Merge

D. Terence Langendoen, University of Arizona*

1. Introduction: The minimalist program

The minimalist program is an effort to discover the degree to which the human language faculty is determined by sensorimotor and conceptual-intentional "interface conditions" together with considerations of "virtual conceptual necessity," in particular by "general considerations of simplicity, elegance, and economy."[1] If it is entirely determined by these factors, then the human language faculty is a "perfect system, meeting external constraints as well as can be done." (Chomsky 1995a: 385-386) Chomsky calls the assumption of linguistic perfection "the strong minimalist thesis SMT," and contends that SMT should guide all linguistic theorizing until shown incorrect. (2001: 3)

The centrality of the operation merge

Chomsky also assumes that a language is a system whose "generative engine," called "narrow syntax," constructs a derivation for each choice of "lexical array." (2001: 4) Presumably other arrangements consistent with SMT are possible, but the exploration of what these might be is not at issue here. I am concerned only with the specific question of what operations narrow syntax makes use of in constructing a derivation from a lexical array, and whether their existence is consistent with SMT. Chomsky asserts that narrow syntax "has one operation that comes 'free,' in that it is required in some form for any recursive system: the operation Merge…." The condition that language is a recursive system is imposed by the conceptual-intentional interface. Merge is therefore "free," a consequence of general principles, because recursion is impossible without it. Moreover, any other operation in narrow syntax besides merge "requires empirical motivation, and is a prima facie departure from SMT." (2001: 6)

1.1. A minimalist approach to a functional phenomenon

A major concern of the minimalist program is the reduction of the computational load in carrying out a derivation. A natural extension of that concern is the reduction of the complexity of the generated objects themselves, such as their degree of embedding, without sacrificing expressive power. Syntactic transformations, as they were first formulated in generative grammar, to some extent had the property of reducing the structural complexity of the generated objects. (Miller and Chomsky 1963, Langendoen 1970) In section 3.2.1, I consider a minimalist update of this idea in which a specific form of merge reduces the structural complexity of the objects it generates.

2. Forms of merge

Merge is not a single operation, but a family of operations. To belong to the merge family, an operation must be able to yield an infinite set of objects from a finite basis. What type of merge

---

* Prepublication version of paper that appeared in *Formal Approaches to Functional Phenomena: In Honor of Eloise Jelinek*, ed. by Andrew Carnie, Mary Willie and Heidi Harley, 307-318. Amsterdam: John Benjamins.

[1] I thank Andrew Carnie, Noam Chomsky, Arnold Koslow and a mystery reviewer for helpful comments on earlier versions of this article, and Heidi Harley and Massimo Piattelli-Palmarini for their steadfast encouragement of this work.

operations are appropriate for narrow syntax depends on the nature of the infinite sets of objects it is required to create.

2.1. Set merge

Chomsky identifies the simplest form that merge can take as the formation of two-member sets:

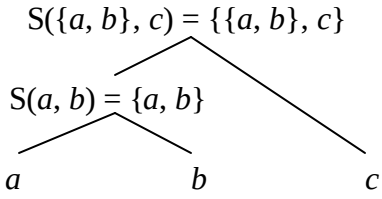
> Applied to two objects α and β, Merge forms the new object γ. What is γ? γ must be constituted somehow from the two items α and β …. The simplest object constructed from α and β is the set {α, β}, so we take γ to be at least this set. (1995a: 396)
>
> [Merge] takes two elements α, β already constructed and creates a new one consisting of the two; in the simplest case {α, β}. (2001: 6)

Chomsky calls this form of merge "set merge." (2001: 18) Further, set merge is "external" if α and β are separate objects, and "internal" (essentially "move") if one is part of the other, e.g. if β is part of α, in which case β is said to be a "copy" of its occurrence in α. He contends that external set merge builds "argument structure," whereas internal set merge builds the structures required for "scopal and discourse-related properties." (2001: 9) In (1), I show the result of externally merging *c* with the result of externally merging *a* and *b*.[2]

1) S(S(*a*, *b*), *c*) = {{*a*, *b*}, *c*}

The derivation can also be diagrammed as a tree as in (2), where the root and internal node labels are sets, not categories. Since the relation *S* is symmetric, the order of nodes in the tree is not significant.[3]

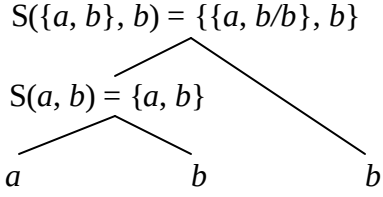
2) Tree diagram of the derivation in (1)



To distinguish external from internal set merge, I write the result of the latter as {α/β, β} if β is part of α, and {α, β/α} if α is part of β. In (3), I show the result of internally merging *b* with the result of externally merging *a* and *b*. In the corresponding tree diagram (4), the *b* that is the sister of {*a*, *b*} is the copy of the *b* that is the sister of *a*, and *b/b* is the "trace" of the original *b* which is bound by its copy.

3) S(S(*a*, *b*), *b*) = {{*a*, *b*}/*b*, *b*} = {{*a*, *b/b*}, *b*}

4) Tree diagram of the derivation in (3)



[2] The objects *a*, *b*, and *c* are not necessarily atomic (lexical).
[3] In addition, the operator *S* is commutative.

2.2. Pair merge

Chomsky (2001) maintains that a more complex form of merge is required to generate adjoined structures; he calls this operation "pair merge:"

> But it is an empirical fact that there is also an asymmetric operation of adjunction, which takes two objects β and α and forms the ordered pair <α, β>, α adjoined to β. Set-merge and pair-merge are descendants of substitution and adjunction in earlier theories. Given the basic properties of adjunction, we might intuitively think of α as attached to β on a separate plane, with β retaining all its properties on the "primary plane," the simple structure. (2001: 18) [4]

He also argues that the presence of pair merge in narrow syntax is motivated by a requirement of the conceptual-intentional interface, so that its existence is consistent with SMT:

> [R]ichness of expressive power requires an operation of predicate composition: that is not provided by set-Merge.... But it is the essential semantic contribution of pair merge. (2001: 18)

I represent the pair merge of β adjoined to α as $P(\alpha, \beta) = \langle\alpha, \beta\rangle$.[5] The discussion of the use of pair merge for generating adjunct structures continues below in section 3.

2.3. Inadequacy of set merge to represent argument structure

When the objects α and β are merged, one of them is the head. But the result of their set merge doesn't indicate which one, since $\{\alpha, \beta\} = \{\beta, \alpha\}$. If α is lexical and β phrasal (i.e., if β but not α is of the form $\{\gamma, \delta\}$), then from the configuration $\{\alpha, \beta\} = \{\alpha, \{\gamma, \delta\}\}$, one might infer that α is the head and β its complement (COMP). However if both are lexical, one must further specify which is the head. Similarly, if $\alpha = \{\gamma, \{\delta, \eta\}\}$ and β is lexical, then either β is a specifier (SPEC) of γ, or α is the COMP of β, and one must further specify which.

Chomsky (1995a) solves this problem by proposing a variety of merge that "labels" the result of the operation. The passage from Chomsky (1995a) quoted in section 2.1 continues as follows:

> Does this [= set merge] suffice? Output [= interface] conditions dictate otherwise…. 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.... If α projects [is the head], then $\gamma = \{\alpha, \{\alpha, \beta\}\}$. (1995a: 397)

Since the construct $\{\alpha, \{\alpha, \beta\}\}$ is the standard way of representing the ordered pair $\langle\alpha, \beta\rangle$ set-theoretically (Enderton 1972: 6), the use of labels to represent argument structure conflicts with the use of pair merge to represent adjunct structure. Further, Collins (2001) argues against the use of labels in narrow syntax, showing that elements combine properly under set merge using information independently required in the lexical array. However, Collins's arguments do not address the inadequacy of the output of set merge to *represent* argument structure. Another form of merge must be found to do so.

---

[4] Technically, the pair merge relation is antisymmetric, not asymmetric, since $\langle\alpha, \beta\rangle = \langle\beta, \alpha\rangle$ if and only if $\alpha = \beta$. In addition, the pair merge operation is noncommutative. Pair merge is comparable to the LIST operator of Common LISP, since (list 'α 'β) = (α β), which is the ordered pair $\langle\alpha, \beta\rangle$ in LISP notation; see Touretzky (1997: 164).

[5] The adjunct and the head are in the opposite order from Chomsky's definition above, for reasons that will become clear below.

2.4. List merge

The simplest and most elegant and economic way of representing argument structure is by listing the predicate and its arguments in a sequence. For example, Mostowski (1979) proposes that while a predicate in general is a function on infinite sequences of objects, every propositional function has finite "support." Koslow remarks:

> It is difficult to say whether this practice was inspired by the "observation" that a natural language ... has verbs with a natural number of places or whether the influence is just the reverse, from logical practice to the analysis of [natural language] verbs and predicates. (1992: 181)

Whatever the direction of influence, the requirement of the conceptual-intentional interface for argument structure, together with "general considerations of simplicity, elegance, and economy" dictates that the proper representation of argument structure in narrow syntax should be in the form of a list, an ordered sequence of objects.

In keeping with logical practice, let us suppose that the predicate occurs first in each list, followed its arguments in sequence, first its COMP and then its SPECs. Pair merge can represent the first step in constructing such lists, since if α is a predicate and β its COMP (its first argument), then $P(\alpha, \beta) = <\alpha, \beta>$. However, if γ is the first SPEC of α (its second argument), then pair merge gives the wrong result on the next step, since $P(<\alpha, \beta>, \gamma) = <<\alpha, \beta>, \gamma>$, not the desired $<\alpha, \beta, \gamma>$.

To get the desired result, I propose a type of merge called "list merge" *L*, according to which $L(\alpha, \beta) = P(\alpha, \beta) = <\alpha, \beta>$ if α is atomic, and $L(<\alpha, ...>, \beta) = <\alpha, ..., \beta>$, an "ordered *n*-tuple" (Enderton 1972: 6), a list of *n* items.[6] In the ordered triple $<\alpha, \beta, \gamma>$, α is the head (the predicate), β is its COMP (its first argument), and γ its SPEC (its second argument). Further application of list merge yields additional SPECs, precisely as Chomsky (1995a: 432; 2001: 6) argues for. For example $L(<\alpha, \beta, \gamma>, \delta) = <\alpha, \beta, \gamma, \delta>$, an "ordered quadruple" in which γ and δ are SPECs of α. Finally, since the output of list merge is less structured than that of pair merge, list merge must, like set merge, be considered conceptually simpler than pair merge.[7]

2.5. Another inadequacy of set merge

According to Chomsky, application of internal merge of β with α, where β is part of α, must result in the creation of a new SPEC:

> [D]isplacement [of β] from within α [must] be to the edge of α, yielding a new SPEC. (2001: 9)

To get this result using set merge, it must be explicitly stipulated that β is not the head of the resulting structure.[8] However with list merge, the result follows immediately from the fact that the internal list merge of β with α can only be expressed as $L(<\alpha, ...\beta...>, \beta) = <\alpha, ...\beta/\beta, ... \beta>$, and in that structure β is a SPEC of γ.

---

[6] List merge is comparable to the Common LISP operator APPEND, since (append '(α ...) 'β) = (α ... β), which is the LISP equivalent of $<\alpha, ..., \beta>$.

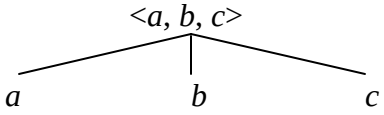
[7] A similar argument shows that the list-merge counterpart to set merge, call it "union" merge *U*, where $U(\{\alpha, \beta\}, \gamma) = \{\alpha, \beta, \gamma\}$, is conceptually simpler than set merge; see also Chomsky (2001: 14, n. 50).

[8] Chomsky (2001: 9) derives this requirement from what he calls the "extension condition" on set merge. No such condition is required for list merge.

2.6. A possible objection to list merge

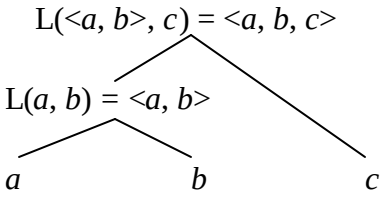
Suppose we grant that in the structure (2) generated by set merge, *a* is the head, *b* is its COMP, and *c* is its SPEC. In that structure, *c* asymmetrically c-commands *b*, and that relation can be used in an account of why an anaphor occurring in the position of *b* can be bound by an antecedent occurring in the position of *c*, whereas an anaphor occurring in the position of *c* cannot be bound by an antecedent occurring in the position of *b*. On the other hand, in the structure (5) of the syntactic object <*a*, *b*, *c*> generated by list merge, *b* and *c* c-command each other, and some other explanation for the asymmetry of the antecedent-anaphor relation is required.

5) Tree diagram of the structure of the ordered triple <*a*, *b*, *c*>



In fact, two different explanations are possible. One is that the asymmetry of the antecedent-anaphor relation is based on the positions of *b* and *c* in the list, rather than on c-command. The other is that in the tree diagram of the *derivation* of the ordered triple shown in (6), *c* does asymmetrically c-command *b*. Therefore, there can be no objection based on the c-command relation to the use of list merge to account for argument structure.

6) Tree diagram of the derivation of the ordered triple <*a*, *b*, *c*>



3. Using pair merge for generating adjunct structures

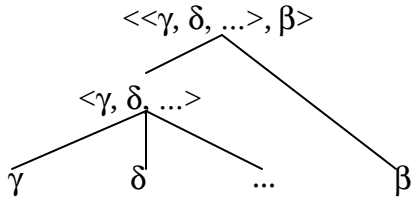
Using list merge for generating argument structure leaves pair merge available for generating adjunct structure, as Chomsky (2001: 15) proposed; see section 2.2. The essential property of adjunction is the fact that if β is adjoined to α, then α as a whole is the head of the structure, not the head of α. That is, the adjunction of β to α = <γ, δ, ...> is <<γ, δ, ...>, β>, where the analysis of α as a list prevents β from being construed as a SPEC of γ, the head of α. To obtain this structure however, we need only apply pair merge, since P(<γ, δ, ... >, β) = <<γ, δ, ...>, β>.

The contrast between adjoining β to α = <γ, δ, ...> and adding β as a SPEC of γ, the head of α, can be seen by comparing (7) and (8), and the resulting trees in (9) and (10), where the ordering of nodes is significant, unlike in (2) and (4).

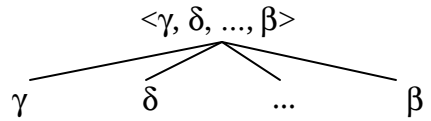
7) P(α, β) = P(<γ, δ, ...>, β) = <<γ, δ, ...>, β>

8) L(α, β) = L(<γ, δ, ...>, β) = <γ, δ, ..., β>

9) Tree diagram of the pair merge of $\alpha = \langle\gamma, \delta, ...\rangle$ with β; i.e. the adjunction of β to α

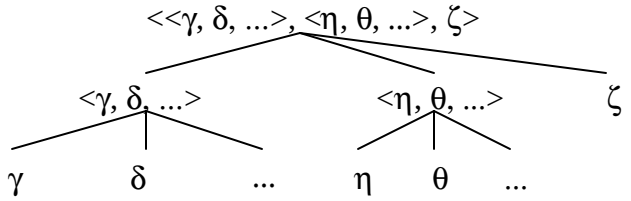


10) Tree diagram of the list merge of $\alpha = \langle\gamma, \delta, ...\rangle$ with β; i.e., the addition of β as a specifier of γ, the head of α

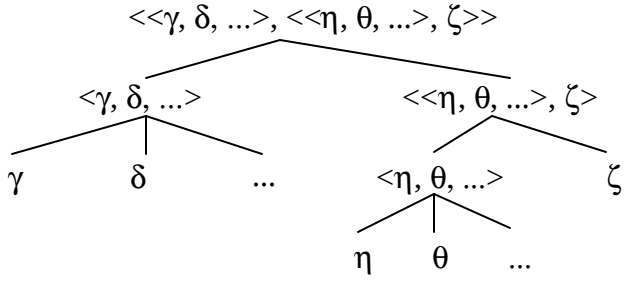


Subsequent addition of elements to an adjunct structure are carried out by applications of list merge as in (11).[9] As a result, adjuncts can be added to a head, in principle without limit, just as SPECs can. The formula in (11) shows "high" attachment of the second adjunct ζ. On the other hand, the adjunction of ζ to $\beta = \langle\eta, \theta, ...\rangle$ results in "low" attachment, as in (12).[10]

11) $L(P(\langle\gamma, \delta, ...\rangle, \langle\eta, \theta, ...\rangle), \zeta) = L(\langle\langle\gamma,\delta, ...\rangle, \langle\eta, \theta, ...\rangle\rangle, \zeta) = \langle\langle\gamma,\delta, ...\rangle, \langle\eta, \theta, ...\rangle, \zeta\rangle$



12) $P(\langle\gamma, \delta, ...\rangle, P(\langle\eta, \theta, ...\rangle, \zeta)) = P(\langle\gamma, \delta, ...\rangle, \langle\langle\eta, \theta, ...\rangle\rangle, \zeta\rangle) =$
$\langle\langle\gamma, \delta, ...\rangle, \langle\langle\eta, \theta, ...\rangle, \zeta\rangle\rangle$



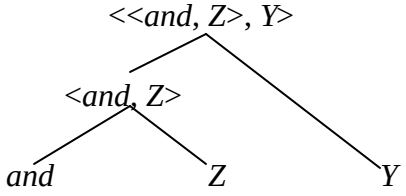
[9] If the order in which adjuncts are added is not linguistically significant, then additional adjuncts should be added by union merge; see note 7.

[10] The structure in (11) represents the interpretation of a phrase like *the box behind the chest next to the crate* in which the box is both behind the chest and next to the crate, whereas the structure in (12) represents the interpretation of that phrase in which the box is behind the chest and the chest is next to the crate.

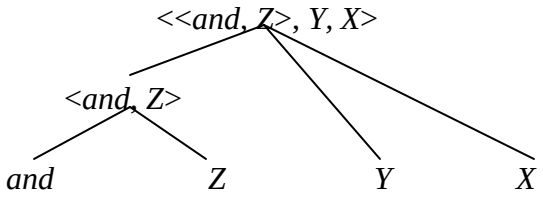
3.1. Analysis of coordinate structure using pair merge

Coordinate structure is distinct from argument structure, so it should not be constructed entirely by list merge (i.e., it is not the "functional projection" of coordination). Asyndetic coordination (coordination without the use of a coordinating particle) can be handled by the combination of pair and list merge as described above. Similarly, the analysis of syndetic coordination requires the use of pair merge of the first member to the result of the list merge of the coordinating particle with the second member. So for example the structure of the coordinate phrase *Y and Z* is given by (13), and of the coordinate phrase *X, Y and Z* in (14); in both structures <*and, Z*> is the head.

13) Structure of the coordinate phrase *Y and Z*



14) Structure of the coordinate phrase *X, Y and Z*



3.2. Adjunction and internal merge

Since internal merge is "freely available," we may expect to find applications of internal pair merge in the formation of adjunct structures; its absence, in Chomsky's words, "would be an imperfection." (2001: 8) Chomsky (2001) does not consider the possibility of internal pair merge, but only what happens when internal set merge (our internal list merge) applies to an adjunct structure as a whole.

In the present framework, two types of internal pair merges are possible: (i) adjunction of β to α, where $\alpha = \langle\gamma, \delta, ...\rangle$ is part of β, as in (15), which I call "internal head pair merge;"[11] and (ii) adjunction of β to α, where $\beta = \langle\gamma, \delta\rangle$ is part of α, as in (16), which I call "internal adjunct pair merge."[12]

15) $P(\langle\gamma, \delta, ...\rangle, \beta/\langle\gamma, \delta, ...\rangle) = \langle\langle\gamma, \delta, ...\rangle, \beta/\langle\gamma, \delta, ...\rangle\rangle$

16) $P(\alpha/\langle\gamma, \delta, ...\rangle, \langle\gamma, \delta, ...\rangle) = \langle\alpha/\langle\gamma, \delta, ...\rangle , \langle\gamma, \delta, ...\rangle\rangle$

---

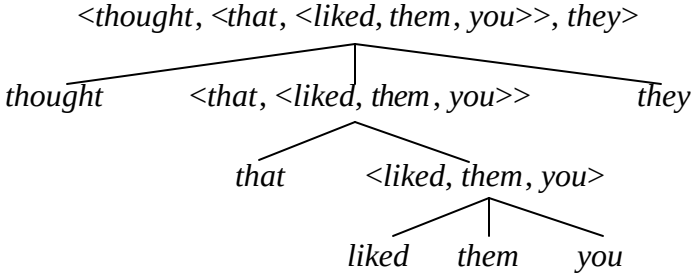
[11] Internal pair merge, unlike internal list merge, is capable of copying into the head position, since pair merge is not an append-type operation.

[12] Head-to-head movement, the adjunction of a lower head to a higher head, is only possible if the lower head is a list and not an atom. But lower list heads can only be constructed by adjunction in the first place, which means that head-to-head movement cannot be part of narrow syntax, as Chomsky (1995b: ch. 4) has proposed; see also Chomsky (2001: 23, n. 69).

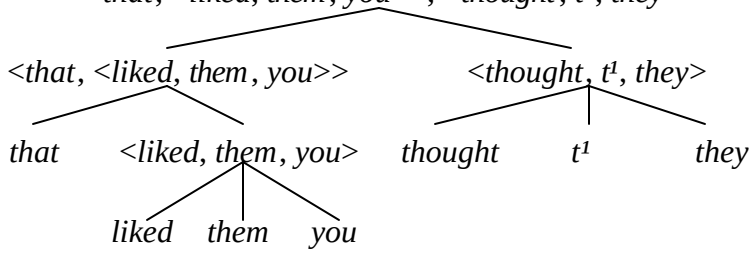
3.2.1. Internal head pair merge

Application of internal head pair merge (adjunction of β to α, where α is part of β) results in the "readjustment" of structure, as in the application of "readjustment rules." (Langendoen 1975) For example in (15), let β be the structure in (17), which is the argument structure of *they thought that you liked them*, and α = <*that*, <*liked*, *them*, *you*>> (so that γ = *that* and δ = <*liked*, *them*, *you*>). Application of internal head pair merge in (18) results in the "readjusted" structure in which <*thought*, *t*¹, *they*> (where *t*¹ = α/α is the trace of <*that*, <*liked*, *them*, *you*>>) is adjoined to <*that*, <*liked*, *them*, *you*>>, and which is realized as the "pseudocoordinate" pattern *they thought t*¹ | *that you liked them*, where the vertical bar indicates an intonation break.

17) <*thought*, <*that*, <*liked*, *them*, *you*>>, *they*>



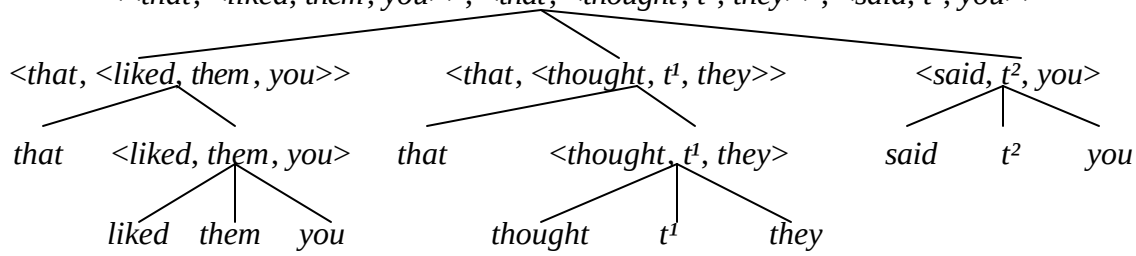
18) <<*that*, <*liked*, *them*, *you*>>, <*thought*, *t*¹, *they*>>



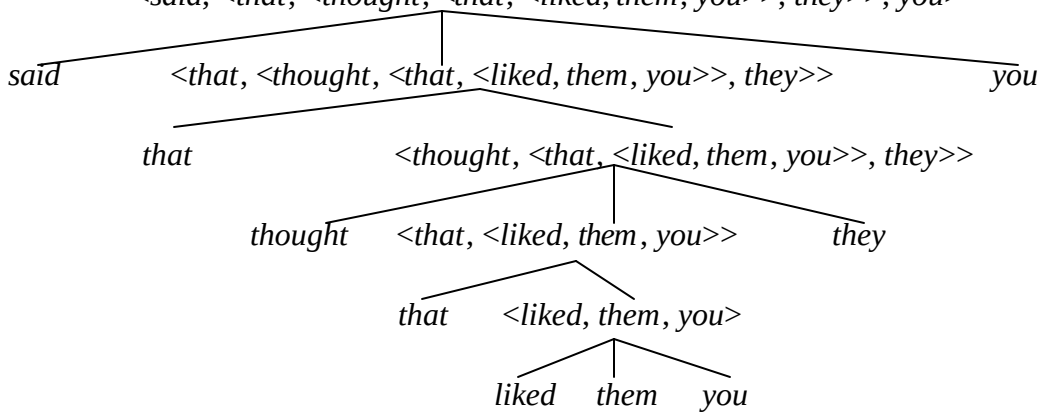
Although <*thought*, *t*¹, *they* > is an adjunct in (18), it can be the locus of further external list merge operations. For example, subsequent external list merges of the complementizer *that*, the verb *say*, and the pronoun *you* results in the structure (19). To this structure, internal head list merge applies, attaching <*said*, *t*², *you*> as a second adjunct to <*that*, < *liked*, *them*, *you*>>, resulting in (20), where *t*² is the trace of <*that*, <*thought*, *t*¹, *they*>>. The realization of this structure is the pseudocoordinate pattern *you said t*² | *that they thought t*¹ | *that you liked them*. As a result, the degree 4 nesting of the nonreadjusted argument structure (21) is reduced to degree 2. With each additional recursion, the degree of nesting of the nonreadjusted structures increases by 2, but that of the readjusted structures remains at 2. Such structures, unlike those generated by list merge alone, can be processed by a finite-state device, thus providing a minimalist account of a functional phenomenon, as promised in section 1.1.

19) *<<that, <liked, them, you>>, <said, <that, <thought, t¹, they>>, you>>*

20) *<<that, <liked, them, you>>, <that, <thought, t¹, they>>, <said, t², you>>*



21) *<said, <that, <thought, <that, <liked, them, you>>, they>>, you>*



3.2.2. Internal adjunct pair merge

Internal adjunct pair merge (adjunction of β to α, where β is part of α) is an alternative mode of analysis for A-bar movements, instead of substitution (movement to the SPEC of a higher "functional projection"). For example, given the argument structure of *who did you like* in (22), simple application of the internal adjunct pair merge in (23) (i.e., without any functional projections) results in the adjunction of *who* to the main clause, where t^w is the trace of *who*, and where I ignore for ease of exposition the effects of subject-auxiliary inversion and *do*-support.

22) *<liked, who, you>*

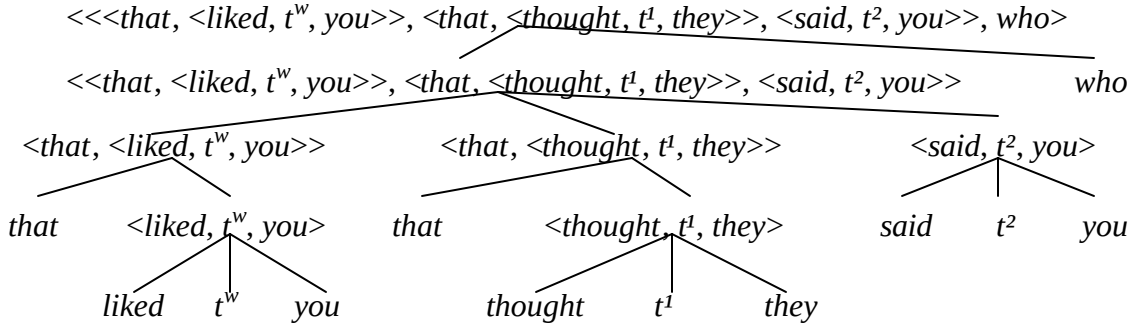
23) P(*<liked, who, you>, who*) = *<<liked, t^w, you>, who>*

If such "*wh*-movement" is attempted from within a multiply embedded complement clause, as in (24), it may be supposed that the target and the goal occurrences of *who* are separated by too many list edges (a kind of "subjacency violation"), requiring intermediate applications of such movement. An alternative is that the structure of the first argument in (24) first undergoes internal adjunct merge as in (25), so that the target and the goal are separated by at most two list edges. Then the application of internal adjunct pair merge of *who* results in the structure (26), whose realization is *who did you say t² | that they thought t¹ | that you liked t^w*.

24) *P(<said, <that, <thought, <that, <liked, who, you>>, they>>, you>, who)

25) P(*<<that, <liked, who, you>>, <that, <thought, t¹, they>>, <said, t², you>>, who*)

26) <<<*that*, <*liked*, t^w, *you*>>, <*that*, <*thought*, t^1, *they*>>, <*said*, t^2, *you*>>, *who*>



This use of internal adjunct pair merge to generate "long distance" A-bar movements suggests that all such cases require the prior application of internal head pair merge, so as to flatten the structure sufficiently to avoid subjacency violations, but further consideration of this possibility lies outside the scope of this paper.

4. Conclusion

I propose that recursive operations are ranked according to their "simplicity, elegance, and economy," with set merge ranked highest, list merge second, and pair merge third (see section 2.3). Then if set merge were adequate for the purposes of narrow syntax (i.e. if it provided the basis for interpretation of linguistic objects at the interfaces), SMT would compel us to use it. However, set merge is not adequate for any such purpose, so that list merge emerges as the next candidate. I find that list merge is adequate for some of the purposes that Chomsky (2001) proposes for set merge, specifically accounting for argument structure. However, list merge (like set merge) is inadequate for adjunct structures, so that pair merge must be adopted for those cases. The presence of pair merge in narrow syntax then makes available a new range of mechanisms for narrow syntax, internal head and adjunct pair merge, whose explanatory power looks quite promising.

References

Chomsky, Noam. 1995a. Bare phrase structure. Gert Webelhuth, ed., *Government and Binding Theory and the Minimalist Program,* 383-439. Oxford: Blackwell.

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

Chomsky, Noam. 2001. Beyond explanatory adequacy. *MIT Occasional Papers in Linguistics* 20.

Collins, Chris. 2001. Eliminating labels. *MIT Occasional Papers in Linguistics* 20.

Enderton, Herbert B. 1972. *A Mathematical Introduction to Logic*. New York: Academic Press.

Koslow, Arnold. 1992. *A Structuralist Theory of Logic*. Cambridge, UK: Cambridge University Press.

Langendoen, D. Terence 1970. The accessibility of deep structure. *Readings in English Transformational Grammar*, ed. by Roderick A. Jacobs and Peter S. Rosenbaum, 99-104. Waltham, MA: Ginn and Company.

Langendoen, D. Terence 1975. Finite-state parsing of phrase-structure languages and the status of readjustment rules in grammar. *Linguistic Inquiry* 6: 533-554.

Miller, George A., and Chomsky, Noam. 1963. Finitary models of language users. *Handbook of Mathematical Psychology* Volume 2, ed. by R. Duncan Luce, Robert R. Bush and Eugene Galanter, 419-491. New York: John Wiley and Sons.

Mostowski, Andrzej. 1979. On a generalization of quantifiers. *Foundational Studies: Selected Works, Volume II*. Amsterdam: North-Holland.

Touretzky, David S. 1997. *Common LISP: A Gentle Introduction to Symbolic Computation*. Redwood City, CA: Benjamin/Cummings.