Is Movement a part of Narrow Syntax?

*Erich Groat, Goethe-University Frankfurt*

22▪V▪13

*The answer tentatively suggested here is* “No!” *Syntactic Objects that appear to move in syntactic structure are always and only remerged. The syntax excludes type-token distinctions, maintains no notion of chains, and does not distinguish between occurrences in any way; thus structure-building of NS has no notion of movement. How multiple occurrences of one Syntactic Object within another is interpreted at the interfaces (here, there, everywhere, some places, nowhere). Movement is thus an “effect of Transfer” – allowing NS to remain a context free system.*

# Basic idea of Movement and GB implementation

(1) The “displacement property” of natural language.

1. Structure S1 contains object X in some structural position P1
2. Structure S2 is transformationally related to (is derived from) S1
3. Structure S2 contains object X in some structural position P2 ≠ P1

The difference in structure may have phonological/interpretive consequences.

**GB tradition:**

(2)

PS Rules → D-Structure

S-Structure

PF LF

Move-

Move-

Phonology

Passives:

(3) a. D-Structure: [S [NP e] was [VP seen [NP John]]]

→ *move * →

b. S-structure: [S [NP John] was [VP seen t ]]]

X=[NP John]

S1=D-Structure P1=daughter of VP

S2=S-Structure P2=daughter of S

## Nodes: how is X identified

### Definition of Tree: relations (dominance, precedence) between independently distinguishable nodes

Phrase structure up to X-bar theoretic generally involved directed graphs, specifically *trees.* For linguistic purposes, the following sort of tree is used:

(4) **Definition:** A *Tree* is a n-tuple <N, D, P, L, B> where

 N is a (potentially infinite) set of distinct nodes;

D is a set of ordered pairs <ni, nj> where ni, nj ∈ N (Immediate Dominance Relations);

 P is a set of ordered pairs <ni, nj> where ni, nj ∈ N (Precedence Relations);

 L is a (finite) set of labels;

 B is a set of ordered pairs <ni, lj> where ni ∈ N and lj ∈ L (Label Relations).

Further constraints apply (such as “no crossing branches”), to be ignored here. PS rules and X-bar theory constrain the relations between nodes in terms of their labels; Kayne’s LCA (1994) removes P from the structure; Relational/Arc Pair Grammar adds labels for D (Edges) and further relations between elements of D (Sponsor/Delete).

### Movement: same node, different dominator

Movement minimally involves replacing one dominance relation (and one precedence relation, if specified) with another:

(5) Before movement: <nVP, nNP>∈D, <nS, nNP>∉D

 After movement: <nVP, nNP>∉D, <nS, nNP>∈D

 (where nVP = node labeled VP, i.e. <n, VP>∈B, etc.)

► **Crucial:** *the constant here is the* **node.**

Separate occurrences of NP John involve *different* nodes:

(6) [S John2 [VP loves John1 ]].

 <nVP, n1NP>∈D, <nS, n2NP>∈D

Both nodes share the same label but are distinct .

## The infinity of nodes

If linguistic expressions can involve infinite recursion, as in (7), then cardinality of the set N of nodes must be infinite:

(7) John1 said2 that3 Mary4 thinks5 that6 Frank7 believes8 that9. . .

This is classically not the case for labels, which are drawn from a fixed set, or derived from lexical features, i.e. projection

► The grammar thus have an infinite generative capacity for nodes, one that parallels, *but is not identical with,* the generative capacity of Phrase Structure Rules/X-bar Theory etc.

This is a curious redundancy!

## Target of movement

Move is a relation between trees in which a node receives a new dominating node: the target of movement.

### Subsitution

Nodes are put into new dominance relations (see (5) above)

S

NPn1

|

e

VP

NPn2

|

John

V

|

seen

Aux

|

was

S

NPn2

|

John

VP

NPn3

|

t

V

|

seen

Aux

|

was

### Adjunction

New nodes called segments are created that are related to a preexisting node.

(8) Topicalization:

 [S John, [S I like t]]

Two “segments” of S, yet one “category” S: requires enrichment of the definition of Tree in (4) above.

## Traces of movement

Movement does not simply remove the old dominance relation, it replaces it with a new one:

(9) a. D-Structure: [S [NP e] was [VP seen [NP John]]]

→ *move * →

b. S-structure: [S [NP John] was [VP seen tNP ]]]

 Before movement: **<nVP, nNP>∈D,** <nS, nNP>∉D

 After movement: <nVP, nNP>∉D, <nS, nNP>∈D, **<nVP, ntnp>∈D**

. . . where **ntnp** is “NP trace,” a distinct node dominating no phonological content, again (presumably) sharing the label NP.

Structure Preservation (Emonds (1970); see discussion in Koster 2007) is about relations between labels, not individual nodes.

### Coindexation and Chains

Binding Theory:

(10) **Binding:**

X *binds* Y iff X c-commands Y and X is coindexed with Y

(11) **Indexation:**

Add I, J to Set n-tuple, where

I is a set of indices

J is a set of pairs <n, i> where n∈N and i∈I

(12) **Coindexation:**

X is *coindexed* with Y iff

∃i∈I [ <X, i>∈J ˄ <Y, i>∈J ]

If coindexation exists as a coreference mechanism, chains can be defined using them (further enriching syntactic structure).

 (13) **Chain** (loose definition for our purposes)**:**

 A *chain* ch is an ordered set C of nodes <n1, n2, …, nn> s.t.

nj c-commands nj+1, and

 ∃i ∀n∈C [ <n,i>∈J ]

► Chains therewith entail coreference of distinct nodes, to the extent that the nodes are of a type implicated in reference.

**My first suspicion:** This connection of chain formation with binding theory and coindexation has too strongly shaped our idea of movement.

Note that, in principle, chains could be defined without coindexation, and coindexation could obtain without chain formation.

# Bare Phrase Structure: Sets

## Chomsky 1995: No labeled nodes, only sets

We begin with “terminal nodes” as LIs and recursively Merge them. Sets in a sense replace the nodes of directed graphs/trees, and set-containment replaces (immediate) dominance.

(14)

Merge (seen, John) → {seen, John}

{seen, John}

 seen John

contains

contains

## Syntactic Objects

Syntactic Objects (SOs) are either LIs, or sets built from them via Merge.

### Complex sets containing labels

Pick the label of one of the parts and merge it directly (LIs are their own labels):

(15) {seen, {seen, John}}

The Syntactic Object is thus this whole set, and it contains two other SOs: *seen* and *John.* The inner set {seen, John} is not itself an SO; its sister *seen* is the label. (Equivalent to ordered pair <seen, John>.)

### Simple sets without labels (labeling “algorithm;” Collins 1998, Chomsky 2008, 2012)

Do not build the label into the structure; just stick with the basic merged set:

(16) {seen, John}

The Syntactic Object is again thus this whole set, and it contains two other SOs: *seen* and *John.*What category is this? Use a labeling algorithm (Collins 2002, Chomsky 2008, 2012, Blümel 2011, Rizzi 2012) when you need to know.

We will use simple sets and put aside the labeling question for now.

## Move as Remerge: Internal Merge

Groat 1995, 1997; Bobaljik 1995; Epstein et al. 1998; Epstein & Seeley 1999, Chomsky 2000: Move doesn’t “copy” anything; a “copy” is really an occurrence of a single syntactic object.

(17) {was {seen John} → {John {was {seen John}}

This point of view *begins* to answer the following questions:

1. Case is checked on higher occurrence of *John;* what about the lower?

*― It’s only one SO; there is only one feature to check!*

1. *John* gets a theta role in lower position; what about the higher?

*― It’s only one SO;it only needs one theta role!*

1. Higher occurrence of John is pronounced, what about the lower one?

*― It’s only one SO; there is only one set of features to pronounce!*

1. These two positions form an “A-chain;” how are they related without extra machinery like coindexation, chains, etc.?

 *― It’s only one SO; there are no multiple “things” to be related!*

## Multidominance

Multidominance falls out naturally under Move-as-Remerge, since dominance is conceived as containment:

(18)

{seen, John}

 seen John

contains

contains

{seen, John}

 seen John

contains

contains

{was, {seen, John}}

contains

contains

was

{seen, John}

 seen John

contains

contains

{was, {seen, John}}

contains

contains

was

{John, {was, {seen, John}}}

contains

contains

## Non-multidominance

If you want non-multidominance, the two SOs in question must differ. However, introducing some minimal featural difference, such as an arbitrary index, will not do. An categorical ontological distinction needs to be made.

### Indices and copying

Chomsky 1995, Collins & Stabler 2011: introduce indices upon selection of LIs from lexicon into the derivation (Select):

(19)

{saw, Johni}

 saw Johni

contains

contains

{saw, Johni}

 saw Johni

contains

contains

{v\*, {saw, Johni}}

contains

contains

v\*

{saw, Johni}

 saw Johni

contains

contains

{v\*, {saw, Johni}}

contains

contains

v\*

{Johnj, {v\*, {saw, Johni}}}

contains

contains

Johnj

Note that even indices involve a kind of “multidominance:”

(20) Johni <John, i> = {John, {John, i}}; <John, j> = {John, {John, j}}

Both pairs contain (i.e. dominate) the LI *John.*

### Introduction of Indices at Select: A suspicious violation of Inclusiveness

Clearly, these indices have no particular interpretation (as noted my Chomsky); they serve only to “differentiate” different occurrences of an LI. A violation of Inclusiveness, as Chomsky admits, but a necessary one (Chomsky 1995). This necessity – even its plausibility under minimalist assumptions – is what I contend against; see Rogers & Uriagereka 2011, cf. Shiroki 2011).

(21) **Inclusiveness**

“… outputs consist of nothing beyond properties of items of the lexicon (lexical features) – in other words, that the interface levels consist of nothing more than arrangements of lexical features” (Chomsky 1995, 225)

Motivation for violation of Inclusiveness:

(22) I told himi that everyonej’s mother loves himi/j.

Different interpretations of *him* possible, thus they must be distinguishable. But does this entail indices?

(23) Thei lion is one of the most remarkable animals on thej planet.

### Definition of “Set Equality”

(24) **Equality:**

Twos sets S1 and S2 are equal iff x∈S1 ↔ x∈S2.

## Chains and SOs

### Occurrences

Occurrences of a single SO can be thought of as

1. The set of the SO’s sisters
2. The set of the SO’s mothers
3. The set (possibly ordered) of paths from a dominating node to the SO
4. The set of the SO’s sharing a single index, i.e. of a single Token.

(25) {John {was {v {seen John}}}}

 Possible sets of occurrences:

 a. {{seen1, {was {v {seen John}2}}

 b. {{seen John}1, {John {was {v {seen John}}}2}

 c. {

<{John {was {v {seen John}}}},

{was {v {seen John}}}, {v {seen John}}, {seen John}>,

<{John {was {v {seen John}}}},

}

d. {Johni, Johnj}

Note that none of these objects are syntactic objects created by Merge (though they contain such objects). Cf. Epstein & Seeley 1999, 2006. Such sets could be used in the definition of chains, but again, new objects not crated by Merge are being defined.

Note incidentally that *any* notion of a “discontinuous syntactic object” is parasitic on the notion of syntactic object, as long as SOs are defined by Merge (perhaps they are not; see Blümel in progress). The original SO, no matter how many times it remerges, remains exactly the same object.

### A further note on indices, complex SOs, and Tokens

Even more of a mess! And it doesn’t straightforwardly get us out of multidominance:

(26) [ [the boy]i saw [the boy]j ]

{the boy}

i

{the boy}

j

{the boy}

i

j

Adding separate indices to the lexical items simply pushes the problem down another level. Indices do not “merely create” tokens: they encode their existence. They must be *interpretable* as such.

Furthermore, the cardinality of indices is infinite (since the grammar can create SOs with an unbounded number of “occurrences” of an SO). Thus we require a generative system to create indices (as Trees/directed graphs do for nodes).

Summary:

► True Tokenization requires the establishment of a *new ontological class* of objects distinct from Types. Following the Strong Minimalist Thesis: this is to be avoided!

► Tokens surely emerge in perception and production.

 But in the Narrow Syntax, all we should refer to are occurrences of single objects.

► Tokens involve indices, infinite in cardinality, bringing us back to a generative system needed in the days of Trees to distinguish nodes, a generative system *independent* of Merge (as generator of structure).

**My second suspicion:** taking the “generational” metaphor too seriously has made us think of the grammar as generating actual sentences whose tokens must be accounted for, when in fact the grammar generates the forms actual real-time sentences may have, delimiting *possible* sound-meaning pairings. Minimally, this would involve no type-token distinction at all.

# Can we do without all indices and objects not created by Merge?

If the above argument is correct, then indices will only work if we introduce a type/token distinction into the grammar. If we reject this, we are left with only occurrences to work with. We can build notions of Chain around them, through mechanisms other than Merge.

## What sets really give us

 (27) a. John’s **mother** told *Frank’s* **mother** about *Frank’s* father.

 b. I told *himi* that **everyonej**’s mother loves ***himi/j/k.*** (Indices for interpretation only!)

 c. *Gargoyles* love *gargoyles*.

In (x.a), there’s only one LI *Frank,* only one LI *mother.* Both contribute exactly the same (possibly ambiguous) semantics to the DPs that contain them.

In (x.b), coindexation does not imply “coreference” in the bound variable reading, but a much more complex indexical relation between quantifier and variable. Coindexation in the non-bound variable reading is incidental, and lack of coindexation falls into the same as a case of “accidental coindexation.”

In (x.c) there is no hint of a Condition C violation.

We should *expect* that identical SOs always contribute the same *potential* information to semantic interpretation. The *context* determines their interpretation.

## Can/need NS see whether something is “*moved?*”

Assume a phase-based approach.

(28) a. [T was seen John]

 b. C [**John** T was seen ~~John~~]

 c. (cf. **John** saw **John**)

If there is only one *John,* does NS need to know whether it is a “new” *John* in Spec TP (External Merge), externally merged “from the lexicon,” or the old *John* being remerged (Internal Merge)?

► Possibility: Internal Merge takes place as part of the operation Transfer (Chomksy 2008, 2012, M. Richards 2012, cf. Hornstein 1999: relation is established as result of “copying”)

Assuming remerge of *John* in (x.b) above is somehow connected to probing by uF, valuation of phi features on T and Case features on *John,* the Remerge is perhaps a part of Transfer. It certainly is for Unvalued Feature-driven Movement (UFM):

(29) [ John vdef [VP seen John ]]

PF “knows” somehow not to linearize *John* in VP. But does the narrow syntax “encode” the fact that this Merge was at Transfer?

i.e.: does the NS *need* to see whether something is moved derivationally? No, not here:

Cf. (29) with (30) below, with accusative-valuing v\*:

(30) [ John v\*[VP saw John ]]

If the higher *John* was merged outside of Transfer, then the interfaces need to “know” that it’s not the “same” *John;* the lower *John* should thus be spelled-out separately.

► But the information is already there: v\* differs from vdef in having ability to value Case, and having ability to assign a subject theta-role.

Spell-Out of VP can thus include *John* in (30) but not in (29). Semantic composition of vP can include John as external argument in (30) but not in (29).

NS doesn’t seem to need to “know” anything about remerger of *John.*

**→ Continuing the derivation of the passive:**

(31) a. [ C [TP was [ John vdef [VP seen John ]]] (Spelled out so far: “seen *t*”) →

 b. [ C [TP John [ was [ John vdef [VP seen John ]]]]

PHON: There are two candidates for Spell-out of *John.* Assumption: if there is more than one occurrence of an SO in a phase, the highest one wins. (Natural if linearization is “Top Down”.)

SEM: *John* does not compose with vP; wrong semantic type. (Or in Groat’s Total Transfer framework: *John* never moved to this position in the first place!). *John* in Spec TP may be associated with “Topic-Comment” relations, specificity, etc., and contributes there.

Note that the C phase does not care what happened in the v phase. It doesn’t know how *John* was merged.

PROBLEM: What about the theta criterion?

(32) \*[ C [TP John was [ John vdef [VP raining]]]]

What’s wrong with this? Is it a syntactic problem, or a semantic one?

POSSIBILE SOLUTION: It’s a semantic problem – a CP has been completed – potentially a full utterance – yet there is a DP in the derivation which has not contributed all of its semantics.

(33) **Semantic Theta-Criterion:**

DP must receive a theta role within a finite clause.

► Discussion point: How can this be understood?

Leah’s suggestion: DP first-merged into a non-theta position immediately crashes (or must be able to receive some other, non-argument interpretation).

**→ Continuing the derivation of the active:**

(34) a. [ C [TP T [ John v\*[VP saw John ]]] (Spelled out so far: “seen John”) →

 b. [ C [TP John [ T [ John v\*[VP saw John ]]]]

PHON: There are two candidates for TP Spell-out of *John.* Assumption, as before: if there is more than one occurrence of an SO in a phase, the highest one wins.

SEM: *John* composes with vP as External Argument. *John* in Spec TP may be associated with “Topic-Comment” relations, specificity, etc., and contributes there.

PROBLEM: If the Semantic Theta-Criterion is correct, it is satisfied here; *John* has gotten two theta roles in CP. Yet these *Johns* are not necessarily coreferent (in fact, if Condition C works here, they must not be).

POSSIBILE SOLUTION: Note that the two occurrences of *John* are associated with two different structural Cases. (cf. Kitahara 1998, cf. Sabel 2012 on Binding: an element can be bound only once its Case is valued; an element can bind only when its Case is unvalued.)

(35) **Case interpretation:**

 DPs with distinct Case values within a finite clause are interpreted independently.

Thus Structural Case is not “uninterpretable:” it is a means by which otherwise featurally identical SOs may receive different interpretations:

(36) Gargoyles love gargoyles.

Note that so far we have needed to say nothing about discontinuous objects, chains, sets of occurrences, etc. in the syntax.

### Obligatory Control as “Movement”

Well, we have no “movement” per se, only remerger of identical SOs! Consider

(37) [ C [ John T [VP ~~John~~ wants [ to [VP ~~John~~ explode ]]]]]

There is only one Case relation here between T and all other occurrences of *John* in the CP, thus all occurrences are interpreted as one. Similarly:

(38) [ C [ John T [ [VP ~~John~~ exploded ] [ to [VP ~~John~~ surprise the guests ]]]]

### Quantifiers and the Equi-NP problem

Quanitifiers: destroyed “Equi-NP analyses” of null elements. A problem from “Mere Multiple Merge?”

These do not mean the same thing:

(39) a. Everyone wants to explode.

 b. Everyone wants everyone to explode.

Interpretation of “trace” of everyone *not* the same as interpretation of head.

(40) a. Scope position: *everyone* = everyone<e, <e,t>>

 b. Base position: *everyone* = x<e>

It seems to be the overt Case position of everyone that binds the variable. Thus (x.b):

(41) a. everyone<e, <e,t>> [x<e> wants to [x<e> explode]]

 b. everyone<e, <e,t>> [x<e> wants [everyone<e, <e,t>> to [y<e> explode]]]

### Parasitic Gaps

Something similar going on with wh-features:

(42) [ Which books did [ you [[VP file ~~which books~~ ] [PP without reading ~~which books~~]]]]

vP

which books

VP

file

PP

XP

without

which books

v

which books

which books

reading

vP

Cwh

which books

CP

Remember: all occurrences of *which books* are **one** syntactic object!

All requirements are met for “linked” interpretation:

* All occurrences within CP
* Case is the same/close enough within CP ( = Case-matching requirement on P-gaps)
* One position available for linearization (Spec CP, highest position wins)

► Single licensing feature on two elements means two elements interpreted together.

Again: no reference to chains, sets of occurrences, Internal vs. External Merge

PROBLEM: what prevents “movement” out of adjuncts?

(43) [ Which books did [ you [[VP file the papers] [PP without reading ~~which books~~]]]]

(Assume as in (42) that *which books* is merged to edge of PP.)

Some property of the matrix C can be satisfied only by the presence of the object whP corresponding to the “true” gap.

► Discussion points:

* What sort of property is this?
* What does this tell us about “Probe Goal” relations?

### Free Relatives

Another case of “shared” elements?

(44) The teacher gave a better grade to [**whoever**’s parents] donated money to his club.

: goal : POSS

**Questions:**

Case mismatches in German etc.?

(45) [WessenGEN Vater] reich ist hat keine Sorgen.

NOM

Structures are also unclear here.

## Final Thoughts

### NS is “ontologically” bare-boned

If this approach is on the right track, we have a theory of syntax which provides an infinite class of arrangements of a finite set of formal features, without requiring an infinity of tokens: there are no tokens. Compared to a computational system involving tokens, this is “ontologically” simpler. It is up to PHON and SEM to decide how to “divide up” occurrences into tokens.

### Is NS context free?

Merge is completely free in such a system; no structural transformations are triggered by local context. Post-syntactic operations provide the illusion that NS is context-sensitive.

### Other things to think about:

Other ontological “problem children:” Numerations, Workspaces

Other “connectedness” situations: Binding, Copy Raising