Balázs Surányi

Towards a purely derivational approach to syntax[*]

0 Introduction

In this working paper I outline a particular implementation of the 'derivational' approach to narrow syntax (see Epstein et al. 1998, Epstein and Seely 2002, 2006), which is purely derivational. The inspiration for the endeavor to devise such an approach comes primarily from Brody's (1995, 1997, 1998, 2000, 2002, 2006) work. Brody is well-known for relentlessly criticizing mixed derivational–representational models of syntax within the minimalist paradigm (see especially Brody 1995, 2002), and for good reasons: there is apparently a lack of solid empirical evidence to back up the view that the degree of power associable with having both representations and (derivational) operations is necessary in syntax. Note that even so-called 'radically derivational' accounts, including Epstein et al. 1998, Uriagereka 1999, Epstein and Seely 2002, employ both (derivational) syntactic operations and (small but genuine) syntactic representations (see Brody 2002 for this point). Given that the enhanced power of mixed models is apparently not necessary, either we should have only representations or only derivations. Brody (1995 et seq.) has been advocating the former view. If taken seriously, Brody's criticism of 'mixed theories' should lead one to a purely derivational approach to phrase structure, if the derivational approach is to meet the challenge.

The paper starts out by highlighting some undesirable complications caused by 'first Merge.' I show that a derivational account that has 'first Merge' and 'second Merge' is problematic on several counts, and 'first Merge' had better not exist, if these complications are to be avoided. What I then propose is a model that dispenses with 'first Merge.' A highly welcome consequence of this proposal is that labeling will be rendered unnecessary. In fact, dispensing with 'first Merge' allows us to develop a stronger proposal, from which the inexistence of labeling actually follows: namely, that in a derivational approach, the postulation of projection can also be eliminated. Within the

* I am grateful to Misi Brody for reading and providing valuable comments on a pre-final version of this paper, and for many discussions, long and brief, over the years. I thank the organizers of the Workshop on Morphology, Syntax and Information Structure held at Lund University in 2006 for inviting me, and the audience there for questions and remarks. Needless to say, all errors are my own. I gratefully acknowledge the support of grant no. D-048454 of the Hungarian Scientific Research Fund.

The Even Yearbook 7 (2006), Department of English Linguistics, Eötvös Loránd University, Budapest
ISSN 1218–8808, http://seas3.elte.hu/delg/publications/even,

derivational model I adopt, this directly entails that no constituents exist in syntax: syntax relates Lexical Items only.

Note that the model developed here does not introduce anything into the (family of) 'radically derivational' approach(es) it attempts to supersede; instead, it takes away what appear to be unnecessary ingredients. Very simply, I propose a model that keeps to the null hypothesis: *only* derivational constructs (operations) exist in a derivational approach.

1 Labelling / Locus and 'First Merge'

My starting point is Chomsky's (2000 et seq.) labeling generalization. Chomsky (2000) suggests that it is always the 'selector' that provides the label, i.e., the element that has some uninterpretable feature to saturate, i.e. check. When a head H is Merged with a phrase K and attracts XP to [Spec,H], there are actually two Merge operations taking place: (i) that of H and K, and (ii) that of [H+K] and XP. Only if it is ensured that at step (i) it is H that projects the label will it be possible at step (ii) for H to act as the labeler of the final phrasal projection. Let us zoom in now on what determines labeling at stage (i). It turns out that there are two factors that may determine H to be the label. If we assume H to c-select K, then H will project the label *qua* selector, as it is H that bears an uninterpretable (c-selectional) feature. But in fact H bears an uninterpretable feature (that K does not bear) anyway, namely the [uF] that 'attracts' the goal XP inside K. Then, the c-selectional feature on H is redundant as far as determining H to project at step (i) is concerned. In fact, it is argued at length in Surányi (2006) that c-selection is not narrow syntactic, and there are no c-selectional features in narrow syntax (see also Nilsen 2003, Chomsky 2004, and references in Surányi 2006). If true, this is a welcome conjecture: we have just seen that c-selection is redundant anyway in narrow syntax, as far as the mechanism of labeling is concerned. Note, nevertheless, that c-selection is redundant only if Agree exists.

Thus the operation of Agree turns out to be essential in Chomsky's system in order to be able to derive labels provided that no checking of c-selectional features exists in narrow syntax. However, as it is argued in Surányi (2006), given that the syntactic relation licensing checking (i.e., the checking configuration) should be a natural relation in the sense of Epstein (1999), Chomsky's (2000 et seq.), Agree should not be the checking relation; in fact, it should not exist to begin with. Checking should also be licensed under some natural relation (or as part of a natural operation, given a derivational approach), a local relation supplied by Merge (either 'sisterhood', or

'immediate containment', depending on how Merge is defined).[1] Assume then that Agree indeed does not exist, instead, XP checks H only when XP is Merged in [Spec,HP] (i.e., under a local relation supplied by Merge). Merger of XP can be performed either overtly or covertly (a view embraced in Chomsky 2004), which implements overt vs. covert movement (see also Pesetsky 2000, Nissenbaum 2000) (compare Bošković 2005, who puts forward a suggestion along these lines, but he limits it to overt movement).

Can we still continue to maintain, nevertheless, that it is the checked element that projects, following Chomsky's 'selector/probe projects' idea? As it turns out, this question depends crucially on what actually determines that the H should project the label in step (i) (step (ii) being straightforward, as there it is apparently the 'selector/probe' that projects the label)?[2] As H does not get checked in step (i), checking will not determine H to be the label in step (i). In fact, it is not even clear what motivates the Merge operation itself in step (i) within narrow syntax to take place, other than lookahead: step (i) makes step (ii) possible, and step (ii) results in the checking of H by XP.

Thus, we have the following problem for labeling: the need of labeling results in lookahead once it is recognized that (a) c-selectional phenomena are not narrow syntactic, and (b) Agree should not exist if syntax conforms to minimalist expectations (the checking function of Agree is to be reduced to Merge). The underlying problem, of course, is: Merger of (functional) head and its complement is not locally triggered. The output of this Merger serves as input to a Merge operation that will ultimately license the checking of some feature of the head, but that can be no trigger for Merger of head and complement (assuming lookahead to be inexistent).

Collins (2002) proposes to do away with labels, which he considers to be a representational residue.[3] Collins' proposal effectively introduces a trade-off that does not obviously result in a genuine simplification of syntax: he replaces labels with a memory buffer storing what he refers to as the 'Locus'

---

[1] For Epstein et al. 1998, checking takes place under derivational sisterhood, i.e., mutual c-command: A c-commands B prior to movement, and B c-commands A, after movement. However, c-command is not a natural relation, see Brody 2000; and also, derivational c-command is a (multi-)representational notion.

[2] It cannot be the checking of H by XP contained in K that determines that H should project at step (i), because then we would not understand why H projects when XP is Merged to [H+K] (because checking of H by XP, by assumption, already took place in the previous step, step (i)). If Agree does not exist, and both overt and covert movements are cases of category movement, then this second dilemma cannot be resolved by reference to an 'EPP' feature on H (rendering H a 'selector/probe' in step (ii) as well): this would not take care of projection (at step (i)) in covert XP-movement.

[3] This view is well-grounded, I believe, only if labels created by the Merge operation are re-used later in the derivation.

of the derivation, i.e., the head that is being checked. One form of operational 'memory' (labels) is traded in for another form of operational memory (memory buffer storing the Locus); both can be taken to be devices that *represent* information about some earlier derivational operation in order to make that information available to some later operation.[4] Ideally, then, the Locus should ultimately go too, if narrow syntax is to be made purely derivational. In the model I propose, as will be shown below, the notion of Locus is dispensable. Collins exploits the notion of Locus through his Locus Principle. A central function of this principle is to warrant the contiguity of Merging a complement to a head H, and Merging a specifier attracted by H to head H. Another function is to guarantee that if a head H is such that it attracts several specifiers, then no new head should be introduced as long as H can still attract a(nother) specifier element. If the (quasi-representational) notion of Locus is disposed of, these descriptive generalizations need to be taken care of in some other way. This issue will be taken up in section 3.

It is easy to see that, besides not obviously improving on the model with labels, Collins' label-free model is also not immune to the problem identified above: although the lookahead for labeling is gone (there being no labels), the deeper problem of the lack of a local trigger for Merge of (functional) head and complement arises in the same manner.

Let us return now to steps (i) and (ii) above. Chomsky terms these 'first Merge' and 'second Merge', both 'triggered' by the 'selector/probe' in H. As we saw above, assuming a more elegant model than the standard one (without Agree, and without narrow syntactic c-selection), makes a problem that is covertly present in the standard model stick out: we do not have a genuine understanding of what locally triggers 'first Merge' in syntax (i.e., without lookahead). The same question can be put differently: why should movement of some element be implemented by a *composite* operation subsuming Agree (licensed by 'first Merge') and displacement/re-Merge/internal Merge (corresponding to 'second Merge'), instead of just the second component of this composite (making it also possible to define checking as a local and

[4] The difference between the notion of label and the notion of Locus is that, even though both ensure that some property of a complex syntactic object (a phrase) is accessible for later operations, while the Locus-based theory makes this information accessible only for a limited number of subsequent operations, labels in principle remain accessible at any later point of the derivation. This consideration, however, does not grant us a simple a priori choice between Label and Locus, as only whole theories can be attempted to be compared, when Occam's Razor serves as the measure. It turns out that in a derivational model with sufficiently small Spell Out domains, i.e., in models where a Spell Out domain (phase) is no greater than a phrase (see Bošković 2002a, 2005, Müller 2004, Surányi 2002, 2006; Epstein and Seely 2002), labels persist no longer than the Locus does, due to the radical cyclicity of Spell Out.

natural operation, as part of (re-/internal) Merge)?

Thus, one question we have is: why do both 'first Merge' and 'second Merge' exist, when on the one hand, 'first Merge' has no clear narrow syntactic trigger in itself, and on the other hand, implementing movement requires only the 'second Merge' operation, the 'first Merge' operation being independent of it? We can now add two further dilemmas related to the same 'first Merge' / 'second Merge' dichotomy.

If indeed step (ii) is triggered to eliminate/value some [uF] *of H*, then ideally, not only 'first Merge' but also 'second Merge' should happen *to H itself*. On the standard view, this is not the case for 'second Merge': 'second Merge' applies to the *constituent* created by 'first Merge'. This is a prima facie imperfection (in Chomsky's 1995, 2000, 2001 sense).

Another question we have not so far addressed stands out: what determines the *order* of first and second Merge, e.g., why should T first Merge with *v*P and only after that Merge with DP? Note that the answer cannot simply be that Merge of DP to T is 'internal Merge', i.e., it requires DP to be present internally to the constituent that T has already been Merged with: [SpecTP] can be filled by an element Merged from outside (e.g., an expletive; or in case of lexical heads, an argument).[5] The issue of why the element that will become the 'complement' is Merged first, and why the element that will become the 'specifier' is Merged second has not yet been resolved in Chomsky's (2000 et seq.) model.[6]

---

[5] Also, this would make no sense in the model that is being developed here: as we will see, T and *v*P do not form a constituent, at least not one that Merge applies to in the second step.

[6] Surányi's (2006) approach to head movement within the domain of extended projections provides an answer, but this answer is only partial. I propose there that (1) complex syntactic heads, which are word-level elements containing all inflectional elements as syntactic components, are moved out from each phrasal projection in order to save the derivation from crashing at the next step, on the assumption that every phrase is a domain for Spell Out (i.e., a phase) (see also Bošković 2002a, 2005, Müller 2004, *inter alia*), and (2) when moved and re-Merged to the root node in this manner, a syntactic head reprojects: it projects another one of its inflectional components that bears some [uF] to check. This projected component then enters checking with a matching XP, triggering movement of XP. It follows that Merger of head and its complement precedes Merger of [head+complement] and specifier, in configurations where the head is actually head-moved out of the complement. This model then does not answer the 'order' question for cases when the head is an element that is independent of the complement it takes. One such scenario is when a predicate takes an argument. For this case, it may be safely claimed that heads taking an argument do not take a specifier, because arguments themselves are in the specifier position. This follows the line of Sportiche's (1999) version of an articulated VP-shells (or predicate-shells) structure, where there is a one-to-one relation between verbal predicative heads and arguments (see also Kratzer 1996, and Basilico 1998, who take direct objects to invariably be generated in a Spec,VP position).

2 Complementation as an extra-syntactic phenomenon

Before I proceed to address these problems, let us step back for a moment to consider an issue that I have not addressed yet, but which is highly relevant to the mechanism of 'first Merge': namely, functional complementation.[7] I pointed out above that c-selection should not form part of narrow syntax. On such a view, the functional hierarchy (which is also standard to view as determined via syntax-internal c-selection, see Cinque 1999) can only be determined in some interface component. There are basically two options: either the relevant component of grammar interfacing with narrow syntax is the semantic interface (a common conception, and one that is more than likely to be correct for at least a number of cases; see e.g., Ernst 2002, Nilsen 2003), or it is the Lexicon, should there exist aspects of the functional hierarchy that are not reducible to semantic requirements (see Surányi 2006 for a possible Distributed Morphology implementation of this approach, where Merge operations are 'checked against' the (contextual) definitions of Vocabulary Item entries). Evidently, we would like to see as much of the functional hierarchy (FSeq, for short, see Starke 2001 for this term) to belong to the domain of the first of these two components. Work on shifting as much as possible from the second category to the first has been started only recently, and with considerable success (see, e.g. Koeneman 2000, Ernst 2002, Nilsen 2003). The general layout has long been clear, which in itself makes the success of this endeavor highly plausible: the hierarchy proceeds from internal argument licensing to external argument licensing, event structure and aspect, through to modality, mood and tense, and then finally to discourse-related

---

The other scenario is when a functional (inflectional) head is realized as a free morpheme, taking a phrase, typically another functional phrase, as its complement (e.g., C takes TP, and C can also attract a *wh*-specifier). So, the ordering problem does not go away in Surányi's (2006) model either.

[7] I ignore lexical complementation here (under sisterhood with the head), as non-existent in the sense of the previous footnote: all arguments occupy a specifier position. In a sufficiently articulated shell-structure for 'lexical' projections, no lexical argument needs to be treated as a complement: even the lowest argument (treated as a complement on a Larsonian approach) can be analyzed as a specifier, similarly to all higher arguments.

I follow Chomsky (2004) in taking a lexical predicate (like a verb) *not* to bear an uninterpretable c-selectional or theta-feature. A lexical predicate *itself* is nevertheless uninterpretable without its (internal) argument, and the same is true for 'argument-taking' 'verbal' heads like *v*, Appl(icative), etc. Hence, Merging an argument expression to them turns these 'argument-taking' verbal heads into interpretable structures. Merger of arguments satisfies Last Resort: Last Resort requires that a syntactic operation take place only if it eliminates some uninterpretable property from the domain that is to undergo Spell Out (to be interpreted).

properties of focus and topic. I will assume here that FSeq is regulated by requirements on semantic interpretation, in particular, in the form of the lexical semantic requirements of the elements entering the derivation. Given that lexical semantics of elements are subject to a certain degree of variation, this view allows for what surfaces as slight variations in FSeq across languages. Note, finally, that I adopt the view that semantic interpretation is built on semantic representations: interpretation involves a number of truly representational concepts, like scope and variable binding, and even the most elementary of operations (like conjunction) can apply to complex elements. In fact, semantic requirements of elements that produce an FSeq on the surface are often *relative* to some other, non-local element, e.g. A must/cannot be in the scope of B (see Nilsen 2003).

In a cyclic Spell Out approach, checking whether the FSeq segment that has been built is semantically licensed takes place cyclically in small chunks, which cuts down potential overgeneration (of ill-formed FSeq segments). The smaller the Spell Out domain is, the less overgeneration there can be. If every phrase is a Spell Out domain, as suggested in Frampton and Guttmann (1999), Müller (2004), Bošković (2002b, 2005) and Surányi (2002, 2006), overgeneration is filtered out (almost) *immediately***.**[8] As soon as a (functional) phrase is built, and is subjected to Spell Out, it is verified (by the semantic component) that the phrase that is built is in conformity with conditions regulating FSeq. I adopt this view of the size of 'phases'/Spell Out domains here.

3 Towards a purely derivational approach

3.1 Eliminating 'First Merge'

We are now in the position to develop an answer to the three problems identified at the end of section 1. Let us begin by addressing the second one of these issues, the problem of why it is not H itself that is involved in both 'first Merge' and 'second Merge'. One potential resolution of this second complication is to adopt the view that both 'first Merge' and 'second Merge' apply to H itself, i.e., to the element to be checked. To illustrate, this means that when *v*P is built, then TP is constructed by applying the following Merge operations (here, and below, A + B stands for Merge(A,B)):

[8] Only *almost* immediately, because projection after 'first Merge' is checked only after the whole phrase has been built, and sent to undergo Spell Out, i.e. after 'second Merge' has taken place.

(1) a. T + *v*P
    b. T + DP

Evidently, on such an account we have no TP constituent in the sense of standard phrase structure. If Merge is translated into immediate dominance, derivations like (1) create representations with multiple dominance of head elements. This is nothing new: Chomsky's (2004) approach to movement as 'internal Merge' involves multiple dominance of the 'moved' element. Note that a repercussion of having a derivation like (1) is that the result of T + *v*P does not need to be labeled (by T), as T is Merged directly with the specifier element.

Having isolated 'first' and 'second' Merge in this manner, I can now attend to the other two problems. The (admittedly bold sounding) proposal I would like to make is the following. Given that we genuinely understand neither why 'first Merge' takes place (what it is triggered by), nor why it takes place *first*, it would be better if it did not take place in narrow syntax at all. I propose that it doesn't.

Consider what this amounts to. If there are no Merge operations in the derivation that correspond to earlier 'first Merge' operations, then (1a) above is eliminated. Checking of a head by a 'specifier' element continues to involve the direct Merger of the head itself with the 'specifier' element. In fact, this could not be otherwise, as 'first Merge' does not exist, hence the head cannot form part of a complex element when 'second Merge' is applied.

To illustrate, the derivation of a sentence like *Who loves Mary?* runs roughly as follows (assuming, for concreteness, that AgrPs exist and subject-*wh*-phrases move to CP too):

(2) a. V + Obj
    b. Agr + Obj
    c. *v* + Subj
    d. T + Subj
    e. Agr + Subj
    f. C + Subj

Each Merge operation of (a–f) is licensed by Last Resort: (a) turns the verb that is uninterpretable because lacking an (internal) argument into a saturated, hence interpretable predicate (see Fn. 7); the same applies to (c), modulo the difference in the identity of the argument and the semantic/theta role played by it; finally, (b), (d), (e) and (f) all involve feature checking. Once again: as each Merge operation Merges a 'specifier' element in order to eliminate an uninterpretable property, each Merge operation is properly triggered. (I return

to the precise status of the 'specifier' elements below.)

Recall that we adopted the view that FSeq is checked syntax-externally, upon Spell Out. Notice that Spell Out domains/phases can be kept minimal in the present model: each of (2)(a–f) corresponds to a phrase, there being no first Merge. Thus, compliance with syntax-external requirements placed on FSeq can now be checked *immediately* after Merge, in principle (compare Fn. 8). But if there is no first Merge operation, then complementation can never be checked *internal* to one Merge operation (including a head and its complement). This, however, does not remove the possibility of checking the semantic requirements placed on FSeq. The functional complementation hierarchy is encoded in a derivation here in the *order* of the Merge operations themselves (in fact, it is also encoded in the very same order in the mainstream theory). All that needs to be assumed is that when the result of a Merge operation undergoes Spell Out (i.e., interpretation by the interface systems), then (as far as semantic interpretation is concerned) it is incrementally added to the semantic representation that has been constructed thus far. I take it that this assumption is nothing new: it is (explicitly or tacitly) shared by most 'derivational' models based on cyclic Spell Out. The only potential novelty here is that it is not the direct syntactic relation of any two syntactic elements that ensures the incremental growth of the semantic representation: this is simply the effect of the consecutive applications of Spell Out. If the Merge operations in (2) are not ordered according to the semantic requirements of the elements that get Merged and then Spelled Out immediately in a radically cyclic fashion, then the derivation crashes at once.

As a concrete example, assume that some semantic need dictates that Tense should directly apply to the semantic expression representing the eventuality, the latter being the semantic interpretation of (what is represented in standard phrase structure as) the *v*P. If after stage (2c) above we did not apply Merger of T (with some subject element), but (2f) (Merger applying to C), then Tense could not directly apply to the eventuality, the semantic representation of which was constructed when Spell Out applied immediately after (2c).

3.2 No Labels / Locus

This picture of the derivation not only takes care of the three complications caused by the 'first Merge' / 'second Merge' dichotomy, but it also has an immediate repercussion for the issue of labeling (or in Collins' model, the Locus). The present model entails that no labels are necessary as far as 'phrasal projection' is concerned. Crucially, a head never Merges with its complement in narrow syntax. Then, labeled nodes are never re-used as such in the course of building a phrase containing a specifier element. Labeled

nodes are also not accessed again in movement, assuming that movement is nothing more than Merger of another element, non-distinct in its composition from the lower, base occurrence; that is, if no copying operation is involved, as suggested in Bobaljik (1995), Epstein et al. (1998), Starke (2001), Gärtner (2002), Zhang (2004), *inter alia*. The identity or distinctness of the two DP occurrences for the purposes of interpretation (whether the two DPs form a 'chain' or not) is computed in semantics (see for instance, Brody 1998, Starke 2001, Epstein and Seely 2002), assuming that no indices or other syntax-internal devices exist to mark chains, by Inclusiveness (see Chomsky 1995, *et seq*.). If the constituent produced by Merge is never accessed again later, then labels are not necessary for syntactic computation, contra Chomsky (2005a).[9]

The eliminability of labeling is a welcome result, as labeling appears to be a source of complications: for one thing, labeling requires one to posit more complex structure than would be necessary if labels did not exist, and for another, even a simple generalization about which element projects as the label such as Chomsky's (2000) 'selector projects' has remained a stipulation. I hasten to add that not only we do not need labels, we do not need a separate memory buffer like the Locus either (as in Collins 2002). Given that in a sufficiently elaborate cartography of functional projections (as advocated by L. Rizzi, G. Cinque and others), each head bears only one uninterpretable feature, the sole function of this memory buffer for Collins (2002) is to warrant the contiguity of 'first' and 'second' Merge to the same head, and the contiguity of multiple instances of 'second Merge' (for instance, in multiple *wh*-movement constructions). As for the first of these two functions is concerned, no memory buffer for Locus is required, because there is no first Merge. As for the second function is concerned, the contiguity of multiple instances of 'second Merge' to the same head H is ensured by whatever semantic specification requires H to be in the position within the FSeq that it is. If, for example, a derivation had *Int* + WhP_1 followed by *Foc* + *FocusXP*, followed by *Int* + WhP_2, this would be ruled out by whatever semantic requirement forces the interrogative operator to be interpreted above Foc / FocusXP (see Rizzi 2004 for the FSeq segment Int > Foc). No Locus is required to derive these effects.

[9] Chomsky (2005a: 14) reasons that "labels, or some counterpart, are the minimum of what is required, on the very weak assumption that at least some information about a syntactic object is needed for further computation, both for search within it and for its external role." As his formulation reveals (cf. "on the very weak assumption"), Chomsky, I believe correctly, does not take this need for granted. Indeed, I contend in the present paper that there is no such need as far as the syntactic computation is concerned.

3.3 No projection and no constituents

In reality, the conclusions of the preceding subsection can (and should) be strengthened. If no head–complement units are created in narrow syntax, and if no later operation needs to access what correspond to phrases in a standard model, i.e., steps of the derivation involving head–specifier pairs as in (2) above, then there appears to be no need for Merge to actually involve projection at all. If Merge involves no projection, then this means in the present context that Merge does not actually construct a complex element. In other words, no constituents arise in the course of the derivation.

There being no syntactically complex elements constructed that could then be accessible for a later operation (i.e., no 'constituents'), it follows that Merge itself cannot apply to complex constituents either: only Lexical Items (LI) can undergo Merge. We do not need to formulate Merge as an operation yielding a syntactic object different from its input elements. No questions of what exactly this syntactic object looks like arise. Further, a potential concern about the degree of restrictedness of Merge is eliminated: given the way Merge functions in the mainstream derivational model of phrase structure, it is unavoidable to allow it to apply *both* to LIs and to sets of LIs (and sets of sets of LIs, etc.), which are distinct types of elements. In contrast, here Merge needs to apply exclusively to LIs.

These appear to be welcome results. If this approach can be maintained, then we have a system that fully adheres to the primacy of LIs advocated in the minimalist approach (see Chomsky 1993 et seq.), and conforms to a maximally strong form of Inclusiveness: no new syntactic objects are created, not even by set formation, in narrow syntax. I would like to argue next that the approach can be maintained when extended to movement.

I adopted the view in section 3.2 above that movement is nothing more than Merger of another element, non-distinct in its composition from the lower, base occurrence (see the references cited there). Then, the case of Merging in a moved element in the landing site position is not syntactically different from the Merger of the base occurrence. Both the base position and the landing site position are occupied by the LI that corresponds to the topmost head in the given phrase in a standard phrase structure. Consider (3)/(4) as an illustration of how this works.[10]

[10] For concreteness' sake, I am adopting AgrPs, and a split VP, and I am assuming demonstrative determiners to be specifiers of D, and the genitive 's to be D. Also, for the sake of simplicity, I am taking functional heads targeted by head movement to be morphophonologically empty, following Chomsky's strongly lexicalist approach. I represent such heads by category symbols here. Of course, none of this is crucial for the purposes of this paper.

(3) Which boy loves Pat's pizza?

(4) a. *n* + pizza
b. 's + Pat
c. loves + 's
d. Agr + 's
e. *n* + boy
f. D + which
g. *v* + which
h. T + which
i. Agr + which
j. C + which

Syntax is viewed here as involving the consecutive applications of an operation, which serves to eliminate uninterpretable properties of elements before they are sent to the interfaces (i.e., before they are Spelled Out). Merge, then, has become a misnomer: the operation simply relates two elements (appearing in (4) on the same derivational line), so that (at least) one of them should become interpretable. Merge merely serves the elimination of uninterpretability, before Spell Out is applied.[11]

Let us consider the nature of Merge a bit further, as its status appears to be crucial as far as the ultimate success of the present enterprise to work towards a purely derivational alternative is concerned. Merge has two basic functions: (i) it creates complex syntactic objects, and (i) given the discussion in section 1 above, it licenses checking of uninterpretable features (for the purposes of the present paper, specifier–head checking). As we have seen, the first of these two functions is eliminable. As for the second, Chomsky (2001, 2004) has argued that the notion of uninterpetabilitiy of features can be coherent only if uninterpretable features are checked (eliminated for semantic interpretation: valued and/or deleted) as part of the operation of Spell Out (Chomsky 2001: 5) (or Transfer). Besides the confounding issue of the 'visibility' of the property of uninterpretablity to the syntactic computation, a crucial argument for this view is that an uninterpretable feature can be interpreted phonologically even

[11] Let me note that a key result of the present enterprise, i.e., the lack of projections, displays unintended, and hence in my view particularly interesting, convergence with a central tenet of Brody's (2000) Mirror Theory (MT). Although MT does have constituents, crucially, it does not posit categorial projections (=the Telescope hypothesis). Thus, it appears that whether we adopt a purely representational model (like MT) or a purely derivational approach (like the present one), projections have no role to play in syntax.

though it is not interpreted semantically. This should mean, as Chomsky argues, that uninterpretable features are not eliminated as such *within* syntax, a conclusion I also embrace. Given that Merge does not involve the projection of a complex constituent, and given that 'feature checking' takes place as part of Transfer, and not under the Merge operation, Merge itself has no function in the present model. In effect, there is no operation that would apply to syntactic objects as input and would yield some other syntactic object as output. This result, in my judgment, is the cornerstone of the present endeavor to formulate a purely derivational approach to syntax. With no Merge, we no longer have syntactic representations, and we no longer have a duplication between the operation Merge and the syntactic relations (immediate containment and/or sisterhood) that it creates.

Although Merge does not appear to be necessary, some operation like Transfer does. Transfer is an operation that takes syntactic objects as input, and maps them onto phonological and semantic representations. Even though it is a conceivable option to stipulate that Transfer necessarily involves feature checking, this stipulation would rule out Transfer of a singular element, which does not seem to be desirable. This is because there apparently exist head elements that do not bear any uninterpretable property, and hence are not matched by a specifier element. A whole range of lexical complementizers exemplify such heads, and zero-place predicates instantiate the same option. In terms of the present model, such head elements, lacking an uninterpretable property, undergo Transfer on their own. The point at which they are introduced and immediately undergo Transfer is determined, as usual for the complementation hierarchy, by semantic needs. By the same token, a 'specifier' element can in principle also be on a derivation line on its own, unaccompanied by a 'head' element. Starke (2001) argues at length that this option is available, and in fact, preferable to the standard view, according to which a specifier element cannot exist without a head that it is hosted by. In Starke's terms: not only a head, but also a specifier element can project a functional phrase. In sum, we can assume that Transfer can apply to a singular, fully interpretable element. From this it follows that Transfer does not necessarily involve checking: it applies checking only if checking is possible.[12]

---

[12] As far as checking is concerned, I embrace Chomsky's (2001) conception of feature checking as valuation, rather than deletion, as in Chomsky (1995). As for Case-checking, I adopt Pesetsky and Torrego's (2001, 2004) approach, according to which structural Case is uninterpretable tense on D. In difference to Chomsky (2001), I am not making the extra assumption that valuation is necessarily symmetric in the sense that if α values some feature [F] of β, then it must be the case that β values some feature [G] of α (i.e., the valuation relation is asymmetric) (see also Note 15 for how this property might be

This conclusion has the consequence that Transfer cannot be stipulated to be inherently binary (taking exactly two syntactic objects as input). But this consequence does not seem to be borne out, insofar as it translates as freely allowing *n*-ary branching flat semantic structures. The question of the PF order of *n* elements on the same derivational line is also begged. I would like to argue that derivational lines with n>2 elements are ruled out not by Transfer itself, but by Linearization, which is a component of the grammar I have not yet specified. This is what I turn to next.

3.4 Linearization

Linearization at PF At a practical level, the algorithm for PF-linearization basically has two parts, if the view of a Universal Base stemming from Kayne (1994) is correct:

(5) a. Spec > Head
    b. Head > Compl

Brody (2000) argues that the way (5) (a) and (b) are derived in Kayne (1994) is not obviously better than stipulating them directly (see also Brody 1998). Pesetsky and Fox (2005) adopt precisely (5a) and (5b), as more or less direct linearization rules. In the present model we only need to assume a rule along the lines of (5a), and (5b) is dispensable as a separate linearization rule: (5b) is ensured by the way Spell Out maps onto a linear phonological representation. In particular, 'Head' precedes 'Complement' simply because it undergoes Spell Out 'later' than 'Complement' (where 'Complement' is understood here as the set of (head and specifier) elements that correspond to the set of nodes dominated by the complement on a standard account of phrase structure). No *new* assumption is introduced here: the idea that what is Spelled Out 'earlier' is pronounced 'later' is the way phonological mapping is defined in multiple Spell Out models in general (compare Epstein et al. 1998, where linearization tracks the application of Merge operations).[13]

---

explained). As a further departure from Chomsky's (2001) theory of checking, following *inter alia* Castillo et al. (1999), Bošković (2002a), Epstein et al (2005), and Epstein and Seely (2006), I am not assuming an uninterpretable EPP feature.

[13] Note that the proposal of this paper is neutral w.r.t. the direction of the derivation: a top-down derivational model (e.g., Phillips 1996, 2003, Richards 1999), where direction of derivation matches direction of pronunciation, can be implemented just as well. It can be considered a drawback of some top-down approaches that what is a simplex element (an LI) at one stage, may have to be turned into a complex element (a phrase) later; or what is in a complement position at a certain point, may have to be reanalyzed into the specifier of

Let me return to (5)(a) *Spec* > *Head* very briefly. As within the frame of the ideas entertained here only LIs appear in the syntactic derivation, and these LIs are not syntactically related to each other (only by Transfer, which applies checking to them if it is possible), the notions of 'specifier' and 'head' must be defined differently than in a standard phrase structure model.[14]

Let us restrict attention first, for the sake of simplicity, to cases involving only simple LI's as Spec elements. Given a PF representation that incrementally grows with every application of Transfer, and which is then phonologically spelled out in the inverse order of the applications of Transfer in a bottom-up approach (or in the order of the applications of Transfer in a top-down approach; see Note 11), what (5)(a) states can be conveniently reformulated as in (6) below (in a bottom-up approach).

(6) *Phonological Linearization*
If β checks α, then α is added to the phonological representation before β is added to that representation.

It follows from (6) that if there are two elements on a single derivational line DL to undergo Transfer, then DL is linearizable at PF only if one of them asymmetrically checks the other. If there are three elements X, Y and Z on a single DL, then there obtains exactly one scenario in which this DL is linearizable at PF: if X checks Y, and Y checks Z. Then (6) will demand that Z is added to the phonological representation first, then Y, and finally, X. The question of course is whether this option is actually instantiated in language. Interestingly, Grewendorf (2001) and Sabel (2001) propose an analysis of Bulgarian/Rumanian type multiple *wh*-fronting (see Rudin 1988, and Bošković 2002a for a typology of multiple *wh*-fronting) that is based on the assumption that one *wh*-element can check another. Sabel (2001) extends the same analysis to multiple *wh*-questions in Japanese, and to multiple focus

---

that complement position; in other words, derivations are not monotonic in the way bottom-up derivations are. In fact, implementing a top-down derivational approach within the frame of the present assumptions has the advantage for the top-down approach that it makes any reanalysis unnecessary.

[14] In fact, it is far from trivial how to state the *Spec* > *Head* linearization rule within Chomsky's (2001, 2004) model, which incorporates a bare phrase structure approach, but has no notion of specifier. One option would be to require a maximal-level projection of a category α to precede all non-maximal level projections of a distinct category β. This option, however, is only available in a model where Spell Out domains are sufficiently small: each phrase should be a phase, and multiple specifiers are not allowed.

Note that in Brody (2000) the linearization rule *Spec* > *Head* has a very different status, insofar as the relational notions of 'specifier' and 'head' are primitive, the specifier–head relation being a primitive relation in Mirror Theory.

movement in Malagasy. If indeed one element can serve as a checker of another element of the same type, then cases of multiple movements to the same functional head can plausibly instantiate the scenario of a DL in which X checks Y, and Y checks Z, etc. Such a treatment of multiple *wh-*/*foc-*/*neg-*/...-fronting to the same functional head F preempts the need for a delayed deletion approach to the uninterpretable feature on F (in terms of a distinction between erasure and deletion (see Chomsky 1995), or in some other way), or multiplying feature occurrences on F (e.g., multiple *wh*-features), or attributing [+multi]/[AttractAll] features to uninterpretable features themselves (e.g., Bošković 2002b).

To recapitulate, PF-linearization allows three kinds of derivational lines DL, in practical terms: (i) a singular 'head' or a 'specifier' element, (ii) a 'specifier' and a 'head', and (iii) multiple 'specifiers' of the same type with or without a 'head' element. Other cases would be non-linearizable by (6), as no linear (i.e., total) order would be produced.

Note that I have been assuming Transfer to incrementally grow not only the phonological representation, but the semantic representation too. Given that more than one element can appear on the same DL, the question is whether Transfer adds elements on the same DL to the current semantic representation simultaneously (creating a flat, multiple-branching semantic configuration). Within a strictly compositional semantic interpretation for natural language that is based on a typed lambda-calculus (e.g., Heim and Kratzer 1998 and references therein), Frege's Conjecture (that the elementary operation of semantic composition is function application, i.e., lambda-conversion), entails that semantic composition at each point involves the application of a single function to a single argument. If this is so, then some kind of ordering is necessary within the mapping by Transfer to semantic representations. Out of the three cases allowed by PF-linearization, only (ii) and (iii) are relevant, as only these involve more than one element on the same DL. These cases are generally assumed in a theory assuming a mainstream model of phrase structure to involve the following syntactic hierarchical relations (in terms of which element is Merged in a higher position; which is mapped to semantic representations with an isomorphic structure). First, 'specifier' is structurally higher than 'head,' and second, a 'specifier' S1 that is to the left of a 'specifier' S2 is structurally higher than S2. All this structure is not represented in the currently entertained approach in the syntax, but, if semantic composition invariably involves one function composing with one argument, then Transfer needs to add elements on the same DL not simultaneously, but in a certain order. In other words, not only the mapping to the phonological representation needs to follow a linearization rule, but the same is apparently also true of the mapping of multiple elements on a single

DL to the semantic representation. I formulate this latter linearization rule as below:

(7) *Semantic Linearization*
If β checks α, then α is added to the semantic representation before β is added to that representation.

An appreciable aspect of (7) is that it shows a striking symmetry with (6). This allows (in fact compels) one to formulate a single generalized rule of linearization that Transfer conforms to with respect to the mapping to both interface components:

(8) *Generalized Linearization*
If β checks α, then α is added to the interface representation before β is added to that representation.

Given that a rule of linearization is also part of mainstream derivational models, the present approach is superior in that mainstream derivational models employ both syntactic structure (representations) and linearization, while mine makes use only of the latter.[15]

Summarizing what has been said in this section so far, what Transfer does (besides implementing 'checking') is that it gives instructions to the (incremental) growth of phonological (PF) and semantic representations. The basic instruction that Transfer provides, partly through the order of its successive applications (constrained at the semantic interface by the semantic requirements of the elements involved), and partly through allowing Linearization to exploit the checking relations between elements undergoing one and the same application of Transfer (see (8)), lies in specifying the order in which elements are incrementally added to phonological and semantic representations.

---

[15] The asymmetry of the checking/valuation relation (defined over the set of LIs participating in a derivation) assumed in Note 12 is actually derived by (8). This is because, as pointed out immediately below (6), what follows from the linearization algorithm in (6), and its generalized form in (8), is a stronger property which entails asymmetry: the checking relation is necessarily antisymmetric, given that if A checks B and B checks A, A and B will not be linearizable. In this sense, antisymmetry of the checking relation is a consequence of the requirement of linearizability at the interfaces, assuming (8) as a linearization algorithm. On this view, antisymmetry is not a stipulated property of the checking relation: the checking relation is syntactically not restricted with respect to symmetry (i.e., it is asymmetric) (cf. Brody 2006). However, non-antisymmetric checking relations cannot result in a linearizable derivation.

Thus far, we have kept to the simple case, whereby all specifier elements are simplex (both with regard to their phonological and their semantic structure; i.e., they are comprised by a simple LI). It does not appear to be straightforward to accomodate complex specifier elements within a model, such as the one proposed here, which lacks syntactic representations. Recall, however, that what is claimed here is not that representations are inexistent in the grammar, but only that they do not have a role to play within narrow syntax. Given that incrementally growing phonological and semantic representations continue to exist, grammar should continue to be able to exploit them at the syntax/PF and syntax/semantics interface, respectively. Complex elements (made up of more than one LI) exist not because they are extant within narrow syntax, but because they are constructed in the PF, and in the semantic component, respectively.

Specifically, what I'd like to propose is a revision concerning the 'interpretation' of Transfer at the interfaces. So far we have been assuming implicitly that the consecutive applications of Transfer work according to (9), observing the Generalized Linearization Algorithm in (8).[16]

(9) *Transfer* (to be revised)
   a. Transfer (α), α a set of LIs λ, results in Transfer (λ) for all λ ϶ α.
   b. Transfer (λ) adds a phonological/semantic representation Λ (corresponding to λ) to an (already constructed) phonological/semantic representation P, yielding P′.

Adding Λ to P manifests itself as concatenation in the PF component, while it takes the form of function application in the semantic component. As noted immediately above, PF and semantics construct representations, and exploit representations already constructed. Indeed, P in (9b) is an (incrementally growing) representation, which—after the first application of Transfer—contains more than one LI. After each application of Transfer, the output representation P′ resulting after the previous Transfer operation is used in a recursive manner. I propose that it is not only the object that PF concatenation / semantic function application keeps adding elements to (i.e., P) that can be a representation larger than a single LI, but also the objects themselves that are added (by PF concatenation / semantic function application) to P (i.e., Λ) can contain more than a single LI. Accordingly, but still keeping to the simplest

[16] As far as I can see, distributivity as in (9a) is a property of the mapping from syntax to the interface components in all current minimalist models: if a syntactic expression α is mapped to the interface components, then this involves the mapping of every LI contained in α.

hypothesis that narrow syntax accesses LIs only, I suggest that (9b) is to be modified as in (10) (NB. containment is reflexive).

(10) *Transfer (λ)*
Transfer (λ) adds a phonological/semantic representation Λ′ containing Λ (Λ corresponding to λ) to an (already constructed) phonological/semantic representation P, yielding P′.

This interpretation of Transfer in the interface components produces derivational segments like the following for complex Spec elements. As an example, consider the subject DP *John's mother* in the specifier position of a TP headed by the modal auxiliary *will*, in the sentence *John's mother will visit us*. Consider the stage where the PF and semantic representations of *visit us* are already constructed. Then, the consecutive applications of Transfer in (11a,b,c) result in the parallel growth of the PF representation in (12a,b,c), and the growth of the compositional semantic representation in (13a,b,c), respectively. (The order of LIs within the brackets reflects the direction of checking between them, for reasons of expository convenience.)

(11) a. Transfer (*mother*)
b. Transfer (*John*, *'s*)
c. Transfer *('s*, *will*)

(12) a. mother
b. John 's mother
c. John 's mother will *visit us*

(13) a. [mother]
b. [John [['s] [mother]]]
c. [[John [['s] [mother]]] [will [*visit us*]]]

Note that to integrate a complex specifier into the PF/semantic representations, an LI appears on two derivational lines (in our case, this LI is the D element *'s*). Given the way Transfer is interpreted by the interface components in (10), this does not entail, nevertheless, that the LI in question must itself appear twice in the PF/semantic representation. For instance, (11c) (or more precisely, the second step it involves in the mapping to the interface representations, in accordance with (8)) does not result in inserting a second instance of the genitive determiner, but rather, it results in adding a PF/semantic representation containing the genitive determiner itself (viz. *John's mother*) to another (viz. *will visit us*, which is created as a first step on

the basis of (11c)). This choice ensures that the previously created representation *John's mother* is integrated into the PF/semantic representation through being combined with *will visit us*.[17]

The same mechanism extends to cover the generation of a syntactically complex element in a landing site position in the case of movement. In derivations involving movement of *John's mother*, an operation Transfer (*'s*, λ_1) will be followed by Transfer (*'s*, λ_2); in the case of a more complete analysis than the one given in (11–13), λ_1 is an LI of category *v*, and λ_2 is an LI of category T (here: *will*). The most straightforward assumption with regard to the interpretive components is that movement of *John's mother* involves two subderivations of the type (11–12): one for the pre-movement occurrence, and one for the landing site occurrence. More generally, movement of a constituent K then involves deriving K twice, and merging in the two occurrences into the pre- and post-movement positions, respectively, an approach that has been proposed before (e.g., Brody's 1998 'distributed chains').[18] On such an account, interface operations and/or semantic principles are responsible both for the interpretation of the two instances of the 'moved' syntactic object (SO) as members of a 'chain,' i.e., as a particular kind of semantic dependency involving one thematic element, and also for any

---

[17] Assuming that human grammar seeks to reduce operational complexity to its minimum (see Chomsky 2001, 2004; a view naturally matched by the notion of a transparent parser, cf. Berwick and Weinberg 1984, Mulders 2002; see also Phillips 1996, 2003 for the view that parser = grammar), the following principle limiting operational complexity can be adopted. (i) relates closely to what is commonly referred to as the immediate attachment principle in parsing (see Schneider 1999 and references there). Upon the application of (11c), (i) enforces the integration of the LI *will* with the previously constructed representation [visit us], and the previously constructed representation (13b) with [will [visit us]].

(i) Minimize the number of yet unintegrated representations in the working space.

[18] The copy theory of movement (cf. Chomsky 1993, Nunes 2004) is similar in having two distinct tokens of a syntactic object SO in two positions, but it differs from the 'multiple merge' approach in not having two identical subderivations of the moving SO. Note that the copy theory involves an extra operation, Copy, which applies to a complex SO α (or, depending on the formulation, to constituent containing this SO), and yields another token of α. The 'multiple merge' operation simply exploits Merge itself, and holds that the property of identity is a requirement imposed by the 'chain' interpretation in the semantic component. Unless some further device like indexation or some equivalent is introduced, identity (in the sense of 'chain') is not expressed syntactically within the copy theory either, i.e., nothing is gained by applying the Copy operation in this regard (Copy+Merge creates a representation that is indistinguishable from the corresponding representation created by a 'multiple merge' derivation). As complex SOs do not exist in the model developed here, the copy theory of movement cannot be integrated into it in any meaningful way. See Brody (1998) for further discussion.

adjustments made to this SO prior to interpretation (as in 'reconstruction,' 'trace conversion,' partial or complete phonological deletion etc.). The 'chain' interpretation requires identity of the two instances of SO, which is the case only if the two syntactic subderivations yielding SO are identical.[19]

It is interesting to note that the alternative commonly referred to as the Remerge approach is unformulable within the frame of assumptions I have proposed. Epstein et al. (1998), Gärtner (1999, 2002), Wilder (1999), Kracht (2001), Starke (2001), Chomsky (2004; cf. 'internal merge') and Zhang (2004), *inter alia*, have suggested (albeit in different forms) that in movement the very same object is affected by the syntactic combinatory operation (viz. Merge) more than once (resulting in structural multidominance in traditional terms of constituent structure).[20] Implementing this through the operation of Transfer as defined in (10), Remerge (i.e., re-Transfer) of an element λ requires a semantic object Λ′ containing Λ (corresponding to λ) *that is part of a semantic object P* to combine semantically with P itself. This plainly does not yield any well-formed semantic object (unless Remerge (i.e., re-Transfer)

[19] This type of approach (call it 'multiple merge' approach) makes available a rather straightforward characterization of the so-called 'Lebeaux-effect' in adjunct reconstruction (for discussion and references, see Chomsky 1993, 1995): details aside, what needs to be assumed is that the identity requirement of the two instances of SO does not extend to the external adjuncts of SO. This construal of Lebeaux-effects, which is unavailable without extra assumptions both within the copy theory and within the Remerge theory of movement, involves a scenario where the SO in the landing site position includes an external adjunct, while the same adjunct is absent from the SO in the base position.

[20] As Chomsky's (2005) notes in passing, there appears to have been a terminological confusion around his notion of copying/Remerge/internal Merge, which he has meant ever since his first programmatic minimalist paper to be interpreted as merging an element E with a category K that properly contains E. To be sure, there still remain some elusive aspects to this operation. What seems sufficiently clear is that if it is part of the grammar, the concept of internal Merge enforces a mixed derivational/representational model of syntax, since without a representation(al memory) no element is retained until a later derivational point, when it is re-merged. What has remained unclear is how remerge *qua* internal Merge is formalized in terms of Chomsky's (2001, 2004) set theory notation (if internal Merge is to be conceptualized along the lines of Starke (2001)). Merge(A,B) corresponds to set formation yielding {A, B}. If so, then given a category C={A, B} properly contained in the root category R, if B is to be re-merged to R, then this should yield {R, B}. It follows that this latter token of B and the one immediately contained by C cannot be identical, i.e., they cannot be the same token. That means that movement cannot simply be 're-merging the same element' again. Internal Merge is not only undefinable in Chomsky's set theory notation. As Brody (2005) argues (simplifying somewhat), internal Merge is also undefinable once the primitive structural relation is not taken to be immediate containment, as Chomsky's Merge-based model has it, but containment. This is because a structure that involves an element E that has undergone internal Merge (i.e., movement) is indistinguishable from a structure in which E has not undergone this operation (whence a model based on the primitive of containment is more restrictive w.r.t. generative power, and therefore preferable).

involved different tokens of Λ, in which case what we have is not a Remerge-based theory, cf. note 20)).

4 Conclusion

In this paper, I have followed the main thrust of Epstein and Seely's (2002) as well as Brody's (1998, 2002, 2006) efforts to rid narrow syntax of any residual redundancy it incorporates. In this sense, the present enterprise falls into the category of 'eliminative minimalism' (see Epstein and Seely 2006), a category that forms a proper part of 'methodological minimalism' (Martin and Uriagereka 2000).

In particular, I have highlighted several complications for 'first Merge' ((i) it has no proper trigger, (ii) it preempts direct Merger of head with specifier, and (iii) its order w.r.t. 'second Merge' is unexplained), and went on to propose that 'first Merge' does not exist. As a consequence of this proposal, projection (or a memory buffer of Locus) and constituents themselves have become dispensable in a derivational approach. The resulting picture then is a strongly derivational approach, which is characterized by a lack of projections, a lack of constituents, and adherence to a strong form of Inclusiveness, keeping to the primacy of LIs in narrow syntax. The model outlined here has no Merge operation, and posits no syntactic representations either. It is purely derivational in the sense that the operation Transfer accesses LIs in a cyclic fashion, providing appropriate input to both phonology and semantics. Transfer implements checking, and follows a Linearization rule that is generalized across the two interface components. Importantly, recursion is no longer part of narrow syntax: it is a property of the external interface systems, which continue to be representational components.

The present exploration of what redundancies current 'radically derivational' theories incorporate on a microscopic scale contributes its share to the representations vs. derivations debate. Needless to say, what the present paper offers is only a rough outline of the direction to take. Nevertheless, to the extent that the approach I have suggested can successfully eliminate syntactic representations, it yields two important results. First, it demonstrates that the challenge posed for so-called 'radically derivational' models by Brody (1995) and especially Brody (2002) is real: current mainstream derivational theories are ontologically richer than a purely representational theory, and without compelling empirical motivation. Second, it offers a plausible path to follow if this challenge is to be met.

References

Basilico, David. 1998. Object position and predication forms. *Natural Language and Linguistic Theory* 16: 541–595.

Berwick, Robert C., Amy S. Weinberg. 1984. *The Grammatical Basis of Linguistic Performance*. Cambridge, MA: MIT Press.

Bobaljik, Jonathan. 1995. In Terms of Merge: Copy and Head Movement. *MIT Working Papers in Linguistics* 27, 41–64.

Bošković, Željko. 2002a. A-movement and the EPP. *Syntax* 5: 167–218.

Bošković, Željko. 2002b.On Multiple Wh-Fronting. *Linguistic Inquiry* 33: 351–383.

Bošković, Željko. 2005. On the Locality of Move and Agree: Eliminating the Activation Condition, Generalized EPP, Inverse Case Filter, and Phase-Impenetrability Condition. *University of Connecticut Occasional Papers* 3.

Brody, Michael. 1995. *Lexico-Logical Form. A radically minimalist theory*. Cambridge, MA: MIT Press.

Brody, Michael. 1997. Perfect chains. In: Liliane Haegeman (ed.), *Elements of Grammar*. Dordrecht: Kluwer. 139–167.

Brody, Michael. 1998. Projection and Phrase Structure. *Linguistic Inquiry* 29: 367–398.

Brody, Michael. 1999. One more time. *Syntax* 4: 126–138.

Brody, Michael. 2000. Mirror Theory: Syntactic representation in Perfect Syntax. *Linguistic Inquiry* 31: 29–56.

Brody, Michael. 2002. On the Status of Derivations and Representations. In: Samuel David Epstein and T. Daniel Seely (eds.), *Derivation and Explanation in the Minimalist Program*. Oxford: Blackwell. 19–41.

Brody, Michael. 2006. Syntax and symmetry. Ms. Lingbuzz internet archive, April 2006.

Castillo, Juan Carlos, John Drury, and Kleanthes Grohmann. 1999. Merge-over-move and the EPP. In *University of Maryland Working Papers in Linguistics* #8. University of Maryland, College Park.

Chomsky, Noam. 1993. A minimalist program for linguistic theory. In: Kenneth Hale and Samuel Jay Keyser (eds.), *The view from Building 20: Essays in linguistics in honor of Sylvain Bromberger*. Cambridge, Mass.: MIT Press. 1–52.

Chomsky, Noam. 1995. Categories and transformation. In: *The Minimalist Program*. Cambridge, MA: MIT Press. 219–394.

Chomsky, Noam. 2000. Minimalist Inquiries: the Framework. In Roger Martin et al. (eds.), *Step by Step*, Cambridge, MA: MIT Press. 89–155.

Chomsky, Noam. 2001. Derivation by Phase. In Michael Kenstowicz (ed.), *Ken Hale: A Life in Language*. Cambridge, Mass.: MIT Press. 1–52.

Chomsky, Noam. 2004. Beyond Explanatory Adequacy. In Andrea Belletti (ed.), *Structures and Beyond: The Cartography of Syntactic Structures,* Vol. 3. New York: Oxford University Press. 104–131.

Chomsky, Noam. 2005a. Three factors in language design. *Linguistic Inquiry* 36: 1–22.

Chomsky, Noam. 2005b. On phases. Ms. Cambridge, Mass., MIT.

Cinque, Giulielmo. 1999. *Adverbs and Functional Heads: A cross-linguistic perspective*. Oxford: Oxford University Press.

Collins, Chris. 2002. Eliminating labels. In: Samuel David Epstein and T. Daniel Seely (eds.). *Derivation and explanation in the minimalist program*. Oxford: Blackwell. 42–61.

Epstein, Samuel David. 1999. Un-Principled Syntax and the Derivation of Syntactic Relations. In: Samuel David Epstein and Norbert Hornstein (eds.), *Working Minimalism*. Cambridge, Mass.: MIT Press. 317–345.

Epstein, Samuel David, Erich M. Groat, Ruriko Kawashima, Hisatsugu Kitahara. 1998. *A Derivational Approach to Syntactic Relations*. Oxford: Oxford University Press.

Epstein, Samuel David and T. Daniel Seely. 2002. Rule Applications as Cycles in a Level-Free Syntax. In: Samuel David Epstein and T. Daniel Seely (eds.), *Derivation and explanation in the minimalist program*. Oxford: Blackwell. 65–89.

Epstein, Samuel David, Acrisio Pires and T. Daniel Seely. 2005. EPP in T: More controversial subjects. *Syntax* 8: 65–80.

Epstein, Samuel David and T. Daniel Seely 2006. *Derivations in Minimalism*. Cambridge: Cambridge University Press.

Ernst, Thomas. 2002. *The Syntax of Adjuncts*. Cambridge: Cambridge University Press.

Frampton, John and Samuel Gutmann. 1999. Cyclic computation, a computationally efficient minimalist syntax. *Syntax* 2: 1–27.

Fox, Danny and David Pesetksky. 2005. Cyclic linearization of syntactic structure. *Theoretical Linguistics* 31: 235–262.

Gärtner, Hans-Martin. 1999. Phase-linking Meets Minimalist Syntax. In: Roger Billerey and Brook D. Lillehaugen (eds.), *The Proceedings of WCCFL* 18. Somerville, MA: Cascadilla Press. 159–169.

Gärtner, Hans-Martin. 2002. *Generalized Transformations and Beyond: Reflections on Minimalist Syntax*. Berlin: Akademie Verlag.

Grewendorf, Günther. 2001. Multiple Wh-Fronting. *Linguistic Inquiry* 32: 87–122

Heim, Irene, and Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Oxford: Blackwell.

Kayne, Richard. 1994. *The Antisymmetry of Syntax*. Cambridge, MA: MIT Press.

Kracht, Marcus. 2001. Syntax in Chains. *Linguistics and Philosophy* 24: 467–529.

Kratzer, Angelika. 1996. Severing the External Argument from Its Verb. In: Johan Rooryck and Laurie Zaring (eds.), *Phrase Structure and the Lexicon*. Dordrecht: Kluwer. 109–138.

Koeneman, Olaf. 2000. The flexible nature of verb movement. PhD dissertaion, Utrecht University.

Martin, Roger and Juan Uriagereka. 2000. Introduction. In: Roger Martin, David Michaels and Juan Uriagereka (eds.), *Step by Step.* Cambridge, MA: MIT Press

Mulders, Iris. 2002. Transparant Parsing. Head-driven processing of verb-final structures. PhD dissertation, University of Utrecht.

Müller, Gereon. 2004. Phrase Impenetrability and Wh-Intervention. In: Arthur Stepanov, Gisbert Fanselow and Ralf Vogel (eds.), *Minimality Effects in Syntax.* Berlin: Mouton/de Gruyter. 289–325.

Nilsen, Oystein. 2003. Eliminating Positions. Ph.D. dissertation, Utrecht University.

Nissenbaum, Jon. 2000. Investigations of covert phrasal movement. Ph.D. dissertation, MIT.

Nunes, Jairo. 2004. *Linearization of Chains and Sideward Movement*. Cambridge, MA: MIT Press.

Pesetsky, David. 2000. *Phrasal Movement and Its Kin*. Cambridge, MA: MIT Press.

Pesetsky, David and Esther Torrego. 2001. T-to-C movement: Causes and consequences. In: Michael Kenstowicz (ed.), *Ken Hale: A Life in Language.* Cambridge, MA: MIT Press. 355–426.

Pesetsky, David and Esther Torrego. 2001. Tense, case and the nature of syntactic categories. In: Jacqueline Guéron and Jacqueline Lecarme (eds.), *The Syntax of Time*. 495–537.

Phillips, Colin. 1996. Order and Structure. PhD dissertation, MIT.

Phillips, Colin. 2003. Linear order and constituency. *Linguistic Inquiry* 34: 37–90.

Richards, Norvin. 1999. Dependency Formation and Directionality of Tree Construction. In: Vivian Lin, Cornelia Krause, Benjamin Bruening and Karlos Arregi (eds.), *Papers on*

Morphology and Syntax, Cycle Two. MIT Working Papers in Linguistics #34. Cambridge, MA, MIT.
Rizzi, Luigi (ed.). 2004. *The Structure of CP and IP. The cartography of syntactic structures*. Vol. 2. Oxford: Oxford University Press.
Rudin, Catherine. 1988. On multiple questions and multiple WH-fronting. *Natural Language and Lingusitic Theory* 6: 445–501.
Sabel, Joachim. 2001. Deriving Multiple Head and Phrasal Movement: The Cluster Hypothesis. *Linguistic Inquiry* 32: 532–547.
Schneider, David. 1999. *Parsing and incrementality.* PhD dissertation, University of Delaware.
Sportiche, Dominique. 1999. Reconstruction, constituency and morphology. GLOW handout, Berlin.
Starke, Michal. 2001. Move dissolves into merge: a theory of locality. PhD dissertation, University of Geneva.
Surányi, Balázs. 2002. Multiple Operator Movements in Hungarian. PhD dissertaion, Utrecht University (published by LOT).
Surányi, Balázs. 2006. Cyclic Spell Out and head movement. Ms., Research Institute for Linguistics, HAS, Budapest.
Uriagereka, Juan. 1999. Multiple Spell-Out. In: Samuel David Epstein and Norbert Hornstein (eds.), *Working Minimalism*. Cambridge, MA: MIT Press. 251–282.
Wilder, Chris. 1999. Right Node Raising and the LCA. In: Roger Billerey and Brook D. Lillehaugen (eds.), *The Proceedings of WCCFL* 18. Somerville, MA: Cascadilla Press. 586–598.
Zhang, Nina. 2004. Move is Remerge. *Language and Linguistics* 5: 189–209.

Balázs Surányi
Research Institute for Linguistics, Hungarian Academy of Sciences
suranyi@nytud.hu