

Deriving Labels

JOHANNES JURKA

Abstract

This paper builds on Hornstein's (2005) proposal that Merge (Chomsky, 1995) can be split up into the more basic operations Concatenate and Label. This opens up the possibility that Concatenate applies without Label to generate flat structures, an option which Hornstein & Nunes (2005) explore for encoding adjuncts. The central claim is that Label is not needed as a syntactic primitive if a long distance dependency formation operation is added, which is shown to be necessary on independent grounds. Concretely, the only means of communication between lexical items is the *Unambiguous Paths Strategy* (UPS), defined in terms of Unambiguous Paths configurations (Kayne, 1981). Labelling is no axiomatic operation of the system but is triggered by the need to check uninterpretable features. The taxonomy between arguments and adjuncts is thus reduced to the presence or absence of uninterpretable features. Binary branching for arguments follows, while no such requirement holds for adjuncts.

Introduction

The point of departure of this paper are Hornstein's (2005) ideas of splitting up Chomsky's (1995) Merge into the more basic operations Concatenate and Label. The main theoretical goal of this contribution will be to motivate the application of labelling from inherent properties of the atoms phrase structure operates upon. These atoms are taken to be an idealized form of lexical items, and their properties are coded as morphosyntactic features (cf. Chomsky, 1995). The sole role of syntax in this system is to provide an environment for lexical items to "communicate" with each other in order to satisfy their requirements, i.e. to have their uninterpretable features (uF)¹ checked. Such an environment can be defined in terms of Kayne's (1981) "unambiguous paths". An immediate consequence of this approach will be that we no longer have to assume binary hierarchical branching as axiomatic. We will claim that hierarchy is not dictated by the operations of syntax, but by inherent properties of the atoms it combines.

A clear prediction of such an approach is that, since the elementary operations allow for flat structures, these options should be utilized by language. Following ideas by Chametzky (1996) and Uriagereka (to appear) flat structures are instantiated in (at least) two areas of syntax: iterative structures (*This is a very, very good book.*) and adjuncts. This article focuses on the latter. Adjuncts have been the poor cousins of phrase structure theory, as they are often glossed over in many approaches. Our system not only allows

¹ This abbreviation is borrowed from Pesestky&Torrego (2001), the idea to point this out from Citko (2005).

for a straightforward incorporation of adjuncts but actually argues that adjuncts are more well-behaved than arguments. Metaphorically speaking, adjuncts are the hippies of phrase structure in that they can be self-sustainable. They provide for themselves in that they are able to satisfy their featural requirements internal to themselves without depending on other elements of the structure. This dovetails with their observational characteristics of being optional and generally less restricted in where they can appear. In addition, our system codes earlier proposals that adjuncts may remain unlabeled (Chametzky, 1996). We will argue that since all their features are satisfied, labelling is not triggered and they manifest themselves in a flat structure. Interestingly, they can give up their independence and enter relationships with other elements, for example, for the purpose of encoding information structure. Some uninterpretable discourse feature within the adjunct remains unchecked, and labelling is automatically triggered to provide a proper configuration for this feature to be satisfied by some element external to the adjunct. This leaves us with the desirable result of viewing the difference between arguments and adjuncts as only taxonomic, the only distinction being the presence of an μF which cannot be checked internally.

1 Hornstein (2005): Merge = Concatenate + Label

Let us start out from two well established observations about the nature of the language faculty.

- (1) **First Property**
Language generates a potentially infinite amount of sentences from a finite set of atomic elements.
- (2) **Second Property**
Sentences are hierarchically structured.

These properties have been explicitly assumed ever since the early days of generative theory, starting with Chomsky (1955). The advent of Minimalism has slightly changed the view on these properties. While they had to be (and were) integrated in earlier models, be it in the form of Phrase Structure Rules or X-bar theory, minimalist reasoning views language as an *optimal system* that can be taken to yield nothing more than these properties. Chomsky proposes the operation Merge, which he takes to be “indispensable in some form in any language-like system” (Chomsky, 2000:113) and which was designed to account for both (1) and (2) in a single step.

- (3) **Merge**
Merge “takes a pair of syntactic objects (SO_i , SO_j) and replaces them by a new combined syntactic object SO_{ij} (Chomsky 1995a: 226)

Let us briefly examine the nature of Merge and the implicit assumptions made about it. According to Chomsky (2005) it is the Non-Tampering Condition (NTC) which prevents Merge from applying within syntactic objects, i.e. the application to subatomic elements is disallowed. Informally, the NTC states that an operation which combines two elements should not alter these elements in any way. This is because there is no conceptual reason to allow Merge to change the atomic elements it takes as its input. Assuming that it did would constitute a major departure from the Strong Minimalist Hypothesis (STM) (Chomsky, 2000) and could only be motivated by compelling empirical evidence. Chomsky does at no point provide a formal definition of the NTC, a move which gives him room to manoeuvre within the narrow notion of Merge. At closer inspection we find that his NTC goes one step farther in implying that “Merge” both applies to syntactic atoms and to complex syntactic objects. Strictly speaking, then, the operation Merge splits into two distinct operations, call them Atomic Merge (AM) and Complex Merge (CM). For both AM and CM the NTC holds, i.e. the elements that are input to either AM or CM must not be altered. AM applies to genuinely atomic elements while CM applies to complex elements, which it treats as atomic. It is significant that these are clearly separate operations, which take different elements as their input and although there might be compelling empirical reasons why this approach is on the right track, it is difficult to see how this definition of Merge is nothing more than a consequence of “virtual conceptual necessity”, as it is claimed by Chomsky.

Furthermore, it is not entirely clear why Merge should take exactly two SOs to form a new SO. *Two* seems to be an arbitrary number in as far as Merge is a set-formation operation which could be defined over any number of objects including *one*. Conceivably, Merge could take the atom A as its input and yield the SO {A}. I can see no prima facie objection against that. The atom A is arguably distinct from the set {A}, so a new object would have been yielded. Of course, such an operation would not drive the derivation in any way, so our descriptive generalization in (1) and (2) could not be the result of such an operation. Additionally, the restriction that Merge only applies to pairs of elements seems challengeable as well. In fact, some formulations in the MP itself suggest that Merge might possibly apply to more than two elements: “Applied to two objects α and β , Merge forms the new object K, eliminating α and β ” (Chomsky, 1995: 243). Does this exclude the possibility that, applied to three objects α , β and γ , Merge forms the new object K, eliminating α , β and γ ? This is not immediately obvious.

Let us turn to the alternative operation Concatenate, proposed by Hornstein (2005).

- (4) **Concatenate** (Hornstein, 2005:2)
 $A, B \rightarrow A^{\wedge}B$
 $C, A^{\wedge}B \rightarrow C^{\wedge}A^{\wedge}B$

Concatenate is conceptually simpler than Merge. In addition, note that even though Merge seems generally more powerful than Concatenate, the latter by definition has to take at least two objects as input, i.e. it is impossible to Concatenate A with nothing. As we saw, however, Merge can form {A} out of A and its restriction on $n > 1$ elements has to be stipulated. Furthermore, Merge applied to two identical objects A and A yields {A, A}, which is identical to the set {A}. This is clearly an unpleasant result, since there are two logical ways to arrive at the non-starter set {A}, both of which have to be excluded by stipulation.

One positive aspect of Concatenate is that it trivially satisfies the NTC. It does nothing more than take two atomic elements and stick them together. It is also blind to what these elements are, as long as they are atoms for the purpose of concatenation. In that sense it is not an operation specific to language, but it is so basic that it could be plausibly used across the board by various cognitive systems. In other words, Concatenate is a necessary (but clearly not sufficient) operation to generate a system such as natural language.

Chametzky (2000:128) already investigates concatenation as possible replacement of Merge but ends up rejecting this idea. According to him concatenation is “too minimal”, as it only yields flat structures. He is certainly right in pointing to out that concatenation alone does not allow us to capture our second descriptive generalization in (2), namely that sentences are hierarchically structured. Hornstein thus adds a second operation that allows the system to move from flat strings to hierarchical structures. Concatenate takes only atoms as inputs, so an already concatenated complex object could only be input to a further application of Concatenate if it was atomic itself. This is made possible by a second operation: Label.

- (5) Concatenate: $A, B \rightarrow A^{\wedge}B$
 Label: $A^{\wedge}B \rightarrow [A^{\wedge}A^{\wedge}B]$
 (= (8), p. 4)

Labelling is a very old concept in the tradition of Generative Grammar. It goes back at least to Chomsky’s (1955) “is-a” relationship. Translating it into modern terminology, we immediately see that the “is-a” relationship is nothing other than labelling together with a strict interpretation of BPS, which - first proposed as early as (Muysken 1982) - has become one of the standard assumptions of Minimalism and follows directly from the Inclusiveness Condition (Chomsky, 1995: 228).

(6) **Inclusiveness Condition**

Any structure formed by the computation is constituted of elements already present in the lexical items selected for N(umeration); no new objects are added in the course of the computation apart from rearrangements of lexical properties (in particular no indices, *bar levels in the sense of X-bar theory.*) (emphasis added)

Since the derivation is not allowed to add anything to the atomic elements, the operation Label can yield nothing else than the “is-a” relationship. Let us see what this buys us. When the concatenated string A^B projects the syntactic object $[_A A^B]$ is formed, and when labelling is construed in the sense just explained, it follows that the label A is identical to the atom A. We know that A must be an atom since A has been concatenated and only atomic elements are eligible for this operation. Now A is atomic again for the purpose of concatenation and is thus subject to further applications of Concatenate.

(7) $C^[_A A^B]$

The second application of Concatenate to the element $[_A A^B]$, which is complex as far as its derivational history is concerned, yet atomic for the purpose of Concatenate, unambiguously yields a nested structure. Reiteration of this sequence of steps generates a potentially infinite hierarchical structure. So far so good, but let us take a closer look at the assumptions Hornstein has to make to arrive at these results.

1.1 Where does Labelling come from? Concatenate arguably is the most primitive operation for connecting two elements together. Clearly, such a basic procedure cannot be unique to language and must be used by other cognitive system in abundance. It is thus easy to agree that Concatenate comes for free given our descriptive properties about natural language. But what about the second operation needed, Label? Hornstein argues that Label is the “evolutionary innovation” (5) language picked up to yield recursive embedding. Even if this speculation happens to be true it does not explain why and when Label should apply. Even though Hornstein rids his system of the implicit differentiation between two distinct elementary operations (AM and CM), there is no motivation for Label to apply other than the need to somehow generate hierarchy. Furthermore, it is also completely arbitrary when Label applies, i.e. why is it that it yields binary and not n-ary structures. In other words, Hornstein elicited the stipulated double function of Merge by splitting it up into Concatenate and Label, but he did not provide insight into when and why Label should apply.

What do we need to elevate the strings created by Concatenate to nested structures? An operation like Label obviously does the trick, yet the question remains why Label should apply in the first place. Asked more generally, why do we have hierarchical structures at all? This proposal gives us the opportunity to actually derive this observational generalization about language rather than take it as a primitive. Chametzky (2000) in his discussion of concatenation contends that “the resources of this theory [i.e. concatenation (JJ)] do not allow the construction of the hierarchically structured objects that PS-based approaches assume sentences to be.” (128). This line of reasoning,

presented as a truism, goes beyond the ontology of a syntactic system and premises claims about the nature of the interfaces. Let me elaborate.

The MP assumes that language is a perfect mapping from sound to meaning, i.e. it satisfies the interface conditions to the Conceptual-Intentional (C-I) system and the Articulatory-Perceptual (A-P) system in an optimal way. (cf. MP: 168). As noted by Higginbotham (1983:150-151), “the ordering of formatives in speech [...] is a consequence of the application of the laws of physics to the human mouth”. In other words, pure physics of speech restrict the A-P interface to linearized structures. What about the nature of the C-I system? Scope and binding facts, among many others, provide overwhelming evidence that the C-I system must be able to make use of hierarchical structures. Yet the fact that the C-I system *can* read embedded structures does not mean that it can *only* read embedded structures. Chomsky (1956) provides a means to makes these rather vague notions more concrete. According to his well-known containment hierarchy (referred to as Chomsky-Hierarchy) every level of complexity has every level of less complexity as proper subset. A system of level two or three, which is able to read hierarchical structures, by necessity also has the ability to read flat structures.

Returning to Chametzky’s rejection of concatenation on the grounds of the nature of the C-I system, we can conclude that his arguments cannot stand the way they are. Clearly, there is massive evidence that sentences show hierarchical structures, but this does not imply that hierarchy is a requirement imposed upon the system by the C-I interface. He, in concord with Chomsky, is right in claiming that given the need for hierarchy and *only* hierarchy, an operation such as Merge would be the correct approach. Then you could also argue that Noahistic Merge (Chametzky, 2000:129), i.e. binary Merge, is the simplest operation to satisfy this requirement. It is a major claim of this paper that we mistake the ability of the C-I system to read hierarchical structures with the requirement that it be fed exclusively with such structures. Assuming Concatenate as the most elementary mechanism of syntax makes interesting predictions. Most importantly, it forecasts that language also makes use of flat (non-binary) structures. These are trivially compatible with the A-P interface but also perfectly intelligible for the C-I system. However, we still have not answered the question of how we arrive at embedded structures from our only operation Concatenate. This is going to be the subject of the next subsection.

2 Feature Communication

Where are we now? We adopted Concatenate as the most fundamental operation of syntax and thus have a system which can create flat strings of infinite length. We rebutted objections that a system based on Concatenate is too minimal in that it does not create hierarchical structures. Label was shown to be the additional mechanism needed to yield nested structures, yet it has a clearly axiomatic character.

Let me now hint at how the system presented here is moving from strings to hierarchy without merely stipulating what is needed as a primitive

operation. The core of the argument will consist of the claim that the most restrictive reading of the Inclusiveness Condition can place the trigger for hierarchy only on lexical items (LIs). This section will propose that certain natural conditions have to hold for LIs to be able to communicate with each other. In other words, a certain configuration has to hold for LIs to satisfy their internal requirements, i.e. to check their μ F. For this purpose we will resuscitate an old concept which featured prominently in the early eighties, the notion of paths. We will be especially interested in Kayne’s (1981) unambiguous paths requirement. Let us first recapitulate his motivations to postulate it before seeing how it carries over into our system.

2.1 Paths – Back to the Eighties In the early eighties at least three major works suggested the concept of paths: (Kayne, 1981; May, 1985; Pesetsky, 1982). In the context of tree-theoretical notation paths seemed to be a perfectly natural concept. Paths were defined along the line of (8) and (9).²

(8) **Informal definition of paths** (May, 1985:118)

A path is a “set of occurrences of successively immediately dominating categorial nodes”.

(9) **Formal definition of paths** (Kayne, 1981:146)

- a) $\exists i, j \quad 0 \leq i, j \leq n \quad A_i = A_j \rightarrow i = j$
 b) $\exists i \quad 0 \leq i < n \quad A_i$ immediately dominates A_{i+1} or A_{i+1} immediately dominates A_i

(8) is fairly straightforward and captures the fact that the notion of paths is given once you have a phrase marker represented as a tree. (9a) makes sure that paths only comprise a sequence of distinct nodes. We want to avoid the system to move back and forth between the same nodes. (9b) defines an adjacency requirement for paths, i.e. we do not want our system to skip any nodes. It is important to notice that these definitions do not introduce any new principles. Paths are a necessary by-product of trees and thus as natural as domination or sisterhood and do not have any of the stipulative character of government or c-command, which pick specific relations out of the tree and define them as relevant.

2.2.1 Kayne’s Unambiguous Paths Since Reinhart (1976)³ the notion of c-command gained prominence amongst generative theory. Chomsky (1981) established it as the fundamental relation between two nodes. Most syntactic principles of GB were defined in terms of c-command, e.g. the ECP,

² Pesetsky (1982) defines paths as a sequence of maximal projections only. This is clearly irrelevant in a BPS approach.

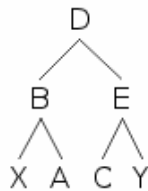
³ The concept of c-command actually dates back to Klima’s (1964) “in construction with”, which is inverse c-command and as such very similar to Kayne’s (1981) Unambiguous Paths.

Government, Binding etc. Even though something along the lines of c-command seemed to be a guiding principle of the grammar, early definitions lacked the naturalness preferable for theoretical principles. When Manzini (1983) and Aoun & Sportiche (1983) developed m-command to design a generalized configuration for theta-role assignment, it became obvious that various command relationships could be defined almost freely and arbitrarily to accommodate certain phenomena. This was noticed by Kayne (1981), who sought a more intuitive replacement of c-command:

- (10) **Unambiguous Path (UP)** (for $i, 0 \leq i < n$)
- a) if A_i immediately dominates A_{i+1} , then A_i immediately dominates no node other than A_{i+1} , with the permissible exception of A_{i-1}
 - b) if A_i is immediately dominated by A_{i+1} , then A_i is immediately dominated by no node other than A_{i+1} , with the permissible exception of A_{i-1}

This formal definition is easily translatable in plain English. When following a well-formed path the system “is never forced to make a choice between two (or more) unused branches, both pointing in the same direction” (1981:146). Let me illustrate this configuration with concrete trees. Consider the relationship between X and Y in (11).

(11)



(12)



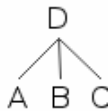
The path from Y to X in (11) is the set of nodes $\{Y, E, D, B, X\}$ and it is $\{Y, E, D, X\}$ in (12). The crucial difference between these paths, however, lies in the fact that in (12) there is no node at which the system has to make a choice as to where to continue.⁴ In (11), on the other hand, there is a bifurcation at B,

⁴ Following a general constraint of direction, going from E to B is blocked.

i.e. the system has the choice between continuing to X or to A. As a result, the path in (12) is taken to be an UP while the one in (11) is well-formed but not unambiguous. Note that the unambiguous paths relationship is to a large part c-command through the looking glass. Following Chomsky's (1981) definition, in (11) X does not c-command Y since the first branching node X is dominated by does not dominate Y. In other words, the c-command relationship is blocked by the presence of the branching node B. The omission of B in a structure such as (12) would allow for a c-command relationship.

An UP thus does not allow any choice points, i.e. there must not be more than one way from which the path could be continued from a specific node. Kayne's main incentive for this proposal was to design a more natural replacement of the c-command relation. Since we will make extensive use of the concept of UPs, it seems worthwhile to compare it to the better known c-command relation. The crucial difference, the one which will be vital for our purposes, can be illustrated by the simple tree in (13).

(13)



There is mutual c-command between the three terminal nodes in (13), i.e. each node both c-commands every other node and is c-commanded by every other node. It is obvious, however, that none of the nodes are connected through an UP. There are always two choices to continue from node D. This naturally extends to any $n > 2$ branching structure. Kayne's theory, then, quite nicely derives what other approaches only stipulated as one of the axioms of the system.⁵

2.2 Establish Dependency At this point we have left our path and taken a quite convoluted detour on our way from the flat structures constructed by Concatenate to hierarchy. Or so it seems. The discussion of UPs is central to my proposal, which will become apparent as we move on. Let us thus fully adhere to Kayne's idea of establishing UP as the only long distance relationship in syntax. Immediate domination and command of any sort are only consequences of UP to the extent that concepts such as these are needed at all. Let us furthermore try to incorporate his notion to our minimalist system of syntax.

Before that, however, we have to settle the issue of whether we really need any form of long distance dependency formation operation. Could we not just follow the spirit of Hornstein (2001) and Chandra (2007) (among others) and claim that everything is Move, i.e. that there is no need for a long distance

⁵ See Jurka (2006: 46ff.) for a detailed comparison between UPs and c-command.

dependency relation because the two dependents are connected to each other by virtue of being the same thing, given the copy theory of movement? Hornstein (2001) shows that systems which distinguish between a lexicon and syntax proper have to rely on an operation like Copy. Given Copy and Concatenate, there is everything we need to move constituents, and displacement might be the correct way of analysing what appear to be long distance relationships. However, I would like to argue that this approach is problematic for conceptual reasons and that some long distance dependency operation is needed. I will embrace the chief aspects of Chomsky's (2000) Agree but replace it by an operation defined in terms of UPs, which we will call Establish Dependency.

If two LIs A and B contain μ Fs the system must provide some way for them to find out where they can find a potential satisfier of their featureal requirement. This is an informal way of stating Chomsky's (2000:122) Match relation. He basically takes Match to be a necessary subcomponent of Agree. Without such an operation any feature based approach would be void since there would be no way for μ Fs to find out where they could find their needed interpretable counterpart. Some matching mechanism which allows LIs to search the tree for appropriate partners must have been assumed implicitly even in the earliest checking system. Even if movement, being a result of Copy and Concatenate, is costless, we still want our system to be purely Last Resort driven. Otherwise, there would be no reason why it should not allow perpetual displacement reminiscent of Move α .⁶

A shrewd reader will have noticed that Agree crucially relies on c-command to define its search domain. In the previous section we abandoned any form of command relation from our system in favour of an approach in terms of UPs. C-command, however, is not an inherent property of a long distance relationship formation operation and can easily be reformulated using the concept of UPs. Feature checking is made possible only via the Unambiguous Paths Strategy (UPS).

(14) **Unambiguous Paths Strategy (UPS)**

A head X's uninterpretable feature F can only be deleted iff it is connected to its interpretable counterpart by an unambiguous path.

UPS is the only means of communication between elements in the course of the derivation.⁷ It is the only valid configuration for checking features. Note that it is independent of distance, i.e. it is equally satisfied by a binary sisterhood-relationship as well as by a potentially infinite long distance dependency. Crucially, it follows from the notion of UPs, as the inverse of c-

⁶ This paper remains agnostic to whether the empirical evidence for Agree holds or not. The reader is referred to Boeckx (2001, 2004) or Chomsky (2000, 2001, 2004) for arguments in favour of Agree. See Hornstein (2001) and Chandra (2007) for arguments against it.

⁷ Hornstein (p.c.) correctly points out that this would rule out Sideways Movement. This could be seen as I either a virtue or a vice of the theory.

command, that we have to assume the lower constituent as our point of departure. UPS could thus be read as inverse Agree, i.e. the uF is on the goal. UPS necessitates the view that what is generally considered to be the goal must now be our probe. Arguably both vantage points are equally plausible and, as a matter of fact, both have been proposed in the minimalist program. Chomsky (Chomsky, 1995:201) introduces the principle Greed, which dictates that an element A can only enter into a syntactic operation if some property, i.e. some uF , needs to be satisfied. He illustrates this with (15)

(15) seems [_{α} to a strange man] [that it is raining outside]

The uninterpretable case feature of *a strange man* is checked by the proposition *to* and as a result Greed prevents any other operations such as Move from applying. Greed cannot be overridden even when the derivation is bound to crash. I am aware of the fact that later additions to the system such as Attract or Agree take up the opposite point of view. Greed is incorporated into Attract by assuming that uF s on functional heads are greedy and attract elements to have their features checked. Conceptually, I do not see good reasons for why either approach should be superior. Empirically there seem to be arguments for both positions. Some features quite strikingly seem to be interpretable on the lexical item and uninterpretable on the higher functional head, e.g. wh-features. Other features such as discourse features are likely to be uninterpretable on the lexical items and interpretable on some higher focus or topic head. With case, it seems that both points of view are possible depending on how you construe case. Following Pesetsky & Torrego (2001; 2004) structural case is an uninterpretable tense feature on the DP. Only a Greed-based approach can trigger a relationship between the DP and the higher T, which can carry anything *but* an uninterpretable tense feature, as this is basically what it is. It seems that features are still too poorly understood to provide sound arguments for one point of view or the other. This paper subscribes to the earlier Greed-based approach.

Assuming this, let us return to UPS. What advantages does it have for our system? We have finally arrived at a syntactic condition which seems to upgrade our linear system to hierarchy in a natural way. Let us see how this works. All UP requirements discussed above apply to UPS. As a result, we need every two nodes which must undergo checking to satisfy Full Interpretation to be in a UP relation, which presupposes that they are connected by a valid path in the first place. UPs crucially rely on a tree-theoretical notation of phrase markers. A simple tree of two elements to which only the operation Concatenate has applied would look like (16).

(16)



Both definitions of paths in (8) and (9), repeated as (17) and (18) for convenience, do not cover a tree like (16), as A and B are not connected by “a set of occurrences of successively immediately dominating categorial nodes”.

(17) **Informal definition of paths** (May 1985:118)

A path is a “set of occurrences of successively immediately dominating categorial nodes”.

(18) **Formal definition of paths** (Kayne 1981:146)

- a) $\exists i, j \quad 0 \leq i, j \leq n \quad A_i = A_j \rightarrow i = j$
- b) $\exists i \quad 0 \leq i < n \quad A_i$ immediately dominates A_{i+1} or A_{i+1} immediately dominates A_i

Neither does A immediately dominate B nor vice versa. It is crucial to emphasize that this does not imply that (16) is not a valid phrase marker per se. The fully legitimate operation Concatenate has applied to two atoms, forming the concatenate A^B . A and B, however, are both likely to carry μ Fs which are properties of LIs. (18) does not allow for the establishing of a well-formed path and thus trivially also does not provide a UP configuration. The μ Fs of either A or B would not be able to find their interpretable counterparts, and so are frozen in the structure causing the derivation to crash because Full Interpretation is violated.

How does the system salvage the derivation? The obvious answer is labelling. For instance, say A bears an μ F which could be checked by B. First, let us assume the most radical reading of the Inclusiveness condition in that there are not only no new objects added in the course of the derivation which have not been already part of the Numeration, but let us confine the freedom of syntax even more by contending that a syntactic derivation is fuelled only by the LIs it operated on from the beginning. Under these assumptions, syntax only provides a restrictive formal system comprising the most elementary operations logically conceivable. I have suggested these operation to be Concatenate and Copy supplemented by UPS. As already stated, these operations do not apply freely, but as Last Resort in the sense of being triggered by LIs only. The reason for LIs to trigger operations is their need to satisfy some formal property, conventionally termed μ F. The following hypothesis seems a logical consequence of such a line of reasoning.

(19) **Uninterpretable Feature Hypothesis (UFH)**

The need to delete uninterpretable features in the course of the derivation triggers labelling.

Returning to our example, A does not know that B bears the interpretable counterpart to its uF since it cannot communicate with B, as UPS does not hold. The uF in A thus triggers labelling and projects to form a tree like (20).

(20)



There is now not only a perfectly well-formed path between A and B but this path is also an UP. The set-up for A and B to be able to interact with each other is provided, A can check its uF with its interpretable counterpart on B, and the derivation can continue and potentially converge. Note that we have derived exactly the configuration normally referred to as *First Merge*.

Let us recapitulate what we have achieved. It is necessary to put strong emphasis on the fact that we have done more than simply adopted Hornstein's operation Label, as presented in section 1. It is not a primitive of our system but follows naturally from UPS as the only viable configuration for the satisfaction of feature requirements. Most importantly, however, the ideas suggested here have a more far-reaching conceptual consequence to the theory of syntax. We have detached hierarchy from its role as an axiom of syntax and are able to make it follow from idiosyncratic properties of the lexicon. Hierarchy is thus not a "virtual conceptual necessity" as neither of the interfaces is likely to be exclusively dependent on hierarchical structures. The C-I interface conveniently is able to read such structures, a capacity which is also employed for many non-linguistic cognitive competences. Even though this seems to be a new hypothesis about how language works, this result is not too surprising given the context of the minimalist endeavour. Our syntactic model consists of atomic elements, i.e. LIs, and the most elementary possible mechanisms of relating these elements to each other. Ultimately, there are the interface conditions imposed by the A-P and the C-I system. Syntax is thus purely interface driven in that it combines atoms in such a way as to rid them from their idiosyncratic properties, formally encoded as uFs , which are illegible at the interfaces.

If we venture a comparison between natural and formal language, it is mainly the presence of such idiosyncratic properties of the atoms of the system which differentiate between the two. Natural language is thus optimal in the same way as formal languages in that it translates sets of LIs into phrase markers legible at the interfaces. Formal languages do essentially the same thing, i.e. they take atomic elements as their input and provide an optimal

mapping from sound to meaning. These atomic elements, however, do not show the same intricate properties as LIs of natural language do. I consider this to be the main and ideally the only difference between the two.⁸

3 Adjuncts

Now we have not only derived labelling, viz. hierarchy. But furthermore, adopting relation like UPS bestows upon us all the other benefits of UPs. Recall that one of the main differences between the former and c-command was that UPs only allow binary branching. How is this implemented into our system? It follows that binary structures are the only viable configurations for feature checking since n-ary ($n > 2$) structures are incompatible with UPS. Note that this does not mean that n-ary structures are *prima facie* impossibilities of the system. To the contrary, we have shown that Concatenate can form linear strings of potentially infinite length. We do not want to exclude them in any arbitrary manner. The UFH, however, dictates that labelling is triggered by *u*Fs. In a situation where there are no *u*Fs to be satisfied, there is no need to label and thus the structure remains flat. As we have hinted at above, this conceptual prediction is instantiated by adjuncts. Adjuncts can be self-satisfying in that they check all their *u*F internally. As a result, there is no need to communicate with the rest of the tree and UPS is vacuously satisfied.

Adjuncts have been treated as the poor cousin of most phrase structure theories both in GB and in the MP. At various occasion, for example, Chomsky admits that adjuncts and adjunction are still far from being well understood. Chomsky (1995:382, fn. 22) notices “we still have no good phrase structure theory for such simple matter as [...] adjuncts of many different types” while Chomsky (2004:117) points out that “there has never, to my knowledge, been a really satisfactory theory of adjunction, and to construct one is no slight task”. This section gives a brief overview over some descriptive generalizations, which seem to hold ubiquitously across languages, and we will briefly examine some prominent proposals for how to encode adjuncts within phrase structure theory. Finally, we will use the mechanisms introduced in the last chapter to show that nothing mysterious has to be added to embed adjuncts into our framework. In fact, we will see that adjuncts are much more well-behaved than arguments in that they may be grammatically self-sustainable, needing no structural licensing from other elements of the tree. This independence exempts them from our UPs requirements, since this only has to hold when two elements need to “talk” with each other. When such communication is not necessary, UPS is satisfied trivially. As a result, labelling does not have to apply in the case of adjuncts. Note that this idea has a clear precursor in Chametzky (1996). However, while he more or less

⁸ It follows trivially from such considerations that language acquisition is essentially the learning of LIs and their various properties. I take all syntactic operations to be universally anchored in UG and thus innate. Additionally, cross-linguistic variation is also restricted to different properties of the LIs (cf .Chomsky, 1995). Parameters, if they exist at all, are harboured in the lexicon and not in syntax.

stipulates that adjuncts have the property of not being labelled because they are adjuncts, this property follows neatly from our system. In other words, there is no *prima facie* distinction between arguments and adjuncts. Their descriptively different behaviour can be fully derived from the latter's property of self-sustainability, which the former clearly lacks. Speaking metaphorically, our system puts the cart before the horse, in treating adjuncts as the canonical and arguments as the non-canonical case⁹. This is exactly the inverse approach to (Chametzky, 1996), as we shall see later.

The fact that adjuncts *can* be self-sustainable and thus do not trigger labelling does not mean that they *must* have this property. As soon as adjuncts have an *uF* they cannot check within their own domain, the UFH dictates that labelling is triggered. Such an *uF* could, for example, be a discourse feature of some kind which needs the constituent to move to some discourse head to be checked. We will see how this derives a host of descriptive properties of adjunct movement.

In the literature adjuncts are often treated together with adjunction. These notions are intimately related to each other and it would be desirable for a theory of phrase structure to treat them in a unified way, yet one must be aware of the fact that they are distinct concepts. Adjunction, often referred to as Chomsky-adjunction, is a syntactic movement operation. In classical GB it was, for example, used to map S-structure to LF (cf. May 1985:53ff.). Adjuncts, on the other hand, are non-obligatory modifiers. It comes as no surprise that these phenomena have been treated together, since adjuncts were taken to show the configuration determined by Chomsky-adjunction¹⁰. Consider the structure in (21):

(21)



Following the discussion in Hornstein & Nunes (2005), we note that in (21) YP has adjoined to XP and the product of this operation has been given the label XP. The lower XP could be called *host* while the higher XP is the new mother of the projection. The mother and the daughter both have the status of a maximal projection. It is easy to note that such a structure is purely stipulated to account for the descriptive properties of adjuncts, while it is theoretically strongly questionable. Especially the label of the mother comes as quite of a surprise, in that the very concept of maximality is reduced to

⁹ This is also noted by Uriagereka (2001: 14): “If modification is more basic than argument taking, adjunction ought to be more basic than merger.”

¹⁰ According to May (1985: 158, fn:1) “Chomsky-Adjunction of a constituent β to a node α yields structures either of the form ‘ $[_\alpha \beta [a\dots]]$ ’ (left Chomsky-Adjunction) or the form ‘ $[_\alpha [a\dots] \beta]$ ’ (right Chomsky-Adjunction).”

absurdity. How can a maximal node such as the host be dominated by a second maximal projection of the same head? Clearly, this implies that labels such as X^0 , X' or XP are purely arbitrary terms which pretend to bear information on the hierarchical status of the node. In fact, (21) premises two contradictory assumptions. Either the label XP actually carries the meaning of maximality, not in a relational but in an ontological sense. In that case an XP dominated by another XP is impossible by definition, and (1) would be an ill-formed phrase marker. The other possibility would be to construe X^0 , X' or XP just as markers of the derivational history, in the sense of “ X^0 is the first instance of X in the tree”. Then these labels could be given the more transparent names X^1 , X^2 , X^3 , ..., X^n . Again we would arrive at the quite undesirable conclusion of having to assume X^n dominating X^n and not X^{n-1} .

In BPS terms (21) is simply undefinable and cannot be maintained. Bar-levels have a relational meaning at best, which means that having both an XP mother and daughter is ruled out. However, (21) was capable of encoding the most important descriptive generalization about adjuncts and has thus been accepted for the lack of a better alternative. (Hornstein et al., 2005a) note five distinctive properties of adjuncts, which I summarize in (22):

- (22) **Descriptive Properties of Adjuncts** (cf. Hornstein & Nunes, 2005)
- a.) bar-level information is conserved
 - b.) category information left intact
 - c.) headedness is preserved
 - d.) Adjunct-plus- XP is itself maximal
 - e.) There is no upper bound on number of adjuncts

All of these properties can be observed empirically across languages. (22a) implies that operations which target a certain projection are affected by the presence of adjuncts (e.g. VP-fronting in (23) and *one*-substitution in (24))

- (23) a. Bill was allowed to [drink beer] and [drink beer] he did.
 b. Bill was allowed to [[drink beer] in the bar] and [[drink beer] in the bar] he did.

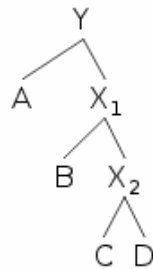
- (24) This [[[student of linguistics] with short hair] from Spain] and
- a. that one.
 - b. *that one of physics
 - c. that one from Portugal
 - d. that one with long hair
 - e. that one from Portugal with long hair

Sub-categorization is generally taken to be determined by the head of phrase. (25) illustrates (22c) in that adjuncts do not affect the selection properties of a phrase. The auxiliary *have* used for present perfect tense selects the past participle of a verb. This remains unchanged when adjuncts are added.

- (25) a. Peter has visited his friend.
 b. Peter has visited his friend in Mösendorf.
 c. Peter has visited his friend in Mösendorf regularly .

Any phrase structure theory has to account for these facts in one way or another. Chomsky-Adjunction does a fairly good job but is, as we saw, incompatible with BPS and thus violates Inclusiveness. This was recognized by Chomsky (1995), who proposes a somewhat different approach to encoding adjuncts. He builds on the distinction between *category* and *segment* introduced by May (1985:57). To illustrate this distinction, consider the tree in (26).

(26)



In (26) B has been adjoined to X. As a result, the projection of X now consists of two occurrences of X, X₁ and X₂. In other words, X₁ and X₂ are two segments of the category X, and only C and D are dominated by X while B is only contained in X. Chomsky (1995a:248) picks up on this and takes adjunction to form “a two-segment category, not a new category”. He continues to point out that “there must be an object constructed from K but with a label distinct from its head H(K). One minimal choice is the ordered pair <H(K), H(K)>”. We arrive at a structure roughly like (27).

(27) [_{<x, x>}[_x ...X...] adjunct]

The structure has been made compatible with BPS, since the labels of the adjunct and the element it is adjoined to are different, i.e. they can both be maximal without violating BPS. However, returning to (22) and the descriptive properties of adjuncts, we see that this analysis of adjuncts in terms of pair-merge misses an important generalization the classical Chomsky-Adjunction structure encoded. We have lost a way of predicting that adjuncts and the elements they are adjoined to show syntactically indistinguishable behaviour. This could be expected under an [XP[XP]] structure, since two elements with identical labels should behave identically. Recall that Chomsky-Adjunction was of course stipulated to accommodate precisely this property, but at least there was a way to structuralize it. The MP account solves the BPS

structure problem by assuming that the adjunct has a different label and loses the generalizations of the earlier approach. This does not seem to be the most desirable result.

A proper understanding of the application of labelling is key in providing a sensible way of encoding adjuncts. To my knowledge, Chametzky (1996) was the first to critically evaluate this relationship. His main claim is that it is viable for a “new mother node in the adjunction structure to remain unlabelled” (112). This is corroborated by the lack of strict semantic compositionality of adjuncts, and the fact that we find simpler semantics in terms of predication. Neo-Davidsonian event semantics (cf. Parsons, 1990) realize this fact and treat adjuncts in a conjunctive fashion.

- (28) a. Dick shot Harry with a rifle on a hunting trip.
b. $\exists e$ [shot (e) & Agent (Dick, e) & Patient (Harry, e) & with a rifle (e) & on a hunting trip (e)]

What is relevant here is the fact that *Dick* and *Harry* modify the event of shooting in their respective roles as agent and patient. The adjuncts, on the other hand, apply directly to the event. On the phonological side, it has been suggested (cf. Wagner, 2005) that adjuncts show prosodic structures similar to lists.

Observations along these lines prompted Uriagereka (2001; to appear) and Gallego (2005) to postulate a flat structure for adjuncts. As we saw, our system provides a perfectly straightforward way of coding these proposals. Hornstein & Nunes suggest that labelling is optional for adjuncts. They argue correctly that such an approach is compatible with BPS (unlike Chomsky-Adjunction), while the descriptive properties of adjuncts can be captured (unlike in the pair-merge approach). The downside of their approach, however, lies in the invocation of optionality. Ideally, we would like our system to behave in a deterministic way without any optional rules with no clear triggers. In the following we will see how our system derives this apparent optionality in labelling with adjuncts without any additional assumptions.

Consider the following paradigm (cf. Hornstein & Nunes, 2005).

- (29) a.) Dick [[[shot Harry] with a rifle] on a hunting trip].
b.) It was shoot Harry that Dick did with a rifle on a hunting trip.
c.) It was shoot Harry with a rifle that Dick did on a hunting trip.
d.) It was shoot Harry with a rifle on a hunting trip that Dick did.
e.) *It was shoot that Dick did Harry with a rifle on a hunting trip.

English allows VP-fronting. Interestingly, we observe that the fronted VP may take any number of adjuncts with it (29b-d), while it is impossible to separate the verb from its complement (29e). How could we analyse this paradigm, without purely describing what is happening but actually explaining this

apparent optionality. The structure in terms of Chomsky-Adjunction, abstracting away from some irrelevant details, would look like (30).

(30) Dick [_{VP}[_{VP}[_{VP} shot Harry] with a rifle] on a hunting trip].

As we pointed out above, clearly, such a structure blatantly violates BPS, which itself follows directly from Inclusiveness. Bar-level notation, however, would allow us to give an accurate description of our paradigm. VPs can be fronted, while V's cannot. We have to find a different way to code this piece of data, without making reference to bar-notational terminology. We can reformulate (30) to make our structure adhere to BPS.

(31) Dick [_v[_v[_v shot Harry] with a rifle] on a hunting trip].

This structure creates a couple of problems, which were not present in the Chomsky-Adjunction version. Why is it that the V *shot* itself cannot move in (29e), if the outermost V and the innermost V are identical? An answer for questions such as this, however, was provided as early as Chomsky's (1964)¹¹ A-over-A condition. The principle dictated that in a structure such as [A...[A...]] only the outermost A can be a target for transformations. Hornstein is able to derive this condition in terms of shortest path, as we have shown in chapter 2.4. Trivially, the path from the outer A is shorter than the path from the inner A. So we have an explanation for why the V *shot* cannot move by itself leaving its complement behind, cf. (29e). However, a structure such as (31) is still agnostic to the optionality of the paradigm in (29), since only the case where the VP and all the adjuncts are fronted is expected.

What if we follow our contention that adjuncts need not be labelled? If adjuncts are unlabeled, it follows from UPS that they cannot be a target for movement since no well-formed path and thus no UP can be established. Our paradigm, however, suggests that adjuncts can be fronted so it must at least be possible to label them. Hornstein & Nunes notice that and ascribe the optionality of adjunct fronting to the optionality of labelling with adjuncts. In my view, this seems to be somewhat short of a genuine explanation. It amounts to saying that you label an adjunct whenever you move it and that you can move an adjunct only when it is labelled.¹²

In the system suggested here all operations have a Last Resort character and the optionality of a paradigm such as (29) is only an illusion. The expressions in (29a-d) all have completely different information structures, with only (29a) being neutral with respect to topic and focus. VP-fronting itself is essentially similar to clefting in that the clefted constituent has to be focus (cf. Prince 1978) and needs strong prosodic emphasis. In more technical

¹¹ Note that the term "A-over-A condition" was not used in the original paper by Chomsky, but was only introduced in Ross (1967).

¹² Hornstein & Nunes do note the focushood of the fronted element but are not explicit as to how and why this should trigger labelling.

terms, the fronted element bears an uninterpretable focus feature (*uFoc*), which has to be satisfied by some focus projection in the left periphery, the domain of discourse.

Returning to our specific problem, this means that adjuncts are self-sustainable unless they enter the derivation with an *uFoc* feature. In the latter case they need to find an interpretable counterpart, i.e. the focus head at the left edge of the clause, to have their feature checked. According to the UFH, the *uFoc* of the adjunct triggers labelling since the feature is still unchecked by the time the adjuncts have been fully assembled, and since the feature would be otherwise trapped in the structure without any prospect of being satisfied. After labelling has applied the adjunct can enter a relationship with the higher Focus head via UPS since an UP is now guaranteed, and movement can be triggered. The same procedure is employed for all other adjuncts carrying an *uFoc*. If they are discourse neutral, their feature matrix is checked internally to themselves and labelling is not triggered.

This construal of VP-fronting explains the paradigm in (29), with the only difference lying in the presence or absence of a discourse feature. Note that this analysis is not circular since we do not merely claim that we label when we move and we move as we label. The focushood of the fronted adjuncts seems to be a solid fact which can be easily elicited from speakers of various dialects of English. This fact is coded by our technical machinery in terms of an *uF*. From this point onwards we need not make any more assumptions since our system proceeds deterministically to yield the desired result. Optionality in syntax as such is illusory, since it must be always due to the presence of some feature.

This discussion of VP-fronting must be taken as only one example of how our system deals with adjuncts in a very natural way. Adjuncts can be self-sustainable in that they do not need to enter any relationship with other heads of the clause, i.e they check all of the features internally to themselves. This is the default case in which labelling is not triggered, and the adjuncts, metaphorically speaking, dangle loosely at some node of the tree. However, it might be the case that some feature remains unchecked by the end of the compilation of the adjunct. The only possible way to reconcile this is finding an adjunct-external head somewhere in the tree to satisfy this feature. UPS dictates that this communication between features is only licensed in UPs configurations. The *uF* thus triggers label and the entire adjunct can then be a target for intra-arboreal operations.

4 Conclusion

This paper introduced the possibility of deriving hierarchy from the need to provide appropriate configuration for syntactic items to communicate with each other. I tried to show that a marriage of Hornstein's operation Concatenate and Kayne's UPs allows us to fairly straightforwardly derive nested structures. The system also allows for the possibility of flat structures

for certain phenomena such as adjuncts, which was argued to have desirable empirical consequences.

Acknowledgments

Needless to say, this article is strongly influenced by Norbert Hornstein's recent work. The central idea of this contribution stems from a seminar paper for his class. I would also like to thank him for extensive comments and discussion of earlier versions. Thanks also go to Cedric Boeckx and the MWPL16 committee, Akira Omaki, Ivan Ortega-Santos, Jon Sprouse and Matt Wagers.

References

- Aoun, Joseph, and Sportiche, Dominique. 1983. On the Formal Theory of Government. *Linguistic Review* 2:211-236.
- Boeckx, Cedric. 2001. Quirky Agreement. *Studia Linguistica* 54:354-380.
- Boeckx, Cedric. 2004. Long Distance Agreement in Hindi: Some Theoretical Implications. *Studia Linguistica* 58:1-14.
- Chametzky, Robert. 1996. *A theory of phrase markers and the extended base*: SUNY series in linguistics. Albany: State University of New York Press.
- Chametzky, Robert. 2000. *Phrase structure : from GB to minimalism*: Generative syntax ; 4. Malden, Mass.: Blackwell.
- Chandra, Pritha. 2007. *(Dis)Agree: Movement and Agreement Reconsidered*. PhD, University of Maryland, College Park, Md.
- Chomsky, Noam. 1955. The Logical Structure of Linguistic Theory. Ms., *Harvard, University*. Cambridge, Mass.
- Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 2:113-124.
- Chomsky, Noam. 1964. *Current issues in linguistic theory*. The Hague,: Mouton.
- Chomsky, Noam. 1981. *Lectures on government and binding*: Studies in generative grammar ; 9. Dordrecht, Holland ; Cinnaminson, [N.J.]: Foris Publications.
- Chomsky, Noam. 1995. *The Minimalist Program*. Cambridge, Mass.: MIT Press.
- Chomsky, Noam. 2000. Minimalist Inquiries: The Framework. In *Step by step : essays on minimalist syntax in honor of Howard Lasnik*, eds. Howard Lasnik, Roger Martin, David Michaels and Juan Uriagereka, 89-115. Cambridge, Mass.: MIT Press.
- Chomsky, Noam. 2001. Derivation by Phase. In *Ken Hale : a life in language*, eds. Kenneth L. Hale and Michael J. Kenstowicz, 1-52. Cambridge, Mass.: MIT Press.
- Chomsky, Noam. 2004. Beyond Explanatory Adequacy. In *Structures and beyond*, ed. Adriana Belletti. New York: Oxford University Press.
- Chomsky, Noam. 2005. On Phases. Ms., *MIT*. Cambridge, Mass.

- Citko, Barbara. 2005. On the Nature of Merge: External Merge, Internal Merge, and Parallel Merge. *Linguistic Inquiry* 36:475-496.
- Gallego, Angel. 2005. Connectivity in Markovian Dependencies. Ms., *Universitat Autònoma de Barcelona*. Barcelona, Spain.
- Higginbotham, James. 1983. Logical Form, Binding, and Nominals. *Linguistic Inquiry* 14:395-420.
- Hornstein, Norbert. 2001. *Move! : a minimalist theory of construal: Generative syntax ; 5*. Malden, Mass.: Blackwell.
- Hornstein, Nobert. 2005. What Do Labels Do: Some Thoughts on the Endocentric Roots of Recursion and Movement. Ms., *University of Maryland*. College Park, Md.
- Hornstein, Nobert, and Nunes, Jairo. 2005. Some Thoughts on Adjunction. Ms., *University of Maryland*. College Park, Md.
- Jurka, Johannes. 2006. *Paths Untrodden: A Minimalist Theory of Phrase Structure*. M.A. University of Vienna, Vienna, Austria.
- Kayne, Richard S. 1981. Unambiguous Paths. In *Levels of Syntactic Representation*, eds. Robert May and Jan Koster, 143-183. Dordrecht, Holland ; Cinnaminson, N.J., U.S.A.: Foris Publications.
- Klima, Edward. 1964. Negation in English. In Fodor, Jerry and Katz, Jerrold. *The Structure of Language: Readings in the Philosophy of Language*, 246-323. Prentice-Hall, Englewood Cliffs, N.J.
- Lasnik, Howard, and Saito, Mamoru. 1992. *Move [alpha] : conditions on its application and output*. Cambridge, Mass.: MIT Press.
- Manzini, Rita. 1983. On control and control theory. *Linguistic Inquiry* 14:412-446.
- May, Robert. 1985. *Logical form : its structure and derivation: Linguistic inquiry monographs ; 12*. Cambridge, Mass.: MIT Press.
- Muysken, Pieter. 1982. Parametrizing the Notion 'Head'. *Journal of Linguistic Research* 2:57-75.
- Parsons, Terence. 1990. *Events in the semantics of English : a study in subatomic semantics: Current studies in linguistics series ; 19*. Cambridge, Mass.: MIT Press.
- Pesetsky, David. 1982. Paths and categories: Thesis Ph.D. --Massachusetts Institute of Technology Dept. of Linguistics and Philosophy 1983.
- Pesetsky, David, and Torrego, Esther. 2001. T-to-C Movement. In *Ken Hale : a life in language*, eds. Kenneth L. Hale and Michael J. Kenstowicz, 355-426. Cambridge, Mass.: MIT Press.
- Pesetsky, David, and Torrego, Esther. 2004. Tense, Case, and the Nature of Syntactic Categories. In *The syntax of time*, eds. Jacqueline Guéron and Jacqueline Lecarme, 495-538. Cambridge, Mass.: MIT Press.
- Prince, Ellen. 1978. A Comparison of Wh-clefts and It-clefts in Discourse. *Language* 45:883-889.
- Reinhart, Tanya Miriam. 1976. The syntactic domain of anaphora: Thesis. 1976. Ph.D.--Massachusetts Institute of Technology. Dept. of Foreign Literatures and Linguistics.

- Ross, John Robert. 1967. Constraints on variables in syntax: Massachusetts Institute of Technology. Dept. of Modern Languages and Linguistics. Thesis. 1967. Ph.D.
- Uriagereka, Juan. 2001. Pure Adjuncts. Ms., *University of Maryland*. College Park, Md.
- Uriagereka, Juan. to appear. *Syntactic Anchors*. Cambridge: Cambridge University Press.
- Wagner, Michael. 2005. Asymmetries in Prosodic Domain Formation. In *Perspectives on Phases*, eds. Norvin Richards and Martha McGinnis. Cambridge, Mass.: MIT Working Paper in Linguistics (MITWPL).