Some Notes on External Remerge*

Daeho Chung
(Hanyang University)

To constrain external remerge (or multi-dominance) including the so-called quirky cases of remerge, de Vries (2009: 357) proposes a condition called Root Condition (RC), which states that if α and β are selected as input for merge, then α or β (or both) must be a root. He also formalizes RC as No Proliferation of Roots Condition (NPRC), which commands merge not to increase the number of roots. This paper, however, disproves the formalization of RC as NPRC due to some disparity between the two and then shows that neither RC nor NPRC properly excludes quirky remerge cases without stipulating an unmotivated order restriction on merge. Instead the current work suggests a process-based explanation under some plausible assumptions on the economy measure of merge and on process load variations depending on the work space arrangement at a relevant point of derivation.

Keywords: merge, remerge, extension condition, work space, Root Condition (RC), No Proliferation of Roots Condition (NPRC)

1. Introduction: Restrictions on Remerge and Root Condition (de Vries 2009)

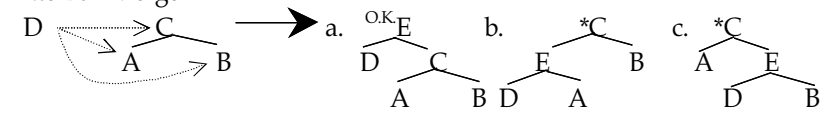
Merge, especially external merge, is a major and indispensable syntactic operation if syntax is generative. As the operation traditionally called movement is also recast as a complex operation including merge, or more precisely internal remerge, one stand-out property that merge displays is that syntactic terms can

* I would like to thank Mark de Vries, Myung-Kwan Park, and Keun Young Shin for the valuable discussions that I had with them. I also thank three anonymous reviewers for *SGG* for constructive comments and suggestions which helped me reshape the paper, although I could not incorporate them all in this paper. All remaining errors are solely my own.

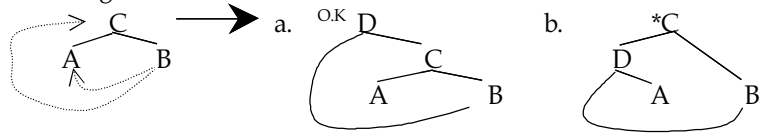
be recycled in the process of structure building. Given the legitimacy of internal remerge, a natural question that arises is whether external remerge is also available as a syntactic operation. In other words, can a term merge with another term outside of the current work space? Various linguists claim or support the existence of such an operation, although they give this operation various names, as pointed out by de Vries (2009, 349): 'interarborial movement' (Bobaljik 1995, Bobaljik & Brown 1997), 'sideward movement' (Nunes 2001, 2004), 'multidominance/multidomination/multiple dominance' (McCawley 1982, Ojeda 1987, Blevins 1990, Wilder 1999, 2008, Chen-Main 2006, Johnson 2007, Bachrach & Katzir 2009), 'sharing' (Guimarães 2004, Chung 2004, de Vries 2005, Gracanin-Yuksek 2007), 'grafting' (van Riemsdijk 1998, 2006), and 'parallel merge' (Citko 2005). (See Carnie 2008, chapter 10 for a brief overview. See also Park 2010, who proposes an interesting account of both so-called shared and non-shared right node raising structures in terms of external remerge.)

How is then external remerge constrained? Before tackling this question, let us first consider how merge in general is constrained. As far as external merge and internal merge (in a work space) are concerned, Extension Condition (Chomsky 1995: 190, 327) suffices to constrain the merge operation: As a structure-building (not structure-changing) process, merge takes place at the edge only and no counter-cyclic tampering is permitted. Consider the following derivations (adapted from de Vries 2009: 354-356, his (14)-(16)).

(1) External Merge



(2) Internal Merge

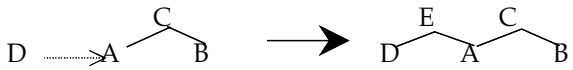


(1) exemplifies instances of external merge in which an (external) element merges with a previously built structure. In (1a) D is merged to the edge of the previously built structure C and the result is legitimate. In contrast, (1b) and (1c), where D is counter-cyclically merged with an internal element, A in (1b) and B in (1c), violate Extension Condition (or No Tampering Condition). (2) exemplifies instances of internal merge in which an element within a previously built

structure remerges, i.e., merges again. In (2a), B within C is taken and merges with C, the root at the relevant point of derivation. A new structure being built due to this remerge process, Extension Condition (or No Tampering Condition) is satisfied. In contrast, (2b), where B is taken within C and merges with A, another element within C, i.e., a non-root element, violates Extension Condition (and No Tampering Condition), accounting for the illegitimacy of the newly built structure.

Now turn to the cases of external remerge. Whether external remerge satisfies Extension Condition or not seems to depend on the choice of syntactic terms to be merged. Consider the following typical example of external remerge, where an (external) element merges with another element within a previously built structure:

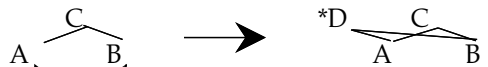
(3) External Remerge (de Vries 2009: 356, his (16c))



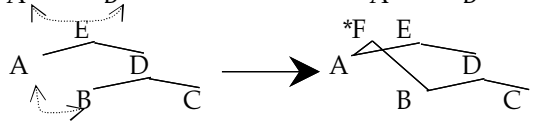
C remains unchanged, apparently violating Extension Condition. However, E is newly built due to the remerge operation, satisfying Extension Condition.

Now consider the cases where elements to be merged are both non-root terms, as in the following structures:

(4) a.



b.



Since the newly built structure does not tamper the previously built one, the remerge operations in (4) do seem to satisfy Extension Condition. As such free remerge comes into the picture, Extension Condition does not suffice.

To constrain merge in general (including external remerge), de Vries (2009: 357) proposes Root Condition (RC) in (5) below:

(5) **Root Condition (RC)** (= de Vries 2009, his (19))
If α and β are selected as input for merge, then α or β (or both) must be a root.

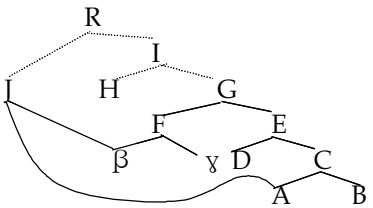
RC appears to well explain the contrast between (3) and (4). (3) satisfies RC since D is a root, although A is not, when the two combine. (4a) and (4b), however, violate RC since the two terms, A and B, are non-roots when they combine for the second time.

de Vries (2009) claims that RC excludes what he calls quirky internal remerge and quirk external remerge as well, as in (6) and (7) respectively, (cited from de Vries 2009, 356-357, his (17) and (18),) where grey lines indicate yet-to-be merge operations and exclamation marks indicate problematic instances of merge:[1]

(6) (Merge (β, ɣ) → F)

Merge (A, B) → C
Merge (D, C) → E
Merge (F, E) → G
! Merge (A, β) → J

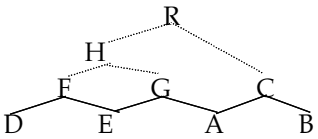
Merge (H, G) → I
Merge (J, I) → R



(7) (Merge (D, E) → F)

Merge (A, B) → C
! Merge (E, A) → G

Merge (F, G) → H
Merge (H, C) → R



Instances of merge with an exclamation mark are problematic since two terms to be merged are already used in the previous derivation, violating RC. In (6), J results from merging β and A, both of which are non-roots since they are previously used. Likewise in (7), G results from merging E and A, both of which are non-roots.

de Vries (2009) further claims that RC can be formalized as follows:

[1] de Vries (2009) calls derivations like (6) 'quirky internal merge': 'internal', since both input elements for J come from the previously built structure (pp 356-357); 'quirky', because both input elements are non-root (p 358). As expected, quirky external merge will then involve two non-root inputs from two separate structures (or work spaces).

(8) **No Proliferation of Roots Condition** (NPRC, de Vries 2009: 358, his (20))
If the derivation proceeds from stage i to i+1 through merge (α, β) → γ, then |{x∈{α, β, γ}: x is a root at stage i+1}| ≤ |{x∈{α, β}: x is a root at stage i}|

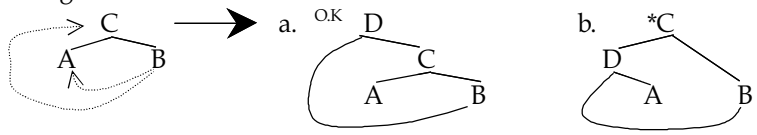
To state informally, merge should not increase the number of roots. NPRC seems to rule out the derivations in (6) and (7). Given the derivation order in (6), at the stage where J is produced, merge increases the number of roots by one (from one to two). Similarly in (7), at the stage where G is produced, merge increases the number of roots by one (from two to three).

In this paper, however, it will be shown that RC and NPRC are not co-extensional, casting doubt on the validity of the formalization of RC as NPRC, and that neither RC nor NPRC properly excludes the quirky cases of remerge as in (6) and (7). This work instead suggests a process-based account under some plausible assumptions on the economy measure of merge (or remerge) and on the process load variations depending on the work space arrangement.

2. Can RC be formalized as NPRC?

In this section, it will be first pointed out that de Vries' (2009) formalization of RC as NPRC is incorrect since the two are not co-extensional. Notice first that RC is based on the root vs. non-root property of the elements to be merged, while NPRC compares the numbers of roots before and after a merge operation. Thus, the two make different predictions, especially as to non-cyclic movement. For example, consider the merge operations in (2), repeated below:

(2) Internal Merge



(2a) satisfies both RC and NPRC. At the relevant point of merge, B is a non-root but C is a root node, satisfying RC. The number of roots remains the same (from one to one), satisfying NPRC as well. Now consider the illegitimate derivation in (2b). The non-cyclic merge operation violates RC, but not NPRC. D results from merging two non-roots, violating RC. Notice, however, that the

merge operation does not increase the number of roots, satisfying NPRC. The number remains one since D is embedded under C.

Obviously, non-cyclic operations like the one in (2b) can be ruled out by Chomsky's (1995) Extension Condition on merge (or Chomsky's 2008 NTC, no tampering condition). Although de Vries (2009) does not explicitly propose to replace Extension Condition by RC/NPRC, he seems to intend to do so, when he (2009: 358) claims that RC can also be interpreted as a requirement that 'derivation is always "active at the top"' (p 358). The illegitimacy of cases like (2b) indicates, however, that RC, especially when it is equated with NPRC, does not perfectly replace Extension Condition.

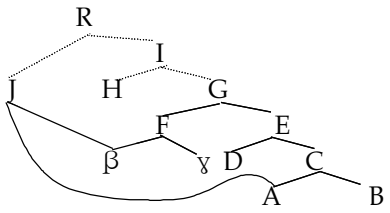
3. Neither RC nor NPRC Excludes Quirky Remerge Structures

In this section it will be shown that neither RC nor NPRC appropriately excludes the so-called quirky internal or external remerge structures.[2] Consider the structures in (6) and (7), repeated below:

(6) (Merge (β, ɣ) → F)

Merge (A, B) → C
Merge (D, C) → E
Merge (F, E) → G
! Merge (A, β) → J

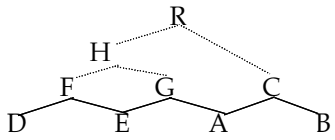
Merge (H, G) → I
Merge (J, I) → R



(7) (Merge (D, E) → F)

Merge (A, B) → C
! Merge (E, A) → G

Merge (F, G) → H
Merge (H, C) → R



What de Vries (2009) crucially assumes to rule out such quirky cases of internal

[2] The distinction between quirky 'internal' vs. 'external' remerge turns out to be meaningless, if the discussion in this section is on the right track. What they have in common is that there are two (or possibly more) nodes are shared at the relevant stage of derivation.

or external remerge is that the problematic remerge applies latest at the relevant stage of derivation. In (6), for example, Merge (A, β) → J follows Merge (β, ɣ) → F and Merge (A, B) → C. Similarly in (7), Merge (E, A) → G follows Merge (D, E) → F and Merge (A, B) → C.

Notice, however, that such an order restriction on external remerge cases comes from nowhere. When the 'problematic' merge operation with an exclamation mark applies prior to at least one of the other two, the derivation should go through without violating RC or NPRC. All the following orders of derivations should be legitimate as to RC or NPRC as well: JFC, JCF, FJC, or CJF in (6); GFC, GCF, FGC, or CGF in (7).[3] For example, suppose that, in (6), J is first formed and then F and C follow. Then the derivation will satisfy both RC and NPRC, as the following table illustrates:

(9)

Derivation Order	Root vs. non-root status of inputs	RC	Root numbers before and after merge	NPRC
a. Merge (A, β) → J	(R, R)	√	2 → 1	√
b. Merge (β, ɣ) → F	(nR, R)	√	1 → 1	√
c. Merge (A, B) → C	(nR, R)	√	1 → 1	√
d. Merge (D, C) → E	(R, R)	√	2 → 1	√
e. Merge (F, E) → G	(R, R)	√	2 → 1	√
f. Merge (H, G) → I	(R, R)	√	2 → 1	√
g. Merge (J, I) → R	(R, R)	√	2 → 1	√

As shown in the table, all the merge operations in this order of derivation combine two roots or a non-root plus a root. There is no instance in which two non-roots merge, satisfying RC. All the operations either reduce the number of the roots by one (operations a, d, e, f and g), or maintain it (operations in b and c)), satisfying NPRC. Thus, the derivation in this order poses no problem with respect to RC or NPRC. As the readers can verify, orders like JCF, FJC, or

[3] Mark de Vries (p.c.) admits that his system faces such a problem.

CJF in (6), and GFC, GCF, FGC, and CGF in (7) will also satisfy RC and NPRC.

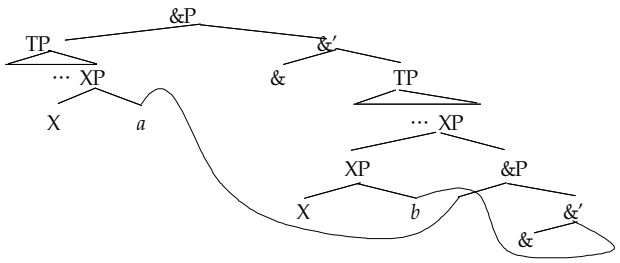
To sum up, de Vries' (2009) RC or NPRC works only when a special order of derivation is stipulated. Since such an order restriction comes from nowhere, however, neither RC nor NPRC appropriately rules out the quirky cases of remerge.

4. A Process-based Account of Restrictions on Merge

This section suggests that the restrictions on external remerge should be accounted for by process considerations under some plausible assumptions on economy measure of merge and on the process load variations depending on the work space configuration. Before directly undertaking the quirky remerge cases, let us first consider the so-called right node raising (RNR) construction like (10) below. The RNR construction instantiates a simple remerge (simple parallel merge or multi-dominance) structure under the multi-dominance analysis of the construction (McCawley 1982, Wilder 1999, 2008, Abels 2004, Chung 2004, among others), as roughly represented in (11) below. [4]

[4] As pointed out by an anonymous reviewer, (11) is not the unique structure for RNR constructions. In his series of paper, for example, Park (2005, 2007, 2009, 2010) proposes a new version of the multi-dominance syntax of the RNR construction, called a 'midway conjunction analysis': the rightmost element in each conjunct is taken and then remerged respectively into the specifier position and the complement position of a base generated &P, which itself is adjoined to the second (or more generally final) conjunct, as schematically represented in (i) below:

(i)



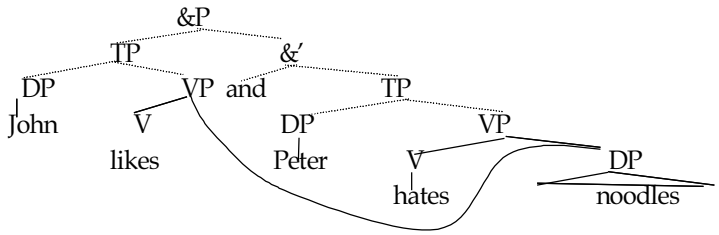
The midway conjunction analysis is primarily motivated by the so-called non-shared RNR constructions, as in (ii) below:

(ii) a. John loves and Mary hates oysters and clams, respectively. (Postal 1998: 134)
b. Greg captured and Lucile trained 312 frogs all together. (Postal 1998: 137)

Park claims that the so-called shared RNR constructions as in (10) result when a=b in (i) and some morphological adjustments take place within the &P adjoined to the second (or final) conjunct.

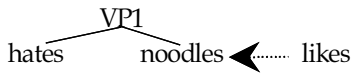
(10) John likes, and Peter hates noodles.

(11)



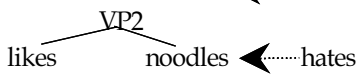
After DP *noodles* is built up, it merges with verb *hates* and verb *likes*. There are several possibilities to form such a multi-dominance structure. One possibility is that *hates* and the DP first merge, forming a VP, and then the DP remerges, i.e., merges again, with *likes*, forming another VP, as in (12) below. Another possibility is to reverse the order: the DP first merges with *likes*, forming a VP, and then remerges with *hates*, forming another VP, as in (13). A third possibility is to merge the DP with both of the verbs simultaneously, forming two VPs at the same time, as in (14).

(12) a.

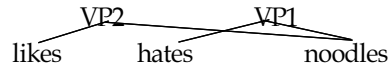


b.

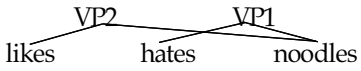
(13) a.



b.



(14)



The three options produce the same final structure, but they differ in the timing of 'remerge'. Derivations in (12) and (13) include a process, as a separate step of derivation, in which DP *noodles* is remerged. In such derivations, since the DP is a non-root term when a second merge is about to occur, a sort of tree-traversing search is required to properly implement the second merge.[5] In contrast, when a simultaneous merge takes place as in (14), there is no burden of such a search-down procedure for an obvious reason. I assume that the third possibility

---

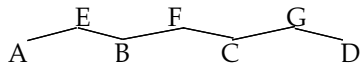
I do not assess the theoretical mileage of the midway conjunction analysis in this paper. (See Shin and Chung 2011 for some remarks.) I just would like to point out that Park's system is compatible with the process based explanation to be suggested in Section 4, as long as the remerge operations take place in a simultaneous fashion: b simultaneously merges with X in the second conjunct and &; a simultaneously merges with X in the first conjunct and &'.

[5] The later the remerge operation occurs, the severer the burden becomes. In (11), for example, the search down burden will become greater when the shared element remerges after &' is generated than when the VP in the second conjunct is generated.

is more economical than the other two due to the absence of the tree-traversing (or search-down) procedure.[6] Of course, the simultaneous merge should take into consideration two (or multiple) work spaces at the same time.[7] However, it is inevitable to do so as far as two complex nodes, for example, X′ and its specifier, merge at all (Uriagereka 1999).

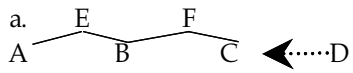
With this assumption in mind, let us now turn to the quirky remerge structures in (6) and (7). They both contain the following structure at a certain point of derivation.[8]

(15)

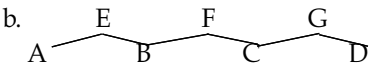


They crucially differ from the structure in (14) in that more than one element is shared at the point of simultaneous merge. One may try to first form E and F via simultaneous merge of B with A and C, and then merge of C with D, forming G at a later stage, as shown in (16) below:

(16) a.



b.



---

[6] In this sense, the so-called external remerge is a misnomer: terminology like parallel/simultaneous/multiple merge sounds more appropriate. In contrast, internal merge necessarily includes a step of 'remerge' since the node that the element remerges with is yet-to-be built.

[7] An anonymous reviewer questions why the derivation in (12) and (13) includes a counter-cyclic search-down procedure, which was claimed in the previous version of this paper. The reviewer is right in pointing out that the derivation in (12) and (13) may not include a counter-cyclic derivation unless every phrase is assumed to be a cycle. So I dropped the non-cyclicity part in this revised version. The reviewer further questions why a search down process is problematic. If a search down process is allowed in internal merge, why not in external merge? My answer to this question is "Avoid it unless it is necessary." In cases of external (re)merge, a search down process can be avoidable, while in cases of internal (re)merge, it is inevitable. Then the question lies whether a search down procedure is costly. It does seem so, as far as the merge-over-move principle is viable. The same reviewer suspects that, in the perspective of derivational steps, the derivation in (12) and (13), rather than the one in (14), appears to be more economical. I do not fully follow what the reviewer means by the perspective of derivational steps. If he or she intends to mean that merge prefers binarity, I would respond that the simultaneous merge does not discard binarity, either; rather it keeps it in a simultaneous fashion, when two or more work spaces share an element.

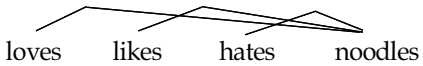
[8] Two of the anonymous reviewers worry that the proposed explanation may not apply to quirky internal cases of merge. I would like to stress, however, that the so-called quirky internal merge and external merge become indistinguishable when the order restriction disappears, as discussed in the previous section. They both include a sort of 'sideward movement' operation in the sense of Nunes (2001, 2004).

Then the last stage of derivation should include an undesirable search-down operation to properly merge D and C, the latter being already embedded.[9] Thus, such a derivation will lead to the aforementioned problem.

It cannot be simply said that (15) is excluded because three work spaces are involved at the same time. Notice that one element can be shared by three (or more) sisters. Consider, for example, a RNR construction in (17a) and the relevant VP structure in (17b):

(17) a. John loves, Mary likes, and Tom hates noodles.

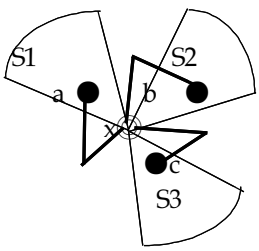
b.



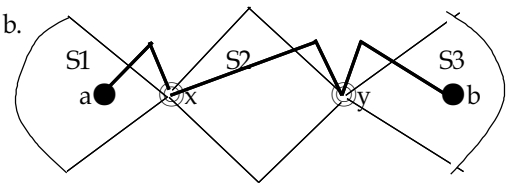
Though three work spaces are involved in (17a), no special problem arises as to the merge operation.

What then crucially distinguishes between the structure in (16) and (17b)? I claim the decisive difference lies in the arrangement of the work spaces at the relevant time of merge. In (17b), all the work spaces at a certain point of derivation share one and the same element, while in (16) there is no element shared by all the work spaces at the relevant point of derivation. Figuratively, (16) has the arrangement of work spaces as in (18b), while (17b) has the one in (18a):

(18) a.



b.



In (18a), all the work spaces (S1, S2, and S3) are adjacent to each other by sharing the unique remerged element x. In contrast, the adjacency relation cannot be obtained in (18b). S1 and S2 are adjacent to each other by sharing the remerged element x, and similarly, S2 and S3 are by sharing the remerged element y. S1 and S3, however, shares no element. Suppose only adjacent work spaces are tolerable at each point of merge, probably due to the heavy process

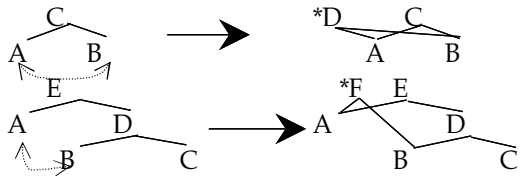
[9] The same search-down operation is required when A, instead of D, merges at the last moment.

load when mutliple spaces are processed at the same time. Then, the structure in (18b) conveys a heavier process load than the one in (18a).

The process load may have to do with the syntactic relation that the shared element and its sisters have. In (18a), the shared element and its sisters have a uniform relation: if the shared element functions as a head, then its sisters as non-heads and vice versa. In contrast, shared elements x and y in (18b) have non-uniform syntactic functions: if x is a head, so is b, while a and y are non-heads; if x is a non-head, so is b, while a and y are heads. In other words, all the work spaces in (18a) can be scanned in a breath, while those in (18b) cannot be.

Finally, let us consider the special cases of merge in (4), repeated below:

(4) a.
b.



In (4a) and (4b), two work spaces share two remerged elements and there will be no added process load, if only the work space adjacency matters.[10] (4b), however, seems to be ruled out due to the simultaneous merge requirements for shared elements. Since A and B are shared, their sisters have to be merged at the same time. But notice that D is produced only after B and C merge, which makes it impossible to merge A, B, C, and D simultaneously. Although (4a) does not suffer from such a problem, it seems to be independently ruled out due to some sort of uniformity violation: Merge produces a unique syntactic object. Thus, D cannot differ from C. If sharing is involved at all, the higher node C (= D), not its daughters A and B, is to be shared at the relevant point of derivation, due to the A-over-A principle that generally applies to syntax (Chomsky 1964, Ross 1967).

5. Conclusion

It seems to be intuitively correct that merge does not increase the number of roots, as de Vries' (2009) No Proliferation of Roots Condition (NPRC) tries to

[10] If the number of shared elements matters, the process load for (4a) and (4b) will be as heavy as the cases like (18b) since there are two elements shared.

capture. This paper has shown, however, that de Vries' (2009) account of restrictions on merge, expecially so-called quirky remerge cases, in terms of Root Condition (RC) or NPRC does not work due to the fact that these conditions may or may not be satisfied, depending on the variation on the order of remerge. It was also shown that the two conditions are not co-extensional, casting doubt on the validity of de Vries' (2009) formalization of RC as NPRC.

This work instead has attributed the restrictions on merge, especially the so-called quirky remerge, to the process load based on some plausible assumptions on the timing of merge and on process load variations depending on the arrangement of work spaces involved. It was first assumed that, as far as external remerge is concerned, syntactic terms merge as many times as possible, in order to avoid the less-economical search-down procedure that would otherwise occur. Thus, the external remerge structure is to be produced by an instantaneous multi-merge operation, but not by a 'remerge' or recycling operation. It was then assumed that the availability of remerge or more correctly multi-merge depends on the arrangement of the work spaces involved: Only the adjacent work spaces are permitted at each stage of merge operation. Given these two assumptions, the so-called quirky cases of remerge are ruled out since they violate the adjacency condition on work spaces at the relevant time of merge.

One important issue that is not dealt with in this paper is whether internal remerge (movement) receives the proposed analysis. The remerged/moved element cannot simultaneously merge with its sister in its original position and the sister after remerge for an obvious reason: The latter is unavailable up until all the nodes inbetween have been generated. Further research is definitely required to solve such a problem (and many others), but for the time being I follow Chomsky (2001) and Di Sciullo and Isac (2008) in assuming that internal merge differs from external merge.

References

Abels, Klaus. 2004. Right node raising: Ellipsis or ATB movement? In Kier Moulton and Matthew Wolf (eds.), *Proceedings of the 34th North East Linguistics Society,* 44-59. Amherst, MA: GLSA Publications, Umass.

Bachrach, Asaf & Roni Katzir. 2009. Right-node raising and delayed spellout. In Kleanthes K. Grohmann (ed.), *InterPhases: Phase-Theoretic Investigations of Linguistic Interfaces (Oxford Studies in Theoretical Linguistics* 21), 283-316. Oxford: Oxford University Press.

Di Sciullo, Anna Maria and Daniela Isac. 2008. The asymmetry of merge. *Biolinguistics* 2. 4: 260-290.

Blevins, James. 1990. Syntactic complexity: Evidence for discontinuity and multidomination. Amherst, MA: University of Massachusetts dissertation.

Bobaljik, Jonathan David. 1995. In terms of merge: Copy and head movement. *MIT Working Papers in Linguistics* 27: 41-64.

Bobaljik, Jonathan David & Samuel Brown. 1997. Interarboreal operations: Head movement and the extension requirement. *Linguistic Inquiry* 28: 345-356.

Carnie, Andrew. 2008. *Constituent Structure* (*Oxford Surveys in Syntax & Morphology*). Oxford: Oxford University Press.

Chen-Main, Joan. 2006. On the generation and linearization of multi-dominance structures. Baltimore, MD: John Hopkins University dissertation.

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

Chomsky, Noam. 1995. *The Minimalist Program* (*Current Studies in Linguistics* 28). Cambridge, MA: MIT Press.

Chomsky, Noam. 2001 Derivation by phase. In Michael Kenstowicz (ed.), *Ken Hale: A Life in Language* (*Current Studies in Linguistics* 36), 1-52. Cambridge, MA: MIT Press.

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

Chung, Daeho. 2004. A multiple dominance analysis of right node raising constructions. *Language Research* 40: 791-812.

Citko, Barbara. 2005. On the nature of merge: External merge, internal merge, and parallel merge. *Linguistic Inquiry* 36: 475-496.

Gracanin–-Yeksek, Martina. 2007. About sharing. Cambridge, MA: MIT dissertation.

Guimarães, Maximiliano. 2004. Derivation and representation of syntactic amalgams. College Park, MD: University of Maryland dissertation.

Johnson, Kyle. 2007. LCA + alignment = RNR. Paper presented at the workshop on Coordination, Subordination and Ellipsis, Tübingen. [Eberhard-Karls-Universität Tübingen, 7-8 June 2007.]

Kayne, Richard S. 1984. *Connectedness and binary branching*. Dordrecht: Foris.

McCawley, James. 1982. Parentheticals and discontinuous constituent structure. *Linguistic Inquiry* 13: 91-106.

Nunes, Jairo. 2001. Sideward movement. *Linguistic Inquiry* 32: 303-344.

Nunes, Jairo. 2004. *Linearization of Chains and Sideward Movement* (*Linguistic Inquiry Monograph* 43). Cambridge, MA: MIT Press.

Ojeda, Almerindo. 1987. Discontinuity, multidominance, and unbounded dependency in generalized phrase structure grammar: Some preliminaries. In Geoffrey J. Huck and Almerindo E. Ojeda (eds.), *Discontinuous Constituency* (*Syntax and Semantics* 20), 257-282. New York: Academic Press.

Park, Myung-Kwan. 2005. When things are cumulated or distributed across coordinate conjuncts. Paper presented at the SICOGG 7.

Park, Myung-Kwan. 2007. RNR in Korean as right-edge coordination. *Studies in Generative Grammar* 17.1: 85-97.

Park, Myung-Kwan. 2009. Right node raising as conjunction reduction fed by linearization. *Language Research* 45.2: 179-202.

Park, Myung-Kwan. 2010. RNR as midway conjunction = external remerge. *Studies in Modern Grammar* 61: 25-50.

Postal, Paul M. 1998. *Three investigations of extraction*. Cambridge: The MIT Press.

Shin, Keun Young and Daeho Chung. 2011. Remarks on the midway conjunction analysis of RNR constructions. *Studies on Generative Grammar* 21.1, 97-116.

van Riemsdijk, Henk. 1998. Trees and scions—science and trees. In *Chomsky 70th Birthday Celebration Fest-Web-Page*. Retrieved from http://cognet.mit.edu/Books/celebration/ essays/riemsdyk.html.

van Riemsdijk, Henk. 2006. Grafts follow from merge. In Mara Frascarelli (ed.), *Phases of Interpretation* (*Studies in Generative Grammar* 91), 17-44. Berlin: Mouton de Gruyter.

Ross, John R. 1967. Constraints on variables in syntax. Cambridge, MA: MIT dissertation.

Uriagereka, Juan. 1999. Multiple spell-out. In Samuel D. Epstein and Nobert Hornstein (eds.), *Working Minimalism,* 251-282. Cambridge, MA: MIT Press.

de Vries, Mark. 2005. Ellipsis in nevenschikking: Voorwaarts deleren, maar achterwaarts delen. *Tabu* 34: 13-46.

de Vries, Mark. 2009. On multidominance and linearization. *Biolinguistics* 3.4: 344-403.

Wilder, Chris. 1999. Right node raising and the LCA. In Sonja Bird, Andrew Carmie, Jason Haugen, and Peter Norquest (eds.), *WCCFL18 Proceedings*, 586-598. Somerville, MA: Cascadilla Press.

Wilder, Chris. 2008. Shared constituents and linearization. In Kyle Johnson (ed.), *Topics in Ellipsis*, 229-258. Cambridge University Press.

Daeho Chung
Department of English Language and Culture

Hanyang University
1271 Sa-3-dong, Ansan-si
Gyeonggi-do 426-791
Korea

cdaeho@hanyang.ac.kr

Received: 2011. 7. 13
Revised: 2011. 8. 15
Accepted: 2011. 8. 20