A derivational view on movement constraints

Christina Unger*

This paper explores conditions on movement in a strongly derivational organization of grammar. The setting is Ulf Brozsiewski's model of syntactic derivations [4]. It offers a local encoding of movement dependencies that combines properties of transformational and feature-based approaches. It is based on simple assumptions about syntactic expressions and the operations defined on them, and is thus suitable to investigate how much is needed to derive, or at least express, fundamental characteristics of movement. Moreover it reduces the amount of representations needed for syntactic operations to a minimum and thus constitutes an exploration of how derivational syntax can be.

The outline of the paper is the following. In section 1, Brosziewski's view on derivations will be introduced. After that, I address a problem that it faces with extraction from phrases that are extracted themselves, and show how to solve it. Then I will turn to generally assumed constraints on movement. In section 3, I review how the Condition on Extraction Domain is expressed in Broszïewski's model. Then, in section 4, I will consider how weak islands can be captured. Furthermore, section 5 deals with accounting for minimality effects. Finally, in section 6, I will also show how across-the-board extraction can be incorporated.

1 On derivations

1.1 Syntactic expressions

In approaches like HPSG and the Minimalist Program, features play a central role in constructing and manipulating syntactic structures. Broszïewski's model of syntactic derivations shares this assumption. Simple expressions are taken to be feature bundles containing phonological, syntactic, and semantic features. In the following, phonological and semantic features will be ignored and focus will be on syntactic features only. With respect to these, Broszïewski sticks to a very simple feature system. For encoding syntactic dependencies he assumes that features come in two varieties: as plain features f, and as features $^\wedge f$ with a prefix that indicates that it wants to be matched with a corresponding plain feature. Following Adger [2] a. o., I add a distinction between two types

[0]For thorough remarks and helpful suggestions, I am very grateful to Gereon Müller, Eric Reuland, and Andreas Pankau.

of features. The first one are category features Cat; they subsume subcategorization features, that are checked when two expressions are merged and that are assumed to be ordered according to the argument structure of the selecting expression. The second one are morphosyntactic features $Form$, that trigger movement and require no ordering. Although this distinction is not made explicitely by Brosziewski, the presentation of features here and relations based on them later on does not differ in an essential way from his view in [4, Section 2.2].

Definition 1. *The set of syntactic features is given by:*

$$\begin{aligned} Cat &::= B \mid {}^\wedge B \\ Form &::= F \mid {}^\wedge F \end{aligned}$$

$$\begin{aligned} B &::= V \mid N \mid P \mid v \mid T \mid C \\ F &::= wh \mid case \mid \dots \end{aligned}$$

In the following I will assume that matching features f and ${}^\wedge f$ are deleted when two expressions merge, and that undeleted syntactic features at the interfaces to phonology and semantics cause a crash of the derivation. However, there is nothing deep in this view on features. In fact a lot of phenomena would call for a more sophisticated theory of features, especially because the combinatorial rules for merge and movement are quite primitive. But since I am mainly interested in core aspects of syntactic operations and restrictions on them, I will leave the feature system as simple as possible.

Based on features as basic building blocks, syntactic expressions are defined as either being simple feature bundles, or being complex expression built of other syntactic expressions. Simple expressions are triples containing a phonological representation phon, an ordered list of category features and an unordered list of formal features.[1] Complex expressions are pairs of expressions.

Definition 2. $Exp ::= (\mathsf{phon}, [Cat], [Form]) \mid \langle Exp, Exp \rangle$

I use round brackets for triples and angled brackets for tuples just to make it easy to distinguish simple and complex expressions at first sight. Moreover, in the following I will use α, β, γ as variables for simple expressions, and x, y, z as variables for arbitrary expressions.

Examples of simple expressions are the following.

(1) $(\mathsf{Ulf}, [N], [{}^\wedge case])$

(2) $(\mathsf{punch}, [{}^\wedge N, V], [\,])$

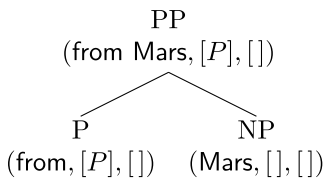
[1] As mentioned before, expressions may also carry a semantic interpretation, which is omitted here.

(3) $(\epsilon, [{}^\wedge V, {}^\wedge N, v], [case])$

(1) constitutes an NP, which has a categorial feature N and a formal feature ${}^\wedge case$ that expresses that it still requires case to be assigned. A transitive verb would look like in (2). It is of category V and additionally subcategorizes for its internal argument of category N. The v-head as in (3) subcategorizes for a VP and an external argument of category N, and also carries a formal feature $case$ expressing that it can assign case. Its phonological content is empty, which is represented by ϵ.

The idea behind simple and complex expressions is the following. Derivations are assumed to not build phrase structures. When simple expressions are merged together, these expressions are combined into another simple expression, whose features and semantic value are a combination of the features and semantic values of the daughters. Since no structure is built, information about the expressions that were combined and possible structural configurations is forgotten. As an example, consider we want to merge two expressions $(\textsf{from}, [{}^\wedge N, P], [case])$ and $(\textsf{Mars}, [N], [{}^\wedge case])$. Assuming that when merging them not only the categorial but also the case features are checked, the two expressions will end up inactive in the sense that they do not carry any features that still require checking. In the further derivation, it will not be necessary to look into the structure (4) but it can rather be treated as one constituent. All that is necessary to keep is the simple expression $(\textsf{from Mars}, [P], [\,])$, because it contains all the relevant information.

(4)



The same holds if a whole CP like (5) was built. Everything inside the CP is inactive, so the subexpressions will be inaccessible for further operations and they and their structural configurations will not play a role in the derivation anymore. They can therefore be discarded; everything that syntactic operations need to care about in the further derivation is the root of the tree corresponding to the simple expression (6).

(5) $[_{\text{CP}}\ C\ [_{\text{TP}}\ \textsf{Mary}_1\ [T\ [_{\text{vP}}\ t_1\ [_{\text{v'}}\ v\ [_{\text{VP}}\ \textsf{punches}\ [_{\text{NP}}\ \textsf{a unicorn}]]]]]]]$

(6) $(\textsf{Mary punches a unicorn}, [C], [\,])$

These simple expressions behave like lexical items in the sense that syntactic operations do not have access to their parts or their structure – similar to the objects obtained after Spell-Out in Uriagereka [23].

However, the situation is different in case there are expressions that have to check formal features at a later stage of the derivation, i. e. have to move[2], like the *wh*-pronoun in What does Mary punch?. When merging the verb punch with its internal argument what, we cannot build another simple expression forgetting about the subexpressions it was built from, because what is not inactive yet; it still has a *wh*-feature to check later in the derivation and therefore needs to be accessible until that later point. To keep information that still plays a role and thus cannot be forgotten, there will be a mechanism to copy expressions. One copy will be merged in base position (where it contributes its categorial feature for fulfilling subcategorization requirements), and one copy will be kept as the first element of a pair and carried along until it reaches the landing site. Merging punch and what will yield the complex expression

(7) $\langle(\mathsf{what},[\,],[wh]),(\mathsf{punch},[V],[\,])\rangle$

Later in the derivation, when the C-head enters the derivation, what can check its *wh*-feature and will be merged into a simple expression.

In the following I adopt Brosziewski's notation. This means that (7) will be abbreviated as $\langle\mathsf{what}*,[\mathsf{punch}\ c_{\mathsf{what}}]\rangle$. Simple expressions are not written as feature bundles but as strings, with c representing the (phonologically empty) copy staying in base position, and marking the percolating copy with an asterisk. For ease of reading, an expression $(\mathsf{foo},[X],[\,])$ will often be written as $[_{\mathrm{X}}\,\mathsf{foo}]$ or $[_{\mathrm{XP}}\,\mathsf{foo}]$. Relevant features will sometimes be indicated by superscripts.

A complex expression $\langle\alpha_1,\langle\alpha_2\ldots\langle\alpha_n,\beta\rangle\rangle\rangle$ can be seen as having built a syntactic expression β together with a list (or stack) of expressions $\alpha_1,\alpha_2,\ldots,\alpha_n$, that were extracted from β, that are now percolated, and that are going to be remerged at a later point of the derivation. β will be called the *nucleus* of the pair and sometimes I will refer to the α_i as being *at the edge* of the complex expression (this notion will be made explicit in Definition 6 in Section 2). In modern Chomskian terms the α_i in $\langle\alpha_1,\langle\alpha_2\ldots\langle\alpha_n,\beta\rangle\rangle\rangle$ could also be seen as being at the edge of a phase, while everything contained in β was sent to the interfaces. I will not go into details of this comparison here. I will turn to a short discussion of the relation between Brosziewski's implementation of unbounded dependencies and the successive-cyclic implementation in Phase Theory at the end of the section. But first I want to demonstrate the mechanism for merging and moving expressions.

The operations of merging and moving are driven by the need of expressions to check features. This will be expressed in their sensitivity to two relations, subcategorization and licensing, that can be defined on the basis of the two types of syntactic features.

Definition 3. *An expression x* subcategorizes *for an expression y if x has a*

[2]I solely concentrate on wh-movement, but in principle nothing prevents one from applying Brosziewski's mechanism to other long-distance phenomena as well, such as long-distance agreement (which is treated at length in other contributions to this volume).

subcategorization feature $^\wedge b$ and y carries the corresponding categorial feature b.

Definition 4. *An expression x* licenses *an expression y if x has a formal feature $^\wedge f$ and y carries the corresponding formal feature f, or vice versa.*

These definitions are general in that they apply to both simple and complex expressions. In order to be applicable to complex expressions, we need to specify what the syntactic features of a complex expression are. At this point remember that in a complex expression $\langle x, y\rangle$, y is the root of the structure built so far, whereas x was merged somewhere in y and is just carried along. Thus $\langle x, y\rangle$ should be subcategorized or licensed by some expression if y is subcategorized or licensed by it. To this end, we say that the syntactic features of a complex expression $\langle x, y\rangle$ are the syntactic features of y.

Now with these two relations it is possible to specify the notions of being a complement, specifier, or adjunct. Broszіewski assumes that specifiers are a special case of adjuncts (cf. [4, p. 25f]).

Definition 5. *Let α and β be two expressions that are merged. Then β is a* complement *of α if α subcategorizes for β. Otherwise, β is an* adjunct*. Furthermore, an adjunct β is a* specifier *of α if α licenses β.*

1.2 Operations

There is one general combinatoric function, **merge**, that combines two expressions into a third one. It is defined for simple and complex expressions and thus is independent of whether movement takes place or not. We will look at all possible cases in turn. In case no movement is involved, two simple expressions are merged. This will be handled by (M1) below. In case movement takes place, a complex expression is merged with either a simple or another complex expression. These cases will be covered by (M2) and (M3) below.

Let us start with the case of merging two simple expressions, i. e. the case where no movement is involved. The result will also be a simple expression.

(M1) **merge** $\alpha\ \beta = \gamma$
where γ has the same syntactic features as α except those that were discharged by β, and the phonological representation of γ is derived from the phonological representations of α and β

One example that already appeared above was merging a preposition with an NP it subcategorizes for, here repeated as (8). The features of the preposition that are discharged by the NP are $^\wedge N$ and *case*, so only P projects.

(8) **merge** $(\mathsf{from}, [^\wedge N, P], [case])\ (\mathsf{Mars}, [N], [^\wedge case])$
$= (\mathsf{from\ Mars}, [P], [\,])$

A case where no subcategorization features are checked is adjunction, as in (9).

(9) **merge** (scientists, $[N], [^\wedge case]$) (from Mars, $[P], [\,]$)
    = (scientists from Mars, $[N], [^\wedge case]$)

Now let us turn to the encoding of movement dependencies. Movement rests on the same basis as dislocation in GPSG ([9],[10]), namely on dividing the relation between base position and landing site into three parts: *bottom*, where the dependency is introduced and encoded (in our case, where the expression enters the derivation), *middle*, where the information about the dependency (in our case, a copy of the expression) is percolated, and *top*, where the dependency is finally established (in our case, the expression is remerged).

Let us first look at *bottom*. The motivation for a movement mechanism is that some expressions play a double role – they fill an argument position as well as check features somewhere else in the structure. So one possible assumption, which is adopted by Brosziewski, is that there is a derivational mechanism that allows to duplicate expressions in order to let them be merged at the two relevant points of the derivation. This mechanism is a function for copying elements in base position where they start the movement. This operation is assumed to be optional.

(C1) *Copying*
    An expression y can be copied in (**merge** x y).

However, copying here does not mean duplicating an expression or features in the literal sense; copying rather splits an expression by creating two new expressions among which the features of the splitted expression are distributed complementarily. One part will stay in base position, whereas the other part will be transported to the landing site. The definition of the function **copy** has the following form:

(C2) **copy** $\alpha = \langle \alpha *, c_\alpha \rangle$

For example, when copying a wh-expression, the expression being merged in base position should contain the categorial feature in order to fulfil subcategorization requirements in base position, while the expression which will be percolated, should carry the *wh*-feature and the phonological representation to the landing site. Copying a wh-pronoun would then look like in (10).

(10) **copy** (who, $[N], [wh]$) $= \langle$(who, $[\,], [wh]$), $(\epsilon, [N], [\,])\rangle$

The question now is how the features are distributed among the copies in general. To start with, categorial features of the copied expression clearly have to resume in base position in order to satisfy the subcategorization requirements of the expression that subcategorize for it. Formal features, however, are features that trigger movement and will be needed later in the derivation. So they should be carried along, unless they can already be checked in base position. This is summarized as follows.[3]

(FD) *Feature Distribution*
All syntactic features of α that can be projected or that can be checked with the element with which α is merged (via subcategorization or licensing) are assigned to the nucleus of (**copy** α).

In other words, features that can be projected or licensed in their base position cannot enter into movement processes.

Now let us turn to the *middle* of a movement dependency, the percolation of copies. Since nothing is supposed to happen with these copies while they are carried along, their percolation can be handled by **merge**. All this operation is supposed to do is combine expressions as specified in (M1) and in doing so simply ignore the percolating copies. However, it cannot quite do that up to now, because in (M1) **merge** was only defined for simple expressions. In order to handle the percolation of extracted expressions, it also has to be defined for complex expressions. This will be done in the definitions (M2) and (M3).

Let us start with (M2). The idea behind it is that if an expression α is merged with a pair $\langle x, y\rangle$, then this merging is actually about the nucleus y and not about the percolating copy x, so α combines with y while x is simply retained at the edge of the pair. The following definition does exactly that (it will be slightly modified later, in section 3):

(M2) (preliminary version)
$\textbf{merge}\ \alpha\ \langle x, y\rangle = \langle x, \textbf{merge}\ \alpha\ y\rangle$

As an example consider again the VP from above, repeated in (11), where the internal argument of the verb punch is a wh-phrase that was copied in order to check its *wh*-feature later in the course of the derivation. (Let us ignore case assignment for the moment.)

(11) $\langle \textsf{what}*, [_{\text{VP}}\ \textsf{punch}\ c_{\textsf{what}}]\rangle$

When this complex VP is selected by a v-head, (M2) applies. This means that v is merged with the VP it subcategorizes for, while the extracted expression what is simply taken along.

[3]This paper will stay ignorant with respect to phonological and semantic features. Just a short note on the effects of their distribution: If the phonological features are not merged in base position but move, it gives the phonological effect of overt movement, whereas in case the phonological features stay in base position, effects of covert movement can be obtained. Similarly, depending on with which expression the semantic value will be associated, the derivation would result in a surface interpretation or can show reconstruction effects.

(12) $\textbf{merge}\ v\ \langle \textsf{what}*, [_{\text{VP}}\ \textsf{punch}\ c_{\textsf{what}}]\rangle$
$= \langle \textsf{what}*, \textbf{merge}\ v\ [_{\text{VP}}\ \textsf{punch}\ c_{\textsf{what}}]\rangle$
$= \langle \textsf{what}*, [_{\text{vP}}\ \textsf{punch}\ c_{\textsf{what}}]\rangle$

But (M2) captures just one case of merging a complex expression, namely when a simple expression selects a complex one. Another case that can arise is that a complex expression selects another expression. This is, for example, the case when the resulting expression in (12) selects the external argument of the verb. Then what to do with the following?

$$\textbf{merge}\ \langle \textsf{what}*, [_{\text{vP}}\ \textsf{punch}\ c_{\textsf{what}}]\rangle\ \textsf{mary}$$

The idea is exactly the same like with (M2): the nucleus of the pair, the vP, should be merged with mary, and the copy of what should be percolated further. The definition (M3) implements that.

(M3) $\textbf{merge}\ \langle x, y\rangle\ z = \langle x, \textbf{merge}\ y\ z\rangle$

So applying (M3) to our case above we get:

(13) $\textbf{merge}\ \langle \textsf{what}*, [_{\text{vP}}\ \textsf{punch}\ c_{\textsf{what}}]\rangle\ \textsf{mary}$
$= \langle \textsf{what}*, \textbf{merge}\ [_{\text{vP}}\ \textsf{punch}\ c_{\textsf{what}}]\ \textsf{mary}\rangle$
$= \langle \textsf{what}*, [_{\text{vP}}\ \textsf{mary punch}\ c_{\textsf{what}}]\rangle$

The rule (M3) also capture the case when a complex expression selects a complex expression. The way (M2) and (M3) are formulated, the second one of two merged pairs is incorporated into the first one. That means, in case of $\textbf{merge}\ \langle x, y\rangle\ \langle z, w\rangle$, (M3) applies before (M2) and we get $\langle x, \textbf{merge}\ y\ \langle z, w\rangle\rangle$, which reduces to $\langle x, \langle z, \textbf{merge}\ y\ w\rangle\rangle$. I will shortly come back to the significance of this in section 5, when discussing the relevance of the ordering in a pair.

Now that we can handle the percolation of copies, we still need to specify what resolves the dependency at the *top*. What happens is the following: if a percolated expression finally can check its formal features with the nucleus of the pair it is part of, i. e. in a configuration $\langle x^f, y^{\wedge f}\rangle$ or $\langle x^{\wedge f}, y^f\rangle$, **merge** is triggered. This is implemented by the following operation **IM** (reminiscent for *internal merge*).

(IM) $\textbf{IM}\ \langle x, y\rangle = \textbf{merge}\ y\ x \quad \text{if } y \text{ licenses } x$

Now if the derivation outlined above in (12) and (13) reaches the point at which the C-head is merged, it will result in the expression:

$$\langle \textsf{what}*^{\wedge wh}, [_{\text{CP}}\ \textsf{mary punch}]^{wh}\rangle$$

This is a configuration where **IM** applies, so what can be merged with the CP whereupon the *wh*-feature is checked.

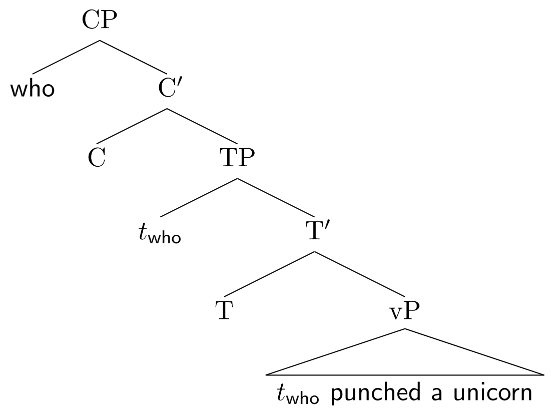
IM is assumed to apply as soon as possible, following the idea that operations in general have to be performed as soon as possible (e. g. stated in the form of the *Earliness Principle* by Pesetsky [18] and later adopted by Chomsky [7] in his condition *Maximize Matching Effects*). This means that movement cannot skip a potential landing site.

Before moving on, let us walk through one more example.

1.3 Example derivation

Consider the sentence Who punched a unicorn with the structure in (14), i. e. the subject wh-phrase originates in vP, first moves to SpecT to check its case feature, and then moves further on to SpecC in order to check its *wh*-feature.

(14)



Suppose we have already built the vP, which corresponds to the expression in (15).

(15) $\langle \text{who}*^{\wedge case,wh}, [_{\text{vP}}\ \text{punched a unicorn}]\rangle$

Next the T-head is merged.

(16) $\textbf{merge}\ T^{case}\ \langle \text{who}*^{\wedge case,wh}, [_{\text{vP}}\ \text{punched a unicorn}]\rangle$
$= \langle \text{who}*^{\wedge case,wh}, \textbf{merge}\ T^{case}\ [_{\text{vP}}\ \text{punched a unicorn}]\rangle$
$= \langle \text{who}*^{\wedge case,wh}, [_{\text{TP}}\ \text{punched a unicorn}]^{case}\rangle$

At this point of the derivation, **IM** can apply, which results in merging the two simple expressions. In doing so, the case feature can be checked.

(17) $\textbf{IM}\ \langle \text{who}*^{\wedge case,wh}, [_{\text{TP}}\ \text{punched a unicorn}]^{case}\rangle$
$= \textbf{merge}\ [_{\text{TP}}\ \text{punched a unicorn}]\ \text{who}*^{\wedge wh}$

Now the two expressions can either be merged into another simple expression by applying (M1), or who can be copied, applying (C1). Since it still has an unchecked wh-feature, let us assume it is copied in order to percolate further.

(18) **merge** $[_{\mathrm{TP}}$ punched a unicorn$]$ (**copy** who$*^{wh}$)
$=$ **merge** $[_{\mathrm{TP}}$ punched a unicorn$]$ $\langle$who$*^{wh}, c'_{\text{who}}\rangle$
$= \langle$who$*^{wh}$, **merge** $[_{TP}$ punched a unicorn$]$ $c'_{\text{who}}\rangle$
$= \langle$who$*^{wh}, [_{\mathrm{TP}}\ c'_{\text{who}}$ punched a unicorn$]\rangle$

Merging the C-head will then result in

(19) $\langle$who$*^{wh}, [_{\mathrm{CP}}$ punched a unicorn$]^{\wedge wh}\rangle$

Now, **IM** can apply again, this time in order to check the wh-feature. Then who does not have any more features to check, so both simple expressions can be merged with (M1) to another simple expression.

(20) **IM** $\langle$who$*^{wh}, [_{\mathrm{CP}}$ punched a unicorn$]^{\wedge wh}\rangle$
$=$ **merge** $[_{\mathrm{CP}}$ punched a unicorn$]^{\wedge wh}$ who$*^{wh}$
$= [_{\mathrm{CP}}$ who punched a unicorn$]$

Written as a feature bundle, the resulting expression is:

$$(\text{who punched a unicorn}, [C], [\,])$$

Now that we have seen how derivations proceed in Broziewski's model, I also want to make a few remarks on the relation of Brosziewski's model to other theories, such as Chomsky's Phase Theory [7].

1.4 Connection to other frameworks

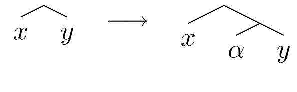
Although Brosziewski's theory had been developed independently of Phase Theory, they share the core idea that expressions assembled by syntax are sent to the interfaces as soon as possible, and that expressions which need to be available for syntactic operations later on have to escape that procedure. In Phase Theory escaping happens by means of movement to the phase edge (usually the specifier of a designated head), which is not sent to the interfaces immediately but is kept until the next phase is finished. In Brosziewski's account escaping happens by copying an expression and thereby keeping its relevant features accessible throughout the further derivation. Both implementations are very similar. In fact, the α_i in an expression $\langle\alpha_1, \langle\alpha_2 \ldots \langle\alpha_n, \beta\rangle\rangle\rangle$ of Brosziewski's model could be seen as being at the edge of a phase, while everything contained in β was sent to the interfaces. One difference to Phase Theory is that, although the extracting expressions α_i are percolated through every step of the derivation, this is not done by application of **merge**, thus it does not require the insertion of edge features that legitimate the application of that operation. Another difference is the way of reducing the part of the structure that is accessible for syntactic operations. Instead of restricting accessibility to a certain domain (like vP and CP, or every phrase XP), it is restricted to active elements, i. e. expressions that carry yet unchecked features – independently of how deep in the structure they were introduced.

I don't want to pursue a more detailed comparison with Phase Theory here, but what is interesting to note is that Brosziewski's model is not per se restricted to a certain view on movement. In principle, it can mimic different possibilities:

- *Movement in one fell swoop* can be implemented by stating that expressions at the edge of a pair are unaccessible for all operations except for percolation and remerging. They would then not be able to check features along the way, and could not show morphological reflexes or semantic effects.
- *Successive-cyclic movement with uniform paths* is what might seem to be the default assumption for Brosziewski's mechanism: extracted expressions are available during all steps in the derivation, and are thus always accessible for syntactic (as well as semantic and phonological) operations.
- *Successive-cyclic movement with punctuated paths* would be achieved by the somewhat less obvious assumption that expressions at the edge of a pair are accessible only at certain points in the derivation. This would mean that they could trigger morphological reflexes or show semantic effects in some intermediate but not all positions.

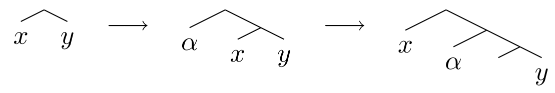
Which of these we prefer to adopt does not play a role for this paper. For a discussion of evidence that points towards punctuated movement paths cf. Abels & Bentzen (this volume).

I want to mention one possible conceptual objection that can arise from viewing complex expressions as trees, namely that (M2) and (M3) look like countercyclic operations because **merge** operates only on a proper substructure of the expression. First note that when thinking of complex expressions $\langle\alpha, \beta\rangle$ as a syntactic expression β together with a stack containing α, this does not have to be the case. Under this view it seems reasonable to assume that the strict cycle only refers to the expression β but not to the stack. This is reasonable since we don't want to require every operation to operate on the stack – especially because during the whole *middle* part of a dependency the stack should just be passed on without being modified. But when thinking of complex expressions in terms of a tree structure, α is indeed merged below the root:

(21) **merge** α 

The only way to avoid this and have a cyclic derivation is assuming that (M2) actually consists of the following two steps: first merging α with the whole expression and then moving x to the edge again.

(22) **merge** α



Arguably the counter-cyclic (21) and the cyclic (22) are not distinguishable in Brosziewski's terms. The results of both derivation courses correspond to the expression $\langle x, \textbf{merge}\ \alpha\ y\rangle$. To accommodate this, we could formulate a sloppy version of the cycle which – analogously to the stack point of view – refers to the nucleus of an expression only and leaves the edge of an expression out of consideration, i. e. which states that an operation may not apply to a proper subpart of the nucleus of an expression. In phase-theoretic terms this is like always tucking in expressions below the phase edge, but requiring that below the phase edge merge respects the strict cycle. In this sloppy sense, none of the operations throughout the paper violates the cycle. In fact it is impossible for any operation to do so, for the nucleus of an expression is always a simple expression without internal structure, so no operation can reach into it. (Which is not surprising at all because that was the idea behind expressions in the first place.)

Another way to think about (21) is to regard it as the adjoining operation of *Tree Adjoining Grammar* ([13],[14]): percolating an extracting element through the middle of a dependency without anything happening to it very much resembles adjoining an intermediate structure. The difference is that under Brosziewski's view the dislocated element is present while building the intermediate structure, thus can trigger morphological reflexes or semantic effects.

2 Remnant movement and Freezing

Starting from the basic mechanism outlined above, Brosziewski concentrates on extending it to also capture head movement (cf. [4, Section 3.3]). I want to depart from his considerations and instead take a closer look at possible and impossible movement configurations. In particular I want to show what remnant movement and Freezing configurations amount to in Brosziewski-style derivations, which problem arises with them, and how to solve it. This will lead to a slight modification of Brosziewski's mechanism, which – as far I can see – is compatible with his modifications for incorporating head movement.

For these explorations always keep in mind that complex expressions arise from copying and percolating an expression that is extracted, and that an XP that contains an element α that is extracted from XP is a pair of the form $\langle \alpha, XP\rangle$ (possibly with more copied subexpressions).

Now, which types of movement are possible up to now? So far, **copy** is only defined for simple expressions, thus movement is restricted to constituents without subextraction. For example the following extractions in (23) are unproblematic because only simple expressions move.

(23) a. $\beta \ldots \alpha \ldots [_{\text{XP}} \ldots t_\beta \ldots t_\alpha \ldots]$
b. $[_{\text{XP}} \ldots \alpha \ldots t_\alpha \ldots] \ldots t_{\text{XP}} \ldots$

However, configurations like in (24) cannot be derived. In both, an XP moves and an expression α is extracted from this XP. So at some point of the derivation, when the whole phrase has to be copied, **copy** would have to apply to the complex expression $\langle \alpha, XP \rangle$. Since up to now the operation **copy** is not defined for complex expressions, this is not possible, and hence it is not possible to move an XP with an extracting subexpression.

(24) a. $[_{\text{XP}} \ldots t_\alpha \ldots] \ldots [\ldots \alpha \ldots [\ldots t_{\text{XP}} \ldots]]$
b. $\alpha \ldots [[_{\text{XP}} \ldots t_\alpha \ldots] \ldots t_{\text{XP}} \ldots]$

What differs between (24a) and (24b) is the derivational order; (24a) corresponds to (A), and (24b) corresponds to (B).

(A) *Remnant movement*
First α moves out of XP, then (the rest of) XP moves.

(B) *Freezing configuration*
First XP moves, then α moves out of XP.

Extraction from a moved XP is generally taken to be prohibited in languages, whereas remnant movement is assumed to be possible. Consider the following examples from German. (25) instantiates remnant movement: the NP das Buch scrambles out of the VP, and then the remnant VP is topicalized. (26) on the other hand shows a Freezing effect: first the NP ein Buch worüber is scrambled, and subsequently the PP worüber is extracted from this NP via wh-movement.

(25)	$[_{\text{VP}}\ t_1$ Gelesen$]_2$	hat	$[_{\text{NP}}$ das Buch$]_1$	keiner	t_2
	read	has	the book.ACC	no-one.NOM	

(26)	* $[_{\text{PP}}$Worüber$]_2$	hat	$[_{\text{NP}}$ ein Buch $t_2\,]_1$	keiner	$[_{\text{VP}}\ t_1$ gelesen$]$?
	about what	has	a book.ACC	no-one.NOM	read

Let us first look at (26). At some point of the derivation, the NP is built, amounting to the complex expression

(27) $\langle [_{\text{PP}}$ worüber$]*, [_{\text{NP}}$ ein Buch $c_{\text{PP}}] \rangle$

If merged with the V-head, this NP would have to be copied in order to move further up.[4] But since it is a complex expression, **copy** cannot apply. So, according to the reasoning above, the Freezing configuration in (26) is indeed excluded. But for the same reason also every instance of remnant movement is impossible. Consider (25). When the VP is built, it amounts to the following expression.

(28) $\langle [_{\mathrm{NP}}\ \textsf{das Buch}]*, [_{\mathrm{VP}}\ c_{\mathrm{NP}}\ \textsf{gelesen}]\rangle$

Which is again a complex expression, that cannot be copied, although it would need to be copied when merged with v.

To remedy this, the definition of **copy** has to be extended to complex expressions. A first try could be to simply keep the list of extracting expressions and copy the nucleus of the pair (for we want to copy the constituent we built and not the list of moving elements):

$$\textbf{copy}\ \langle x, y\rangle = \langle x, \textbf{copy}\ y\rangle$$

This allows for remnant movement. In abstract terms because it is now possible to copy the complex expression $\langle \alpha, XP\rangle$ in the following way:

$$\textbf{copy}\ \langle \alpha, XP\rangle = \langle \alpha, \langle XP*, c_{XP}\rangle\rangle$$

Hence the remnant movement derivation for (25) could proceed with copying the VP in (28) and then merging it with v, yielding the following vP.

(29) $\langle [_{\mathrm{NP}}\ \textsf{das Buch}]*, \langle [_{\mathrm{VP}}\ c_{\mathrm{NP}}\ \textsf{gelesen}]*, [_{\mathrm{vP}}\ v\ c_{\mathrm{VP}}]\rangle\rangle$

The NP and VP at the edge are simple expression and thus can both be merged internally in the course of the derivation. But for the same reason, Freezing effects are no longer obtained. Copying the NP in (27) and merging it with V gives the following VP:

(30) $\langle [_{\mathrm{PP}}\ \textsf{worüber}]*, \langle [_{\mathrm{NP}}\ \textsf{ein Buch}\ c_{\mathrm{PP}}]*, [_{\mathrm{VP}}\ V\ c_{\mathrm{NP}}]\rangle\rangle$

Again, the two expressions at the edge are simple and can be merged internally in the course of the derivation. The situation is exactly the same like with remnant movement. The problem is that copying applies in base position and that there is no way to look ahead that could tell which element lands first, α or XP. Thus keeping both of them as seperate elements at the edge, i. e. building an expression $\langle \alpha, \langle XP, ...\rangle\rangle$, does not allow one to distinguish between Freezing configurations and remnant movement.

[4] Assuming that scrambling is triggered by some feature.

What to do about that? In fact there is a possibility to distinguish between the case where α lands before the rest of the XP (as in remnant movement) and the case where the complete XP lands first (as in Freezing configurations). The trick is to not percolate α and XP as seperate elements at the edge, but instead move the whole expression $\langle \alpha, XP \rangle$ as one constituent. So instead of building an expression $\langle \alpha, \langle XP, ... \rangle \rangle$, we will build an expression $\langle \langle \alpha, XP \rangle, ... \rangle$, with $\langle \alpha, XP \rangle$ at the edge. Now when in the course of the derivation α is merged internally first, what remains of this complex expression is just XP, a simple expression. So in this case, **IM** is applied to simple expressions only. If, however, the complete XP tried to land first, it would at that point of the derivation still be the complex expression $\langle \alpha, XP \rangle$. So in this case, **IM** has to be applied to a complex expression, for which it is not defined.

In a nutshell, the idea lies in exploiting the derivational difference between (A) and (B) by moving the expression $\langle \alpha, XP \rangle$ as one constituent. Then if α lands first and XP lands later on, both can be merged as simple expressions without a problem; however if the whole, complex expression tries to land first, it will fail to do so. Technically, the only thing we need to ensure is that α and XP are not kept as seperate, independent elements at the edge, but instead remain one constituent while moving. I. e. when copying the complex expression $\langle \alpha, XP \rangle$, the derivation should not produce $\langle \alpha, \langle XP, c \rangle \rangle$, but instead should produce $\langle \langle \alpha, XP \rangle, c \rangle$.

When having done so, copying the NP in the Freezing case and merging it with V would not give (30) but result in the following.

(31) $\langle \langle [_{\text{PP}}$ worüber$]*, [_{\text{NP}}$ ein Buch $c_{\text{PP}}]* \rangle, [_{\text{VP}}\ V\ c] \rangle$

At some point of the derivation, the whole NP would want to land, before the PP worüber reaches its landing site. Then internal merge would find the complex expression $\langle [_{\text{PP}}$ worüber$]*, [_{\text{NP}}$ ein Buch $c_{\text{PP}}]* \rangle$ at the edge, for which it is not defined. Thus the Freezing effect is obtained. The situation is different with remnant movement, although it starts out the same because copying the VP and merging it with v does not result in (28) but in the following.

(32) $\langle \langle [_{\text{NP}}$ das Buch$]*, [_{\text{VP}}\ c_{\text{NP}}$ gelesen$]* \rangle, [_{\text{vP}}\ v\ c] \rangle$

The difference is that the NP reaches its landing site first. Since it is a simple expression, it is allowed to be remerged. Assuming that it adjoins to vP, the result is:

(33) $\langle [_{\text{VP}}\ c_{\text{NP}}$ gelesen$]*, [_{\text{vP}}$ das Buch $v\ c] \rangle \rangle$

Now the extracted VP is a simple expression and nothing prevents it from being remerged later. Thus a remnant movement derivation is perfectly fine.

In order for this to work technically, we need to adapt the definitions involved. First of all, we need to make explicit which expressions are considered to be at the edge of a complex expression and thus can be reached by **IM**.

Definition 6. *An expression x is at the* edge *of a complex expression $\langle y, z\rangle$ if one of the following three conditions holds:*

(i) x is equal to y

(ii) x is at the edge of y

(iii) x is at the edge of z

Clauses (i) and (iii) capture the cases we already encountered. For example (i) reaches what in $\langle \mathsf{what}, z\rangle$ and (iii) reaches who in $\langle \mathsf{what}, \langle \mathsf{who}, z\rangle\rangle$. Up to now (ii) would have been equivalent to (i), because y was always simple. Now (ii) captures the new case of reaching what in $\langle\langle \mathsf{what}, x\rangle, z\rangle$.

Next we need to generalize **IM** slightly, in order to consider these new cases of edges as well, i. e. to reach not only the first element of the pair it is applied to but also the edge of this element. What will have to remain, most importantly, is that only simple expressions can be remerged with **IM**. Next, also the definition of **copy** needs to be extended with a clause for complex expressions that yields **copy** $\langle x, \alpha\rangle = \langle\langle x, \alpha*\rangle, c\rangle$. For exact definitions see the appendix.[5]

The remnant movement account outlined in this section is stronger than, for example, the one by Klaus Abels in [1]. This is because it prohibits every type of Freezing configuration, while Abels only prohibits Freezing configurations if the movement of the inclusive XP is 'less urgent' (higher in a certain hierarchy) than the extraction of an element α out of that XP. On the other hand, Abels permits Freezing configurations if the movement of the XP is 'more urgent' (lower in the hierarchy) than α's movement. He supports this claim with data that shows that A-movement can feed wh-movement and topicalization. An approach like Brosziewski's could possibly account for these cases if based on a treatment of A-movement as it is proposed by Kobele (this volume).

3 The Condition on Extraction Domain

A widely assumed constraint on movement is the Condition on Extraction Domain (CED), which originated in Huang [12]. In a simple form, it states that

[5]Pied Piping might seem like a Freezing configuration in this theory, because a whole phrase is moved although only a subexpression x (the wh-element itself) in it would be copied. But this need not be the case. One way to account for Pied Piping is to assume that x is not copied but has the last resort option to project its features. Then the more inclusive phrase will be copied, percolated into landing position and there check the projected feature. This is similar to the treatment of Pied Piping in GPSG with the help of feature percolation. Another possibility is to follow more recent theories like Cable [5]. Adopting Cable's idea, **copy** could not be triggered by wh-features but rather by some feature of a Q-morpheme higher up in the structure.

movement must not cross a barrier, where an XP is a barrier iff it is not a complement. Assuming that complements are exactly those expressions that check category features when they are merged, the CED can be stated as in (34).

(34) *Condition on Extraction Domain*
Extraction is prohibited from expressions which have been merged without checking category features.

In other words, extraction out of complements is fine (cf. (35)), while extraction out of adjuncts (cf. (37)) and specifiers (cf. (36)[6]) is out.

(35) What$_1$ did Mary tell [a story about t_1]?

(36) * What$_1$ has [a punching of t_1] annoyed you?

(37) * What$_1$ did you feel bad [because Mary punched t_1]?

To see how this constraint can be expressed in Brosziewski's terms, recall again that expressions with subextraction are encoded in complex expressions: an XP that contains an element α that is extracted from XP is a pair of the form $\langle \alpha, XP \rangle$. Those expressions are handled by (M2) and (M3), so this is where an account of the CED should be rooted. Note that (M3) can be ignored for that purpose, because application of it always boils down to either the case in (M1) or the one in (M2). So what is called for is a restriction on (M2).[7] Its definition from above is repeated here for convenience.

(M2) (*preliminary version*)
$\textbf{merge}\ \alpha\ \langle x, y \rangle = \langle x, \textbf{merge}\ \alpha\ y \rangle$

Up to now there is no restriction on (M2), so it allows **merge** with complex complements (i.e. complements out of which is extracted), with complex specifiers (i.e. specifiers out of which is extracted), and with complex adjuncts (i.e. adjuncts out of which is extracted). A straightforward way to disallow complex specifiers and adjuncts is to restrict (M2) to elements that are subcategorized for, according to the formulation of the CED.

(M2) (*still preliminary version*)
$\textbf{merge}\ \alpha\ \langle x, y \rangle = \langle x, \textbf{merge}\ \alpha\ y \rangle$ iff α subcategorizes for y

[6]This example assumes that subjects occupy a specifier position. I will discuss data on subjects and their treatment below.

[7]A restriction on **copy** would not make sense. First of all, **copy** does not see whether the expression that is copied is subcategorized for, licensed, or none of both (i.e. whether it is a complement, specifier, or adjunct). Second, a restriction on **copy** would only be able to restrict extraction out of specifiers and adjuncts that move themselves, whereas CED effects also apply to in situ specifiers and adjuncts. For the same reason restrictions on **IM** would not work.

However, (M2) as it is now is too strong, because it also excludes cases like the one we encountered in the second line of (18), here repeated as (38), where an expression that was internally merged is copied again in order to move further.

(38) **merge** $[_{\text{TP}}$ punches a unicorn$]$ $\langle \mathsf{who}{*}^{wh}, c'_{\mathsf{who}} \rangle$

In this case, the TP does not subcategorize for the second element of the pair. Thus **merge** cannot operate. The problem arises from the fact that there is no difference between pairs which result from copying, and pairs, where the first element is a proper subexpression of the nucleus, i. e. between a whole phrase moving (as in (38)) and expressions extracting from a bigger constituent. So Broziewski needs to add another clause to (M2) in order to exempt pairs of copies from the subcategorization restriction. The final version of (M2) then is the following (cf. [4, (73b) on p. 56]).

(M2) (*final version*)
merge α $\langle x, y \rangle = \langle x,$ **merge** $\alpha\ y \rangle$ iff

(i) α subcategorizes for y, or
(ii) $\langle x, y \rangle$ is a pair of copies

After having seen why adjuncts and specifiers are islands and complements are not, let us consider subject islands. The situation there is less clear-cut. In many languages, subjects are islands (as in English), but in some languages extraction from subjects is possible (for example in Japanese, Hungarian, Turkish, Palauan, see e. g. Stepanov [22]).

In Brosziewski's model there are basically two options to treat subjects. The first one is to assume that subjects are specifiers. In that case they are predicted to be islands. Languages that allow extraction from them have to lack the restrictions on (M2). (Specifiers created by movement will still be islands because of a Freezing effect.) The other option is to treat them as complements (of either V or v). This is straightforward, since neither multiple specifiers nor multiple complements are excluded. The prediction then is that subjects are transparent for subextraction. Their islandhood can be derived with the help of Freezing, so that subjects are barriers only if they moved from their base position. This is unproblematic in languages in which the subject obligatorily moves to SpecT, as it does in English. However, there are languages in which the subject can stay in situ. Contrary what is predicted, they might exhibit subject island effects. This can be seen in the following German examples. (That the subject stays in situ is based on the assumption that particles like denn and wohl demarcate the vP edge.)

(39) * Was$_1$	haben	denn	$[_{\text{DP}}$ t_1 für Bücher$]$	$[_{\text{DP}}$ den Fritz$]$	beeindruckt?
what	have	PRT	for books.NOM	the Fritz.ACC	impressed

(40) * [$_{\text{PP}}$ Über wen]$_1$ hat wohl [$_{\text{DP}}$ ein Buch t_1] [$_{\text{DP}}$ den Fritz] beeindruckt?
about whom has PRT a book.NOM the Fritz.ACC impressed

A very interesting fact is that these subjects can become transparent if the direct object scrambles across them, targeting a higher specifier position of vP.[8]

(41) Was$_1$ haben [$_{\text{DP}}$ den Fritz]$_2$ [$_{\text{DP}}$ t_1 für Bücher] t_2 beeindruckt?
what have the Fritz.ACC for books.NOM impressed

(42) [$_{\text{PP}}$ Über wen]$_1$ hat [$_{\text{DP}}$ den Fritz]$_2$ [$_{\text{DP}}$ ein Buch t_1] t_2 beeindruckt?
about whom has the Fritz.ACC a book.NOM impressed

I have to leave open what could account for this behavior of subjects (and indirect objects) here. For different treatments of subjects in the framework of HPSG – either on the argument list of a verb or not – see e. g. Pollard & Sag [19] and Müller [17].

4 Weak islands

Weak islands like wh- and topic-islands do not follow immediately from the derivational mechanism employed so far. In order to account for them, Brosziewski [4, p. 64f] hints at how the concept of subcategorization can be widened to involve both syntactic and semantic information, such that it can be sensitive to contrasts like ±finite, ±propositional, and so on. Islands then amount to expressions that are not subcategorized for. But obviously the success of such an approach depends on an appropriate theory of selection.

I want to propose an alternative route, that makes use only of the formal means we already have, plus a simple additional constraint: the principle of *Unambiguous Domination* from Müller [15]. Müller introduced it to capture a core restriction on remnant movement, namely that remnant XPs cannot undergo a certain type of movement if the subextracted expression that created the remnant has undergone the same type of movement. The derivational version of the principle reads like the following.

(43) *Unambiguous Domination* (derivational version)
In a structure $\ldots [_A \ldots B \ldots] \ldots$, A and B may not undergo the same kind of movement.

This condition can straightforwardly be expressed in Brosziewski's terms as a filter over expressions that states that an element with some feature f may not be percolated across a constituent that bears the same feature f.

[8]These *melting* effects are due to Müller [16]. They can also be observed with indirect objects and also occur with scrambling in Czech.

(UD) $^* \langle x^f, y^f \rangle$

Besides the generalization about remnant movement that motivated (UD), it can cover weak islands such as wh- and topic islands, examples of which are given in (44a) and (44b), respectively. If we assume that *wh*- and *top*-features on a CP are not deleted when checked but remain visible, the embedded CP corresponds to expressions (45a) and (45b), respectively – expressions that are excluded by (UD).

(44) a. * What$_1$ does John think $[_{\text{CP}}$ whether Mary punches $t_1]$?

b. * $[_{\text{PP}}$ With her bare hands$]_2$ John thinks that $[_{\text{CP}}$ [a unicorn]$_1$ Mary punched t_1 $t_2]]$

(45) a. $\langle$what$*^{wh}$, $[_{\text{CP}}$ whether Mary punches $c_{\text{what}}]^{wh}\rangle$

b. $\langle[_{\text{PP}}$ with her bare hands$]*^{top}$, $[_{\text{CP}}$ a unicorn Mary punches $c_{\text{PP}}]^{top}\rangle$

A nice consequence is the following. With Müller's version of *Unambiguous Domination*, it is necessary to define a local domain, in which the condition applies, because otherwise sentences like (46) would wrongly be predicted to be out, for the wh-element in the embedded CP and the more inclusive NP undergo the same kind of movement.

(46)

$[_{\text{NP}}$ Wessen	Frage	$[_{\text{CP}}$ was$_1$	du	magst $t_1]]_2$	hat	t_2	dich	geärgert?
whose	question	what	you	like	has		you	annoyed

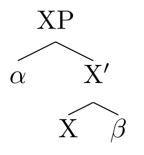
The crucial insight here is that 43 only plays a role in cases where B moves out of A, but not if B moves within A. The representational version that Müller uses (stating that an α-trace may not be α-dominated) does not capture that. However, in a derivational setting, this problem does not arise, because the movement of B will already be finished when A is moved. Thus these two movements should not interfere. And indeed, in a Brosziewski-style derivation of 46, the embedded CP is a simple expression, because was$_1$ already reached its landing site, and even if the *wh*-feature is still visible on the CP, it will not project when the NP is built. The NP will be a simple expression with a $^\wedge wh$-feature itself, it will be copied and percolated until it reaches its own landing site. At no point of the derivation does a configuration like (UD) arise.

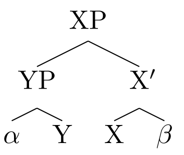
5 The Minimal Link Condition

Besides rigid constraints on movement like the CED and island constraints, there is another kind of movement constraints, that is based on the competition of expressions that carry the same relevant feature. A widely assumed instance of these minimality constraints is the Minimal Link Condition (MLC) [6]. In general terms it requires that of two expressions with the same feature the one which is closest to the target moves. Specifying closeness in terms of c-command or dominace (cf., e. g., Fitzpatrick [8]) it can be stated as in the following definition.

(47) *Minimal Link Condition*
In a structure $\alpha^{\wedge f} \ldots [\ldots \beta^f \ldots \gamma^f \ldots]$, movement to $^\wedge f$ can only affect the expression bearing the f-feature that is closer to $^\wedge f$, where β is closer to γ if β dominates or c-commands γ.

First it should be noted that, since derivations do not build phrase structures, dominance and c-command can only play a role to the extent to which they are retrievable from the expressions built. So let us start with examining in which respect the structure in a pair constitutes a representational residue. Clearly, dominance is encoded in pairs: the nucleus α of a pair $\langle x_1, \langle x_2 \ldots \langle x_n, \alpha\rangle\rangle\rangle$ dominates every extracting expression $x_1, x_2, \ldots, x_n$ (or is equal to it, as would be the case with a pair of copies). This is always the case, because the x_i are extracted from α. With c-command, the situation is different. Although it is the case that a structure in which x c-commands y corresponds to a pair where x is nested deeper in the pair than y, this correspondence does not hold in the other direction. Consider the following structures, where α and β are assumed to have formal features that need to be checked by movement:





Although in one α c-commands β whereas in the other one it does not, the derivation of both will build a pair of the form

$$\langle \beta*, \langle \alpha*, [_{XP} \ldots c_\alpha \ldots c_\beta]\rangle\rangle$$

Closer scutiny reveals that what nestedness in a pair amounts to is the point at which the expressions entered the derivation. In a pair $\langle x_1, \langle x_2 \ldots \langle x_n, \alpha\rangle\rangle\rangle$, x_i was merged before x_{i+1}, i. e. the most recently merged expression will end up deepest in the pair (or: closest to the nucleus). This is due to how (M2) and (M3) distribute expressions over pairs. A consequence of this is that intervention effects are bound to work in terms of derivational order. It is therefore predicted that intervention without c-command should be possible – which indeed is the case, as observed by Heck & Müller [11] and as evidenced e. g. by the following example.

(48) $^{?*}$ Who$_2$ did [$_{DP}$ the woman that punched what] imitate t_2?

However, it is unclear how exactly to account for these data here. The main problem is that the intervening wh-expression is in situ. If in situ expressions are considered not be copied, they should not be able to intervene, unless their features somehow remain visible. The same problem arises if one wants to account for superiority effects in languages where only one wh-expression moves overtly, as in the following standard examples from English.

(49) Who$_1$ t_1 punches what?

(50) * What$_2$ does who punch t_2?

For the moment, let us assume that in a structure like (51a), all wh-expressions are copied and there is a mechanism to delete the copies of those expressions that do not move overtly. (In fact, it is possible to formulate such a mechanism without much ado, see the appendix for details. However, a lot more would be needed to say for the different cases of multiple wh-movement across languages.) Then the complex expression corresponding to (51a) is (51b).

(51) a. $[_{CP}\ C^{\wedge wh}\ [_{TP}$ who^{wh} punches what$^{wh}]]$
b. $\langle$what$*^{wh}$, $\langle$who$*^{wh}$, $[_{CP}\ c_{\mathsf{who}}$ punches $c_{\mathsf{what}}]^{\wedge wh}\rangle$

The next step of the derivation would be to apply **IM** in order to check C's $^{\wedge}wh$-feature. But, although **IM** applies to pairs, nothing was said yet about to which pair it applies if they are nested. **IM** can apply to the inner pair as well as to the outer pair, either checking C's feature with who or with what. In order to derive superiority effects, **IM** has to be required to always proceed inside out. This strategy, however, does not need to be hardwired into the operation **IM** itself but can simply be a fact about how a derivation proceeds.

The advantage of the approach sketched here over the MLC is that superiority is not expected to be a universal effect. Analogously to superiority effects, anti-superiority effects, where the farthest expression moves, can be achieved, simply by chosing the opposite strategy, i. e. by letting **IM** proceed outside in. Moreover, these two strategies cannot only vary across languages, but also across configurations within one language, e. g. in the line of Richards [20], who proposes that anti-superiority is a property of multiple attraction to a single position, whereas superiority is a property of attraction to multiple positions.

Finally, note that (UD)-configurations in (52a), where an element at the edge carries the same feature like the nucleus, look very similar to Minimality configurations in (52b), where two expressions at the edged carry the same feature.

(52) a. * $\langle x^f, y^f\rangle$
b. $\langle x^f, \langle y^f, z\rangle\rangle$

Indeed, (52a) can be seen as a Minimality effect under certain assumptions about **IM**. This becomes clear only when considering how a derivation that has built the expression $\langle x^f, y^f \rangle$ proceeds. The next step involving the pair would be merging it with another expression z. This can have two forms: either the pair $\langle x^f, y^f \rangle$ selects z, or z selects the pair. The first case will again yield a pair with a very similar form:

(53) $\textbf{merge}\ \langle x^f, y^f \rangle\ z$
$= \langle x^f, \textbf{merge}\ y^f\ z \rangle$
$= \langle x^f, [y\ z]^f \rangle$

In the other case, i. e. when z selects $\langle x^f, y^f \rangle$, the pair needs to be copied. Otherwise y would not be able to check its feature f during the further derivation, because it is z that projects its features. The result is the following:

(54) $\textbf{merge}\ z\ \langle x^f, y^f \rangle$
$= \textbf{merge}\ z\ (\textbf{copy}\ \langle x^f, y^f \rangle)$
$= \textbf{merge}\ z\ \langle \langle x^f, y^f \rangle, c \rangle$
$= \langle \langle x^f, y^f \rangle, z \rangle$

At some point in the derivation there will be an expression with a corresponding feature $^\wedge f$, yielding:

(55) $\langle \langle x^f, y^f \rangle, [\ldots]^{^\wedge f} \rangle$

This configuration triggers **IM**. Now if **IM** is required to target the 'highest' element at the edge, i. e. the pair $\langle x^f, y^f \rangle$ instead of x^f, then the derivation will crash because **IM** is not defined for remerging complex expressions. That is, under the assumption that **IM** has to pick the expression that is structurally highest (the one that was merged latest), the prohibition in (52a) can be derived.

6 Across-the-board extraction

Since extraction in Brosziewski-style derivations combines the idea behind dislocation in GPSG with a Minimalist setting, let us finally take a look at a constraint that is very easy to express in GPSG, but turned out to pose serious problems for Minimalist movement approaches: the Coordinate Structure Constraint (cf. Ross [21]).

(56) *Coordinate Structure Constraint*
In a coordinate structure, no conjunct may be moved, nor may any element contained in a conjunct be moved out of that conjunct, unless movement simultaneously affects both conjuncts.

The following examples illustrate this constraint.

(57) * I wonder what$_1$ Mary [[punched] and [t_1]].

(58) * I wonder what$_1$ [[John hugged t_1] and [Mary punched a unicorn]].

(59) I wonder what$_1$ [[John hugged t_1] and [Mary punched t_1]].

The gist of the GPSG account (see especially Gazdar [9]) is that only expressions of the same category can be coordinated. And since gaps are encoded in the category of an expression, this automatically accounts for across-the-board extraction.

Now let us first look at how the generalization that only expressions of the same category can be coordinated can be achieved with Brosziewski's model. This is actually fairly easy; all we have to assume is that a coordinating expression like and subcategorizes for the same category twice. The lexical entry for and that does that is given in (60), where X is a polymorphic feature that can be instantiated by any catgeorial feature.

(60) $\mathit{and} = (\mathsf{and}, [{}^\wedge X, {}^\wedge X, X], [\,])$

But what this lexical item cannot account for yet is the fact that across-the-board extraction is possible, because – unlike in GPSG – extraction is not encoded in the syntactic category. In fact it cannot be, because even when an expression is moved, a copy is merged in base position in order to fulfil subcategorization requirements. Thus from the point of view of syntax, the constituent is complete. In other words, there are no gaps in Brosziewski's model. However, the derivational mechanims does keep track of extracted expressions. Thus across-the-board extraction can be achieved by defining a recursive coordination function like the following.

(CO)
$$\begin{aligned} &\textbf{coord}\ \ \alpha \quad\ \ \beta \quad\ = \textbf{merge}\ (\textbf{merge}\ \mathit{and}\ \beta)\ \alpha \\ &\textbf{coord}\ \langle x, y\rangle\ \langle x, z\rangle = \langle x, \textbf{coord}\ y\ z\rangle \end{aligned}$$

This function is only defined for expressions that have exactly the same edge: either none at all, or equivalent copies of extracted expressions. As an example for how this works, consider (61) with parts of the derivation in (62).

(61) I wonder what$_1$ [[John hugged t_1] and [Mary punched t_1]].

(62)
$$\begin{aligned} &\textbf{coord}\ \langle \mathsf{what}*, [\mathsf{Mary\ punched}\ c_{\mathsf{what}}]\rangle\ \langle \mathsf{what}*, [\mathsf{John\ hugged}\ c_{\mathsf{what}}]\rangle \\ &= \langle \mathsf{what}*, \textbf{coord}\ [\mathsf{Mary\ punched}\ c_{\mathsf{what}}]\ [\mathsf{John\ hugged}\ c_{\mathsf{what}}]\rangle \\ &= \langle \mathsf{what}*, \textbf{merge}\ (\textbf{merge}\ \mathit{and}\ [\mathsf{Mary\ punched}\ c_{\mathsf{what}}])\ [\mathsf{John\ hugged}\ c_{\mathsf{what}}]\rangle \end{aligned}$$

Although an extra function for coordination is needed, it reduces to **merge**, such that no additional means for deleting the edge of one conjunct have to be presupposed. Moreover, since **merge** is a binary function and the first clause of the definition (CO) assumes it to first combine *and* with the second conjunct and only after that with the first one, hierarchy effects for the two conjuncts are expected, as they are found, for example, with binding as in (63).

(63) a. Mary punched [[every unicorn]$_i$ and [its$_i$ owner]].
b. * Mary punched [[its$_i$ owner] and [every unicorn]$_i$].

In the end, one minor flaw is that the definition (CO) still allows for the case of whole conjuncts extracting, as long as they are the same, as e.g. in the ungrammatical What$_1$ did Mary punch [t_1 and t_1]. To exclude this, it would have to be assumed that α and β in the first clause of (CO) cannot be phonologically empty.

7 Conclusion

This paper started by presenting Ulf Brosziewski's model of syntactic derivations as an implementation of a strongly derivational organization of grammar. I then focused on showing how different kinds of constraints on movement can be captured and that actually very little is necessary to cover the core facts. The employed means were of two sorts. The first one were filters over expressions or over the input of functions. One such filter constituted in restricting the operation **merge** such that complex expressions can only be merged if they are subcategorized for. That allowed to account for a range of CED effects. Another filter was Müller's principle of *Unambiguous Domination*, that was expressed as disallowing expressions of the form $\langle x^f, y^f \rangle$. Besides the general restriction on remnant movement that it was designed for, it captured weak island effects like induced by wh- and topicalization islands. Another kind of constraint that seemed to naturally emerge in a derivational approach were conditions on the flow of the derivation, i. e. the order in which operations apply. It was indicated how the order of application of the function **IM** in nested pairs allows to derive minimality effects or the lack of them in a quite flexible way. However, these effects also highlighted a problem, namely how expressions that stay in situ are visible, e. g. in order to be able to intervene. In the appendix I provide a provisional solution for languages in which exactly one wh-expression moves, but clearly more needs to be said about multiple wh-movement across languages and about intervention effects in general.

Another open issue is to derive the considered movement constraints from more general properties of the computational system, requirements of the interfaces, or the feature make-up of lexical expressions. This mainly calls for a refinement of the feature system, with respect to syntactic features as well as regarding phonological and semantic features, whose influence on derivations was not considered yet at all.

At the beginning of section 5 I talked about the representational residue of

Brosziewski's derivational model. So let me finally indicate where it would be classified with respect to Brody's hierarchy of derivational approaches in [3]. Brody calls a derivational theory *weakly representational* if the objects that derivations generate are transparent in the sense that the material they are assembled of is accessible for later operations. With respect to this characterization, Brosziewski's expressions could suitably be called half-transparent. First because only some of the assembled material is still accessible later on, and second because it is so only as long as necessary, but not throughout the whole derivation. So Brosziewski's model would reside somewhere inbetween nonrepresentational and weakly representational theories. Even more, Brosziewski's theory can be understood as a case in point for attempts to reduce the amount of representations in syntax to a minimum.[9]

Appendix

Three things in the main text were left open. For section 2, the extension of **IM** still has to be given. This is captured by the following definition.

(64) $\mathbf{IM}\ \langle x, y\rangle = \mathbf{merge}\ \langle x', y\rangle\ \alpha$ for some α that

(i) is at the edge of x, and

(ii) is licensed by the nucleus of y

where x' is x without α.

The nucleus of an expression can be explicitly defined by the following recursive definition:

$$\begin{aligned} \mathbf{nuc}\ \alpha &= \alpha \\ \mathbf{nuc}\ \langle x, y\rangle &= \mathbf{nuc}\ y \end{aligned}$$

From the same section also the exact, recursive definition of **copy** was postponed. It is given by:

(C2) $\mathbf{copy}\ \langle x, y\rangle = \langle\langle x, y\,[\hat{y} := \hat{y}_1]\rangle, \hat{y}_2\rangle$

where $\hat{y} = \mathbf{nuc}\ y$
$\hat{y}_1 = \pi_1\ (\mathbf{copy}\ \hat{y})$
$\hat{y}_2 = \pi_2\ (\mathbf{copy}\ \hat{y})$

and where $y[x := z]$ is like the expression y except that x is replaced by z, and where π_1 and π_2 grab the first and second element of a pair, respectively

[9] A drawback for this point of view, however, is my use of (UD) to account for weak islands in Section 4, because (UD) is a constraint over the expressions generated – something that Brody would classify as strongly representational. However, instead of prohibiting expressions $\langle x^f, x^f\rangle$, one could also specify $\mathbf{merge}\ x^f\ y^f$ as undefined.

In Section 5, I claimed that a mechanism to delete the copies of those expressions that do not move overtly can be stated rather easily, in order to capture languages in which exactly one wh-expression moves. What is needed are three assumptions:

(i) The operation **copy** is not optional but obligatory (triggered by unchecked formal features).

(ii) After a feature f was checked, all other instances of f in the expression are deleted.

(iii) Occurences of empty expressions $(\epsilon, [\,], [\,])$ can be deleted since they will never play any role in the derivation, i. e. $\langle(\epsilon, [\,], [\,]), y\rangle = y$.

Consider the example (49) from section 5, here repeated as (**??**). When the derivation has built the expression in (66a), it proceeds as follows. As assumed in that section, **IM** will apply inside out, so C's feature $^\wedge wh$ will be checked with the wh-feature of who and we get (66b). Then, according to the second assumption above, all other instances of the wh-feature are deleted, yielding (66c).

(65) Who$_1$ t_1 punches what?

(66) a. $\langle \mathsf{what}{*}^{wh}, \langle \mathsf{who}{*}^{wh}, [_{CP}\ c_{\mathsf{who}}\ \mathsf{punches}\ c_{\mathsf{what}}]^{\wedge wh}\rangle$

b. $\langle \mathsf{what}{*}^{wh}, [_{CP}\ \mathsf{who}\ c_{\mathsf{who}}\ \mathsf{punches}\ c_{\mathsf{what}}]\rangle$

c. $\langle \mathsf{what}{*}, [_{CP}\ \mathsf{who}{*}\ c_{\mathsf{who}}\ \mathsf{punches}\ c_{\mathsf{what}}]\rangle$

Now two cases are possible. First, what∗ can have other unchecked features. In that case the derivation proceeds as usual. Extraction of what is then motivated by some other feature checking and simply does not constitute wh-movement. The second case is that what∗ does not have any other unchecked features. Then this expression can be of two forms: either it is the expression $(\mathsf{what}, [\,], [\,])$ or the expression $(\epsilon, [\,], [\,])$, depending on which copy actually carries the phonological representation. In the first case, the expression is not empty and thus cannot be deleted. Neither can it be remerged with **IM** at any later point (this is because **IM** is always triggered by formal features which the expression does not have). Thus the derivation will never reach a simple expression and thus cannot succeed. In the second case, the expression is empty and can be deleted according to the third assumption. Then the desired effect is achieved: the wh-feature is deleted, so the derivation can proceed. Moreover the phonological content is on the copy that was merged in situ, thus the resulting expression will appear as if what was not moved at all. What this would come closest to in a classical movement analysis is feature movement.

References

[1] Klaus Abels. Towards a restrictive theory of (remnant) movement. *Linguistic Variation Yearbook*, 7(1):53–120, 2007.

[2] David Adger. A minimalist theory of feature structure. Ms., Queen Mary University of London, 2008.

[3] Michael Brody. On the status of representations and derivations. In D. Epstein S. and D. Seely T. editors, *Derivation and Explanation in the Minimalist Program*, pages 19–41. Blackwell, 2002.

[4] Ulf Brosziewski. *Syntactic Derivations. A Nontransformational View.* Linguistische Arbeiten 470. Niemeyer, 2003. (PhD thesis from 2000).

[5] Seth Cable. *The Grammar of Q: Q-particles and the nature of wh-fronting, as revealed by the wh-questions of Tlingit.* PhD thesis, MIT, Cambridge, Mass., 2007.

[6] Noam Chomsky. *The Minimalist Program.* MIT Press, 1995.

[7] Noam Chomsky. Derivation by phase. In M. Kenstowicz, editor, *Ken Hale: A Life in Language*, pages 1–54. MIT Press, 2000.

[8] Justin Fitzpatrick. On minimalist approaches to the locality of movement. *Linguistic Inquiry*, 33:443–463, 2002.

[9] Gerald Gazdar. Unbounded dependencies and coordinate structure. *Linguistic Inquiry*, 12(2):155–184, 1981.

[10] Gerald Gazdar, Ewan Klein, Geoffrey K. Pullum, and Ivan A. Sag. *Generalized Phrase Structure Grammar.* Blackwell; Harvard University Press, 1985.

[11] Fabian Heck and Gereon Müller. Successive cyclicity, long-distance superiority, and local optimization. In Roger Billerey and Brook D. Lillehaugen, editors, *Proceedings of WCCFL 19*, pages 218–231. Somerville, MA: Cascadilla Press, 2000.

[12] Cheng-Teh James Huang. *Logical relations in Chinese and the theory of grammar.* PhD thesis, MIT, Cambridge, Mass., 1982.

[13] Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. Tree adjunct grammars. *Journal of Computer and System Science*, 10:136–163, 1975.

[14] Anthony Kroch and Aravind K. Joshi. The linguistic relevance of tree adjoining grammar. Technical report, University of Pennsylvania Department of Computer and Information Sciences Technical Report MS-CIS-85-16, 1985.

[15] Gereon Müller. *Incomplete Category Fronting.* Kluwer, 1998.

[16] Gereon Müller. On deriving CED effects from the PIC. Ms., Universität Leipzig, 2008.

[17] Stefan Müller. *Head-Driven Phrase Structure Grammar: Eine Einführung*. Stauffenburg, Tübingen, 2007.

[18] David Pesetsky. Language-particular processes and the earliness principle. Ms., MIT, Cambridge, Mass., 1989.

[19] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, 2004.

[20] Norvin Richards. *Movement in language*. Oxford University Press, 2001.

[21] John Robert Ross. *Constraints on variables in syntax*. PhD thesis, MIT, Cambridge, Mass., 1967.

[22] Arthur Stepanov. The end of CED? Minimalism and extraction domains. *Syntax*, 10(1):80–126(47), 2007.

[23] Juan Uriagereka. Multiple spell-out. In S. Epstein and N. Hornstein, editors, *Working Minimalism*, pages 251–282. MIT Press, 1999.