge (as detailed above) and very general, deep principles, specifically: Recoverability and Minimal Computation. It is important to stress again that we are not adding any new principles or “technical” mechanisms of any sort, rather we are appealing to operations (simplest Merge) and deep principles (Recoverability and Minimal Computation) that are independently necessary and well-supported empirically. Our goal is to reveal that core complementizer-trace effects fall out as a natural consequence of components of the system that are “virtually conceptually necessary” or (seemingly) simplest – e.g. Merge as defined in (3i) is by hypothesis simpler than Merge defined as (3i, and ii, and iii).

4.1. Core Cases

Consider the following well-known contrast between (10) and (11):

(10) Who do you think left?

(11) * Who do you think that left?

Let’s consider the derivation of (10), key parts of the derivation of which are represented informally as in (12), with \( C_2 \) the embedded CP:
who do you think:

This tree representation informally depicts the “embedded” clause, which of course, at the “time” of its generation, has not yet become embedded. Recall that there is exactly one lexical item who associated with this derivation (who is selected exactly once from the lexicon) and this who has (at least) three copies in the embedded CP, the copies represented as who₁, who₂, who₃ (where the indices are used just for expository convenience). The lexical item who originated in SpecvP, and it was “cyclically” merged with C₁ yielding C₂ and then it was “countercyclically” merged with T₁ yielding T₂. The creation of T₂ is precisely the “two-peaked” situation we detailed above. In the course of this derivation, T₁ = {T, vP} ends up having two different sisters, namely, C_null, which T₁ was cyclically merged with, and who₂, which T₁ was “countercyclically” merged with. Now, we know that the string of copies of who are not all pronounced (the final SM implementation of the derivation is certainly not /who do you think who who left/). Thus, the merged sisters of the lower occurrences of who are all deleted in accord with Minimal Computation.

To illustrate in more technical detail, the (relevant part of the) derivation “starts” with the lower CP phase:

(13)  {C, {T, {who₁, {v, {left}}}}}}

Since the option of applying EM is now exhausted, phi feature (C to T) transmission and (then) Valuation (of phi of T and Case of who) are carried out. And, crucially for present concerns, there is IM of who to TP (to create SpecTP), and there is also IM of who to CP (to create SpecCP). These simultaneous applications of IM map (13) to both (14) and (15), (where (14) and (15) are informally represented in (12)):

(14)  {who₂, {T, {who₁, {v, {left}}}}}}

(15)  {who₃, {C, {T, {who₁, {v, {left}}}}}}}

At this point, there are three occurrences (i.e. derivational sisters) of who:
Notice that at each derivational step in (16), in fact throughout any derivation, only simple, binary Merge is employed, and sisterhood is characterized derivationally in the simplest way, as “merged with” i.e., X and Y are sisters if and only if Merge(X, Y) applied (see Epstein, Groat, Kawashima and Kitahara 1998). But, interestingly, the object \{T, \{who, \{v, \left\}\}\}\} and \{v, \left\}\}, an occurrence of who, has two (derivational) sisters: namely, the category C (see (15)) and (what is now a copy of) who (see (14)). Given Minimal Computation, the single lexical item who, bearing one and only one set of phonological features that entered the derivation, must be pronounced once and only once, at the highest copy position. But at this point we must consider with some care just how Minimal Computation is to be stated.

The intuitive idea, as we’ve seen, is that only one (the highest) copy of a moved element is realized in the phonological component (only the highest is phonologically implemented). And what this entails is that all other copies determined by the lower occurrences of the moved element must be deleted, and such deletion takes place phase-by-phase. Now we know that in the mapping from NS to SM, there is massive deletion of all but the purely phonological (i.e. SM-interpretable) features; given the strong minimalist thesis, all and only SM interpretable features are present in the representation that exits the phonological component. Thus, all syntactic structures, all semantic features, and all but the highest copy of a moved element must be deleted (i.e. removed) from the SO that undergoes this mapping.

With this in mind, consider again (14) and (15). Assuming that every copy of who represented in (14) and (15) is determined to be the sister of a lower occurrence of who, it must be deleted. Recall, occurrences of X are the (derivational) sisters of X. So, the general deletion algorithm looks at \{T, \{who, \{v, \left\}\}\}\} and \{v, \left\}\} (since they are lower occurrences of who) and says: “Delete all their sisters, i.e. delete the who copies which are the sisters of \{T, \{who, \{v, \left\}\}\}\} and of \{v, \left\}\}. Thus, the operation deletes all the lower copies of who from (14) and (15). But that’s not the end of the story.

As we pointed out above, for the EKS analysis, even though Merge is itself simple(st) and binary, with “countercyclic” IM creating SpecTP, the object \{T, \{who, \{v, \left\}\}\}\} can have two derivational sisters. In short, sister is not uniquely defined in that sisterhood is characterized derivationally. And this is required precisely because the SpecTP copy of who must be deleted (i.e. *Who do you think who left), but under the simplest deletion algorithm (deleting sisters), the C, in cases of subject extraction, is in the same position as the who-copy in SpecTP i.e. both are sisters of \{T, \{v, \null\}\}. So, given simplest Merge, and simplest deletion, whenever the general deletion algorithm deletes a copy of a subject wh
(a subject “trace”), it necessarily also deletes the local (and only the local) C – since the two are in the exact same position i.e. sister to \{T, vP\}. Overall, that-trace effects with subject (not object or adjunct) extraction follow, i.e. we can’t “say” who do you think that left for the same reason that we can’t say Bill was arrested Bill, in both cases we have “pronounced” too much in violation of Minimal Computation.

Returning to the that-trace contrast, exhibited between (10) and (11), it now follows immediately. Recall that, in the “doubly-rooted” tree or the corresponding intersecting set representation (12), C_{null} counts as a merged sister of \{T, vP\}. Now, if the two lower copies of who get deleted, the phonological features of who are in principle recoverable, because the highest copy of who (in the edge of the lower CP phase at this point of the derivation) still remains. And since C_{null} bears no phonological features, deleting the phonological features of C_{null} is vacuous, with no problem resulting from this. Thus, the deletion of the two copies of who and C_{null} does not violate Recoverability in the case of (10). Similarly, in the derivation of (11), the merged sisters of the lower occurrences of who are all deleted in accord with Minimal Computation. However, consider the corresponding “doubly-rooted” tree of (11), in which the complementizer bears phonological features, e.g. /that/, as given in (17):

(17) * who\textsubscript{4} do you think:

There are (at least) two lower occurrences of who in the embedded CP: \{T, vP\} and \{v, VP\}. The merged sisters of these occurrences include two copies of who and C_{that}. Now, if the two copies of who get deleted, the phonological features of who are in principle recoverable, because the highest copy of who (in the edge of the lower CP phase at this point of the derivation) still remains. So, the deletion of the two copies of who does not violate Recoverability. But what about the deletion of C_{that}? Recall again, under the “two-peaked” or intersecting situation, represented in (17), C_{that} counts as a merged sister of \{T, vP\}, and if the deletion rule must (indiscriminately) apply to all the merged sisters of the lower occurrences of who (understood to be the simplest application of this deletion algorithm), then C_{that} must undergo deletion, together with who\textsubscript{2} (= SpecTP) and who\textsubscript{1} (= SpecvP). In other words, the deletion of C_{that} is an inevitable consequence of the demand, imposed by Recoverability (“not too little”) and Minimal Computation (“not too much”). But C_{that} does bear phonological features (and it is the sole copy), so the deletion of C_{that} violates Recoverability. That is, that with a single set of phonological features never underwent SM interpretation of those phonological features. Thus, the only derivation that survives when performing subject extraction, is the one in which C_{null} is (randomly) chosen (and this is without look-ahead, of course).

Notice, if we analyze if as C_{if}, then the contrast between (18) and (19) follows in the same way:
(18) ? how many cars did he wonder [t_{wh} C_{if} [the mechanics fixed t_{wh}]]

(19) * how many mechanics did he wonder [t_{wh} C_{if} [t_{wh} fixed the cars]]

Observing this contrast, Chomsky (2012) notes that what (19) expresses is “a fine thought, but it has to be expressed by some circumlocution.” He speculates that there is something about language design, which poses a barrier to communication. We suggest that Recoverability and Minimal Computation constitute just such barriers.

4.2. Related Cases

As we have seen in the derivation of (11), if C_{that} is chosen, the derivation is in trouble because the deletion rule (applying to SpecTP) will “unrecoverably” delete that occupying the same position as SpecTP which itself undergoes deletion. So, if the deletion rule does not apply to SpecTP, e.g., if SpecTP is phonologically realized, then C_{that} will not be in trouble, and it will be phonologically realized as /that/. This prediction is confirmed. Consider (20), in which SpecTP is phonologically realized as /John/, and (21), in which SpecTP is phonologically realized as /there/ (Safir 1985, Chomsky 1991, 1995b:158):

(20) Who do you think [CP t_{wh} C_{that} [TP John likes t_{wh}]]

(21) How many men did John say [CP t_{how} C_{that} [TP there were t_{how} in the room]]

In each derivation, a wh-element moves directly to (embedded) SpecCP (i.e. there is no movement of wh-element to SpecTP), so the deletion rule does not apply to SpecTP; therefore, as shown above, it does not apply to the local C. Instead, SpecTP is phonologically realized, and C_{that} is phonologically realized as /that/. There is no obligatory deletion of C in such cases, and Recoverability is thereby satisfied. Similarly, this analysis captures the non-deviant status of (22):

(22) Why do you think [t_{wh} C_{that} [John likes Mary t_{wh}]]

Again, in constructing the embedded CP, the wh-element, why in this case, is never compelled to move to SpecTP, instead moving directly to SpecCP, and hence is never a derivational sister to {T, vP} and hence that escapes (obligatory but Recoverability-violating) deletion. In fact, our analysis captures the notorious subject/adjunct asymmetry in γ-marking (see Huang 1982, Lasnik and Saito 1984, 1992) but without invoking any SS versus LF asymmetry in the targets (arguments versus nonarguments) of the γ-marking algorithm.⁷ All such descriptive technicalia are prohibited under minimalist analysis rightly raising the bar on what can count as an explanation versus an unilluminating formal/technical re-description often more “complex” than the data described (as discussed by Chomsky 1995b). In addition, phenomena exhibited by data such as (23) (Lasnik and Saito 1984, 1992) are also accounted for.

(23) Who do you think [I_{wh} C_{that} [John said [I_{wh} C_{null} [I_{wh} left]]]]

⁷ Epstein (1987, 1991) sought to deduce this asymmetry from an independently motivated A versus A-bar asymmetry in indexing, independently proposed in Chomsky’s (1982) analysis of parasitic gaps (which are licensed by A-bar, but not A binding). Epstein’s (1987, 1991) analysis nonetheless resorted to levels, binding, (asymmetric) indexing, γ-features, and stipulations regarding the phonological content of that versus C_{null}.
In (23), a *wh*-element “crosses” two CP phase cycles. In the lowest CP phase cycle, the *wh*-element moves to SpecTP and SpecCP, so the deletion rule applies to SpecTP; hence, only $C_{null}$ is permitted in the embedded $C$. In the next CP phase cycle, however, the *wh*-element moves directly to SpecCP, not to SpecTP, so the deletion rule does not apply to SpecTP. SpecTP is phonologically realized as /John/, and the local $C$, namely $C_{that}$, is phonologically realized as /that/.

### 4.3. Cross Linguistic Variation of Complementizer-Trace Effects

As our deductive analysis now stands, there is a problem. As is well-known, complementizer-trace effects are parameterized, i.e. allowed in some grammars. Our analysis, based on deep principles such as Recoverability and Minimal Computation, presumably un-parameterized, would seem overly restrictive, wrongly predicting that *that*-trace effects are universally barred. Here we attempt to address this problem of how *that*-trace can possibly ever be allowed.

To begin, consider the well-known *que*-*qui* alternation. Roussou (2010) notes that it is possible to analyze *qui* as *que* + *il* (Rooryck 2000) or as *que* followed by the clitic -*i* (Taraldsen 2002). Given these possibilities, consider the following contrast (Roussou 2010:108):

(24) Qui penses-tu qui/*que est venu?
    who think-you that is come
    “Who do you think has come?”

If a pronominal form (such as *il* or -*i*) counts as a phonologically realized SpecTP at some point of a derivation, then there is no reason for the deletion rule to apply to SpecTP, and $C$+SpecTP is phonologically realized as complex /qui/.

If SpecTP is literally absent, as argued in *pro-drop* languages e.g. Italian, then the deletion rule, by definition, cannot apply to SpecTP. If there is no deletion in SpecTP, there will be no corresponding deletion of a $C$ occupying the same position as SpecTP. Given this, consider (25) (Roussou 2010:106):

(25) Chi credi che abbia telefonato?
    who think-2s that has telephoned
    “Who do you think has telephoned?”

If *chi* directly moves to (embedded) SpecCP (Rizzi 1982), i.e. not through SpecTP, then, in principle, nothing prevents $C$ from being phonologically realized as /che/. But the literal absence of SpecTP would circumvent the “two-peaked” analysis of phase-based application of Transfer. That is, under EKS (2012), “countercyclic” creation of SpecTP is precisely what triggers Transfer (to resolve the “two-peaked” situation). One possible way for us to guarantee cyclic Transfer is to pursue the line of Alexiadou and Anagnostopoulou’s (1998) analysis, where the inflection is understood to correspond to the morphological realization of EPP. Specifically, if there is (EPP satisfying) “countercyclic” verb movement to T with rich agreement forms \{V, \{T, vP\}\}, then it will induce a “two-peaked” or intersecting situation, thereby triggering Transfer, as desired for all “finite clauses” (even those lacking SpecTP).

Finally, consider the following pair, which raises a paradoxical problem: just as in (26), in (27) the deletion rule does apply to SpecTP, but $C_{that}$ is nonetheless allowed, and it is phonologically realized as /that/ (Culicover 1992, Browning 1996):
Robin met the man that Leslie said that was the mayor of the city.

Robin met the man that Leslie said that [for all intents and purposes] was the mayor of the city.

In the derivation of (26), C\text{that} must undergo deletion (together with SpecTP), but C\text{that} bears phonological features and it is the sole copy. Thus, the required deletion of C\text{that} violates Recoverability.\(^8\) By contrast, in the derivation of (27), C\text{that} is curiously allowed, given the adverbial. The obvious difference between (26) and (27) is that an adverbial expression for all intents and purposes appears between C\text{that} and SpecTP, but how does this difference receive a principled account?

Browning (1996) argues that (27) differs from (26) in the structure of the embedded CP. She assigns to (27) the following CP recursion structure, where crucially there is complementizer movement from C\text{-}to-C:

\[ [\text{CP}\ldots \text{C}\ [\text{CP}\ [\text{for all intents and purposes}\] \ [\text{t}\ C\ldots\ ]]}] \]

If Browning’s independently motivated CP recursion (or C-movement) analysis is correct, then the observed contrast, we argue, naturally follows. Consider the relevant aspects of the CP recursion structure of (27), illustrated in (29):

\[ [\text{CP} \text{t}_{\text{op}} \text{C}\text{that} \ [\text{CP}\ [\text{for all intents and purposes}\] \ [\text{t}_C \ [\text{TP} \text{t}_{\text{op}} \text{was the mayor of the city}]])]] \]

In the derivation of (27), the merged sisters of the lower occurrences of the operator OP are all deleted. There are (at least) two lower occurrences of OP in the embedded CP: \{T, vP\} and \{v, VP\}. The merged sisters of these occurrences include two copies of OP and one copy of C\text{that}. Recall that, under the “two-peaked” or intersecting situation, the lower copy of C\text{that} counts as a merged sister of \{T, vP\}, and it undergoes deletion (together with SpecTP and SpecvP). Recall, in the core, ungrammatical that-trace effect (11), this results in an obligatory but unrecoverable deletion of that. But notice, if the lower copy of C\text{that} gets deleted in (29), the phonological features of C\text{that} are recoverable, because the higher copy of C\text{that} still remains. Thus, the deletion of the lower copy of C\text{that}, forced by Minimal Computation, satisfies Recoverability, predicting that if that moves to a higher position, there will be no that-trace effect involving the departure site, exactly as in the derivation of (27), informally represented as (29).

5. Summary

We first reviewed simplest Merge and its generation of “doubly-rooted” or set-intersection structure, and then how Minimal Computation simplifies transfer to SM in accord with Recoverability. Recoverability requires C\text{that} to be phonologically realized as /that/, and Minimal Computation requires the lower copies of who to get deleted (since only one set of phonological features of who entered the derivation, so by Recoverability only one can appear at SM). These two (opposing) very general requirements, Minimal Computation (= “not too much”) and Recoverability (= “not too little”), however, appear to yield the that-trace effect applying only when the deletion rule applies to the merged sisters of \{T, vP\}, which include both (local) C\text{that} and SpecTP. As we noted at the outset, our analysis is unique in incorporating binary simplest Merge, yet it generates multiple sisters of a single category, as

---

\(^8\) We take deviance of (26) to mean that the deletion algorithm does not distinguish wh-elements such as who and a null operator OP.
discussed. A final oddity is that surprisingly, under the “two-peaked” analysis, C\textsubscript{that} and SpecTP are not in fact syntactically related, as can be seen clearly in (12) and (14)/(15), i.e. C does not c-command local SpecTP nor does local SpecTP c-command C. In current terms neither can minimally search the other, hence we have provided an account of complementizer and subject-trace effects, which recognizes no syntactic relation between the two positions. Hence, our account of that-trace makes no appeal to a syntactic relation from C to its local SpecTP. However, (a local) C and SpecTP each counts as a merged sister of \{T, vP\}, under the “two-peaked” analysis, employing simplest Merge (3i) barring both replacement (Freidin 1999) and label projection (Chomsky 2012). Given this, we argued that the core that-trace contrast, exhibited by (10) and (11), was deducible from the two very general principles Minimal Computation and Recoverability, neither of which appears to be specific to a particular construction or language, and may be in part or in full, not-specifically-linguistic principles with far more general application in computational (and/or biological, and/or physical) systems of other types that (informally) bar “random deletion and insertion.” That is, perhaps these are both good candidates for third factor laws, hence linguistic deduction from more general non-linguistic principles. But that important, fascinating and unavoidable property of all scientific inquiry – concerning the level at which explanation occurs (see Chomsky (1965:51, 2005) for bringing this aspect of science to Linguistics) – awaits further, interdisciplinary collaboration, given our ignorance concerning the broader issues at hand. We then gave brief preliminary analyses of a cross-linguistically variant range of grammatical complementizer-trace phenomena including the que-qui alternation (Roussou 2010, Rooryck 2000, Taraldsen 2001), pro-drop languages like Italian (Rizzi 1982), adverb that-trace phenomena (Culicover 1992, Browning 1996), and so-called ECP asymmetries (Huang 1982, Lasnik and Saito 1984, 1992, Safir 1985, Chomsky 1991, Rizzi and Shlonsky 2007).

References