Minimalist grammars and recognition

Edward P. Stabler*

Abstract

Recent work has shown how basic ideas of the minimalist tradition in transformational syntax can be captured in a simple generative formalism, a "derivational minimalism." This framework can model "remnant movement" analyses, which yield more complex antecedent-trace relations, suggesting a new and significant sense in which linguistic structures are "chain based." Michaelis (1998) showed that these grammars correspond to a certain kind of linear context free rewrite system, and this paper takes the next step of adapting the recognition methods for "non-concatenative" grammars (Weir, 1988; Seki et al., 1991; Boullier, 1999). This turns out to be quite straightforward once the grammars are set out appropriately.

1 Chain-based syntax

Some recent proposals in transformational syntax involve more movements of larger pieces of structure than have ever been seriously proposed before. According to some of these analyses movement, everything major constituent in a clause moves. For example, in an attempt to get certain linguistic universals about constituent order to follow from the architecture of human grammars, Kayne has proposed that there is no right adjunction and no rightward movement. These proposals are at odds with the earlier transformational analyses of structures like:

(1) They showed [a picture t_i] to me [that I like]$_i$

Here, it was commonly assumed that the relative clause was right adjoined to the noun phrase *a picture* and then right extraposed to its final position.

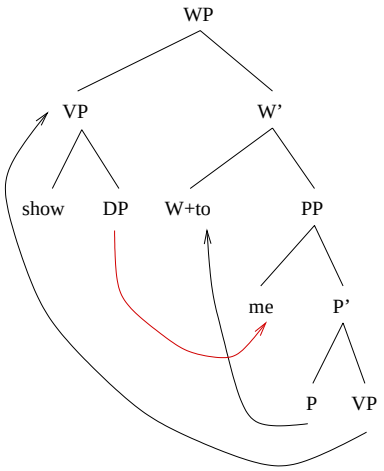
To account for the fact that VO languages tend to be prepositional, Kayne (1999) considers the possibility that P enters a derivation not by merging with its object, but rather entering above the VP, like other case assigners. Simplifying all other aspects of the construction (i.e. leaving out many movements), Kayne proposes a derivation like this:

*A version of this paper was presented under the title "Performance models for derivational minimalism" at "Linguistic Form and its Computation," the final workshop of the SFB340, Bad Teinach, October 10-13, 1999. Thanks to Tom Cornell, Henk Harkema, Ed Keenan, Hilda Koopman, Aravind Joshi, Seth Kulick, Jens Michaelis, and Uwe Mönnich for discussions and ideas which inspired this project.

(2) [show me] → merge P
to [show me] → P attracts DP
me$_i$ to [show t_i] → merge W, W attracts P
to$_j$+W me$_i$ t_j [show t_i] → VP raises to spec,WP
[show t_i]$_k$ to$_j$+W me$_i$ t_j t_k

We can depict this derivation with a tree of the usual kind, but to highlight the chains we have used arrows to point to each landing site rather than coindexing:[1]

(2t)



We can see from the downward arrow that in this structure there are traces which are not c-commanded by their antecedents, even though each phrasal movement is to a c-commanding position.[2]

Kayne's main point about this derivation, though, is that it ties VO structure to PO structure quite directly. If the object has to move out of the VP to end up in clause-final, VO position, and if P needs to be licensed with

[1] This graphical depiction is abbreviated slightly to highlight just the properties relevant to the present discussion. In particular, we leave out the V category (and subcategorial features) of *show*, the DP category of *me*, etc.

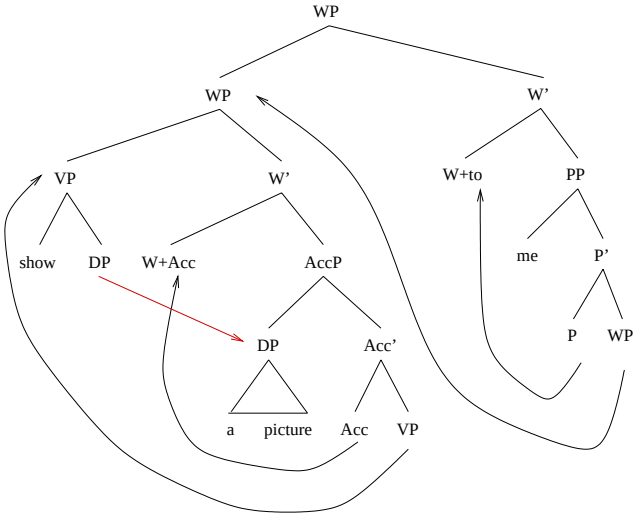
[2] The fact that movement alters command relations can be obscured by certain perspectives. Consider, for example, Kracht's (1998) formalization of movement. Given any labeled ordered tree $T = \langle Nodes, root, <, Precedence, label\rangle$, Kracht says that binary relation R on $Nodes$ is a *command* relation iff (i) $dom(R) = Nodes$, (ii) for any $x \in Nodes$, $\{n|\ xRn\}$ forms a constituent properly containing x, and (iii) if $x \leq y$ then $\{n|\ xRn\} \subseteq \{n|\ yRn\}$. Then any command relation R is preserved by deletion of constituent y of tree T in the following sense: if xRy in T and $x, y \notin y$ then xRy in the tree that results from the deletion of y. Similarly for constituent insertion (which Kracht calls "tagging"). Obviously, if movement is defined to be the insertion of a copy of a subtree t followed by deletion of the original subtree, command relations are preserved, since none of the nodes of the original tree have been rearranged. However, if movement is defined in such a way that the original nodes appear in the result – so that they are moved rather than copied (and a "trace node" may be inserted in the original position of the root of the moved subtree) – then, of course, command relations are not preserved, and in particular, the command relations between traces and their antecedents may be disrupted.

W, then getting OP in VO languages is impossible if there are no alternative clausal structures or further movements of (some constituent containing) the object.

To get a direct object as well as the prepositional object, Kayne (1994, p69) indicates that he prefers some sort of small clause analysis in which each object is introduced in an embedded verbal projection.[3] This idea might be combined with the previous proposal to yield an analysis with WPs and VPs nesting and moving roughly like this:

(3) [show a picture] → DP raises to Acc
a picture$_i$ [show t_i] → merge W, VP raises to spec,WP
[show t_i]$_k$ W [a picture]$_i$ t_j t_k → merge V, DP
me V [[show t_i]$_k$ W [a picture]$_i$ t_j t_k] → merge P
to [me V [[show t_i]$_k$ W [a picture]$_i$ t_j t_k]] → P attracts DP
me$_l$ to [t_l V [[show t_i]$_k$ W [a picture]$_i$ t_j t_k]] → merge W, W attracts P
to$_m$+W me$_l$ t_m [t_l V [[show t_i]$_k$ W [a picture]$_i$ t_j t_k]] → WP to spec,WP
[t_l V [[show t_i]$_k$ W [a picture]$_i$ t_j t_k]]$_n$ to$_m$+W me$_l$ t_m t_n

(3t)



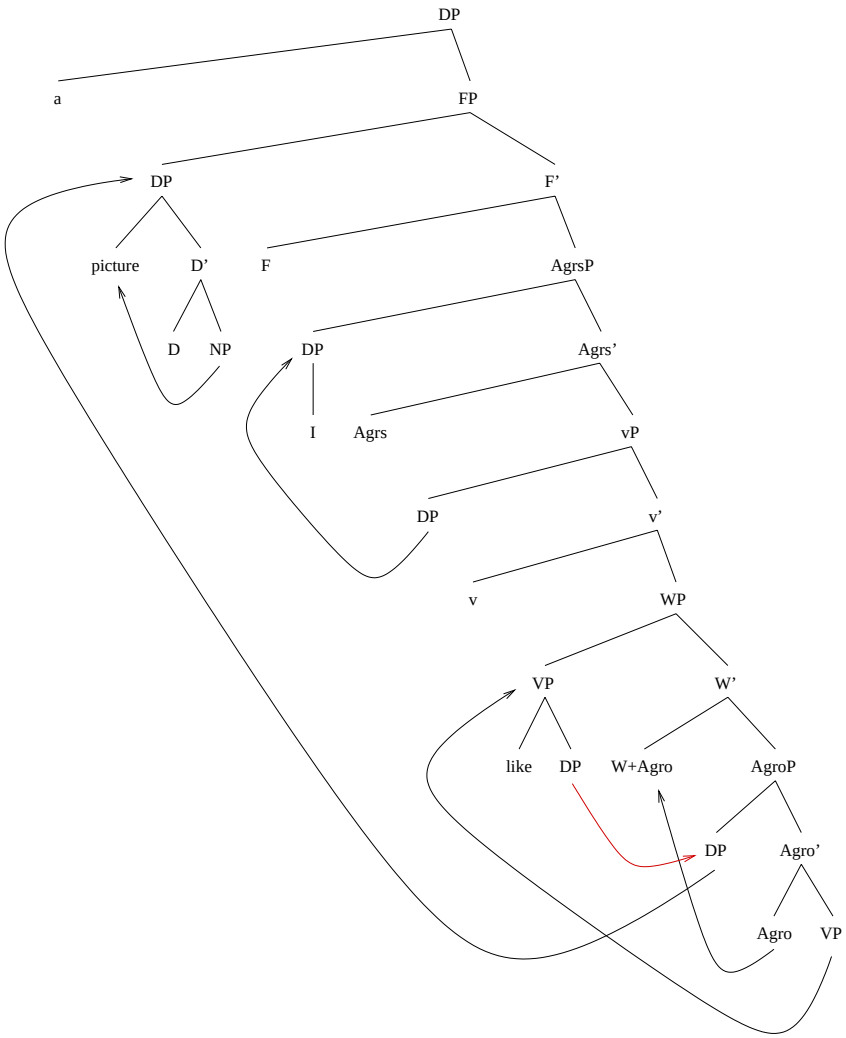
To get relative clauses without right adjunction, Kayne (1994, §8) proposes an analysis roughly like the following, where *D* is a silent relative pronoun:

(4) [D picture] → raise NP
[picture$_i$ D t_i] → merge V, subject, etc
I like [picture$_i$ D t_i] → attract DP to spec,CP
[picture$_i$ D t_i]$_j$ I like t_j → merge higher D
a [picture$_i$ D t_i]$_j$ I like t_j

[3] Cf., e.g., the multiply-headed analyses of Sportiche (1999), Kural (1996), Harley (1995), Pesetsky (1995), Larson (1988).

The graph that depicts this analysis again reveals traces that are not c-commanded by their antecedents.

(4t)



Similar analyses in which the head of the relative is promoted from inside the clause have been proposed for independent reasons by Schacter (1985), Vergnaud (1982) and others.

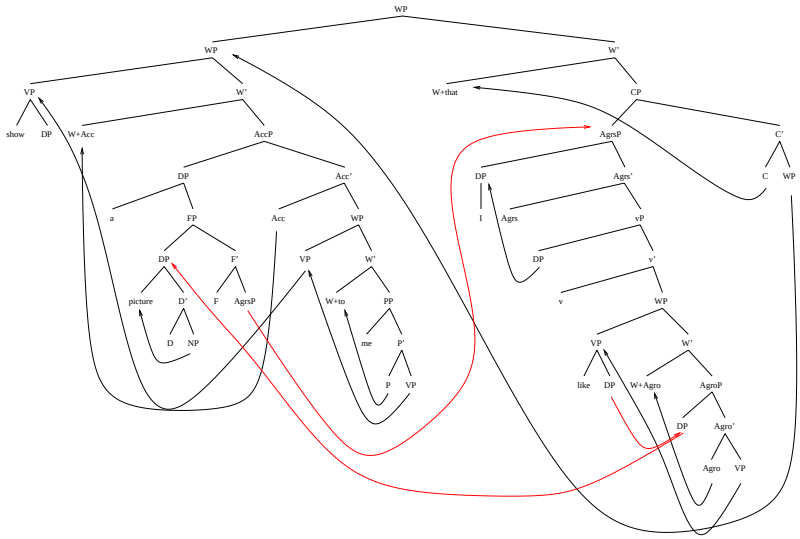
Finally, in an attempt to account for the fact that languages are OV if they are complementizer-final, Kayne (1999) recently extends this idea about relative clauses and prepositions to the so-called "extraposed relatives" as follows:

(5) showed [a picture I like] to me → merge C
that showed [a picture I like] to me → attract relative clause to CP
[I like]$_i$ that showed [a picture t_i] to me → merge W, attract C

that$_j$+W [I like]$_i$ t_j showed [a picture t_i] to me → attract VP to WP
[showed [a picture t_i] to me]$_k$ that$_j$+W [I like]$_i$ t_j t_k

Graphically:

(5t)



The strategy here is the same as the earlier one: to account for the connection between VO and C positions, rather than allowing merge operations which could connect constituents in any order, the placement of complementizer, verb and object are determined together. On this analysis, it is easy to see how the relative position of the complementizer would be altered if the O moved out of the VP before a constituent containing the O and VP moved to WP.

These analyses have some surprising features that are shared by many other recent proposals:[4]

(6) many constituents move

(7) apparent rightward movement might not be

(8) even though phrasal movement is always to a c-commanding position,
traces can end up not c-commanded by their antecedents

(9) in "remnant movement" analyses, a small change in grammar can change a nested to a crossed dependency (Koopman and Szabolcsi, 2000)

Taken together, these ideas suggest a perspective on language which can be expressed in the following familiar form, but with a new emphasis: *human*

[4]See especially Koopman and Szabolcsi (2000), Sportiche (1999), Chomsky (1998), Chomsky (1995).

languages are chain-based. The new idea is that, typically, each pronounced constituent plays a role in licensing a number of structural positions, positions that are not generally regarded as "local" (i.e. in the same elementary subtree or the same maximal projection in the phrase structure skeleton). This is not the initial perspective that one gets in developing a grammar for English along traditional lines; it is natural to begin with a description in which the effects of words and constituents are largely local, tacking on movement to shuffle things around occasionally. (Other theories handle the "shuffling" with path feature percolation rules, path equations, and many other mechanisms.) But we want to allow that the determinants of lexical and phrasal properties are spread throughout clausal structure, and that the configurations of antecedent-trace relations can be quite complex. Rather than a basically hierarchical structure with a few extra things, we are pushed to a perspective where many structures are distributed.

Since moved constituents and their traces do not stand in their usual command configurations in these analyses, traditional slash-passing approaches to movement relations do not suffice to handle these analyses. Traditional slash-passing analyses have been formalized with linear indexed grammars (LIGs), but a simple formalization of the "minimalist" analyses above shows that they are capable of defining string languages that are beyond the expressive power of LIGs (Stabler 1997, 1999; Michaelis 1998). In this sense, these recent proposals really do provide a new perspective on movement dependencies, and they call for a different processing strategy, one that can appropriately assemble chains of related constituents in configurations that earlier transformational grammars forbade. This paper will quickly sketch this simple formalization of minimalist ideas, compare this formalism to similar "multi-component" proposals from the tree adjoining grammar (TAG) tradition, and adapt a recognition algorithm from the TAG tradition for the minimalist grammars.

2 A fragment of minimalist grammar

The analyses sketched above can be captured in the grammar formalism presented in Stabler (1997). Unlike that earlier work, though, we present those grammars in a chain-based notational variant, adapting the insight from Cornell (1998) that this kind of formulation is exactly equivalent to the derivational presentation used earlier.

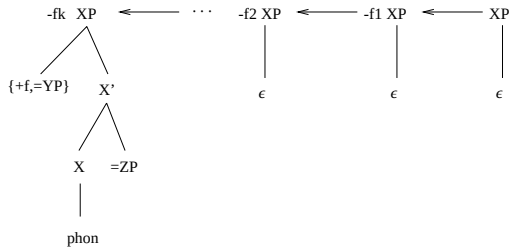
As in Keenan and Stabler (1996), we will assume that the grammar is composed of a finite lexicon and a finite set of structure building rules, $merge$ and $move$, and that the language is all the structures that can be built from the lexicon with these rules. That is, the language is the closure of the lexicon under the structure building rules.

Setting aside head movement for the moment, lexical items are sequences of trees, which we will call *chains*. The nodes of the trees are labeled with finite sequences of features from the following sets:

1. categories: X, X', XP (for X$\in \{N, V, A, P, \ldots\}$)
2. selector features: =XP (for X$\in \{N, V, A, P, \ldots\}$)
3. movement triggers: +f (for f$\in \{case, wh, \ldots\}$)
4. movement requirements: -f (for f$\in \{case, wh, \ldots\}$)
5. nonsyntactic features, such as the phonetic properties of lexical items, will be indicated by a standard orthographic representation of the words. So for example, we will consider the nonsyntactic, "terminal" vocabulary

$$V^{\epsilon} = \{\epsilon, \texttt{you}, \texttt{believe}, \texttt{it}, \ldots\}.$$

The most complex chains in the lexicon have the following form:

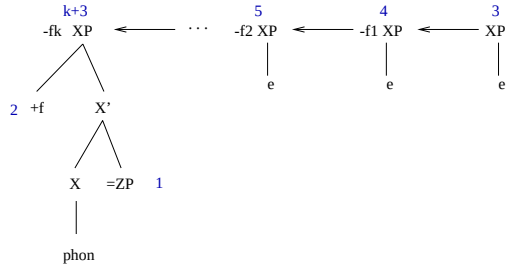


This chain is a sequence of k trees or "links," where k can be 1 or more. The order among these trees is represented by the arrows. I have drawn the first tree rightmost, in recognition of the fact that movements are leftward (though the possibility of disturbing this leftward relation by later remnant movements will be allowed for). The root of each tree in a chain, other than the first, is labeled by a sequence `-f XP` for some requirement `f` and some category `X`. The fact that all but the last tree in every chain is empty represents the simple assumption that all movement is overt. It is not difficult to relax this assumption to allow for various kinds of covert movement and feature movement generally, but we will leave this aside. A constituent that does not move at all is represented by a chain of length one: that is, a single X-bar tree, with a root that is simply labeled `XP`. The specifier position in the last link of this chain is labeled {`+f,=YP`} to indicate that this position can be labeled either with a movement trigger `+f` or with a second selection requirement `=YP`. I will call the positions labeled with elements of V^{ϵ} the **lexical leaves** of the tree.

There are two structure building operations which are called *merge* and *move*, since they are loosely inspired by the operations with those names in Chomsky (1995). To define these operations, we define the accessible node in each chain with the following order:

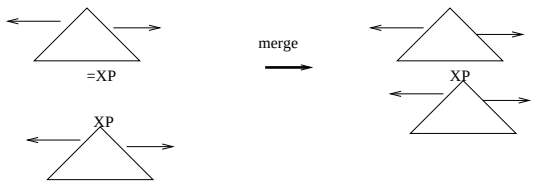
comp < spec < root of 1st link < …< root of last link

The structure building rules must apply to the nodes of a chain in this order. In our graphical representation of the form of each lexical item, we can number the nodes in order of accessibility as follows:

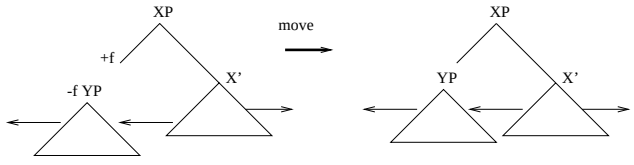


So if a chain has an X-bar tree with a complement labeled =XP, then that node is most accessible, and must be acted upon before any other node in the chain. If the chain either has no complement or else a complement labeled XP (i.e. a complement that has already been filled), and if it has a specifier labeled +f, then this specifier must be acted on first. After the complement and specifier positions have been filled, only then can the first link be attached into another structure, then the second and so on to the last.

The structure building rule *merge* is a binary function: it applies to a pair of structures. When two chains are merged, the result is a collection of trees that are "totally connected" in the sense that every element of the collection is related to every other by dominance, chain-precedence, or their inverses. Let's call such a structure a connected forest. Then we can say that *merge* applies to two connected forests to yield another one: $merge(F_1, F_2) = F$ where the first forest F_1 has an accessible feature =XP and F_2 has an accessible link with root XP. The result F then is obtained by removing this =XP and replacing it by XP and all of the structure F_2 to which it is connected. Schematically:



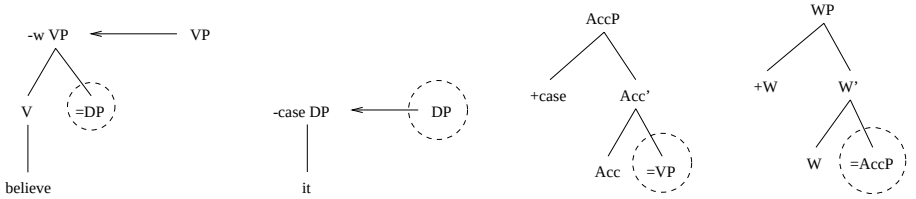
The other structure building rule, *move*, is unary, applying to one forest which has an accessible node labeled +f (for some f) and an accessible link with root labeled -f XP (for some X). The result $move(F_1) = F$ is the structure that results from first deleting the -f from the label -f XP, and then substituting that link for the node labeled +f. Schematically:



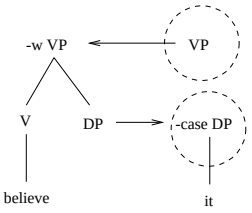
We impose a simple version of the "shortest move condition" (SMC) by prohibiting structures that have two heads with the same requirement `-f` as their first feature. Notice that no lexical item can violate this, given our definition of the form of lexical items. But without this additional restriction on the domains of the structure building rules, they would have been able to form structures in which two requirements `-f` would both be "competing" for the first movement trigger `+f` that appeared in the structure. With this simple SMC, which can be regarded as a restriction on the domains of the structure building rules, there will always be at most one `-f` substructure that can be attracted to a `+f` movement trigger.

2.1 Example grammar 1

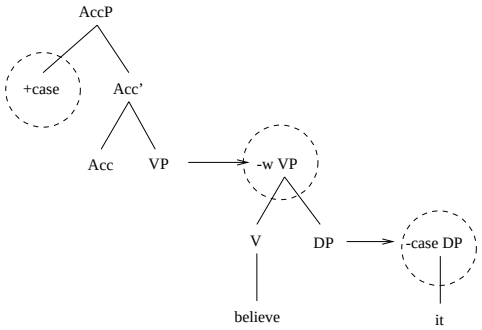
Consider the lexicon containing just these four lexical items, in which I have circled the accessible nodes (and I have omitted the explicit indications of the empty yields ϵ of Acc, W, and the first links of the VP and DP, to enhance readability):



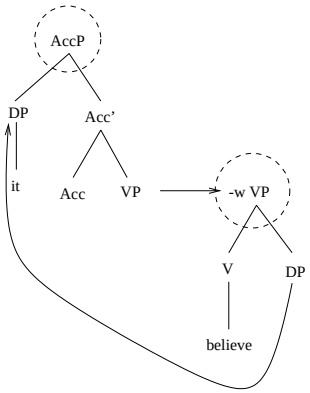
The structure building rules can be applied to derive some new structures from these. For example, the verb selects a determiner, and we have a lexical determiner, and these nodes are accessible, so merge applies to yield the value:



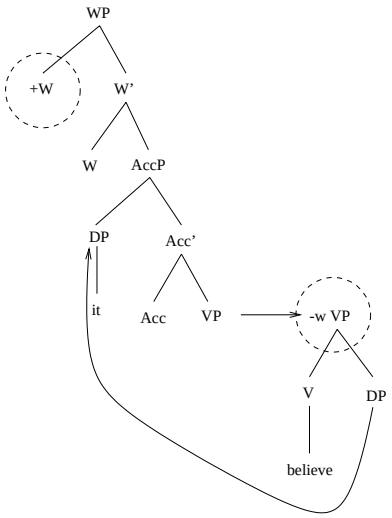
Notice that the chain-precedence relations are preserved in the result, and there is still one accessible node in each chain, so that there are now two accessible nodes. Movement cannot apply because there is no `+case` head, so the only rule we can apply is merge. Since the lexical AccP selects VP, and we have an accessible VP node here, we can merge to obtain:



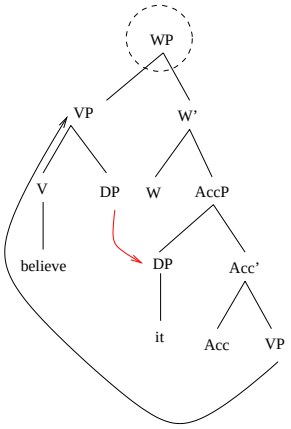
At this point, merge cannot apply, since no leaf has any selection features `=XP` any more, and no root is accessible to be selected. However, move can apply to attach the `-case` link to the `+case` specifier, yielding the following result:



Now we can merge this structure with the lexical WP, to obtain



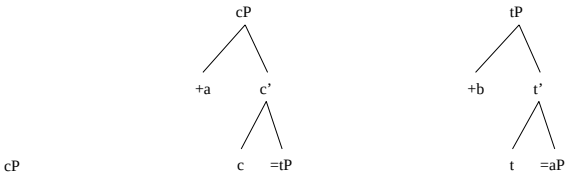
Merge cannot apply to this structure, but move can, yielding

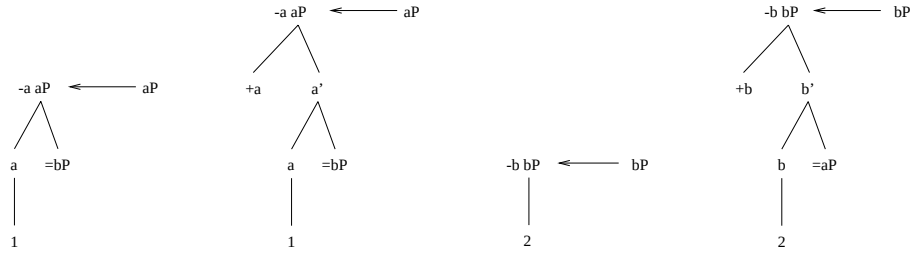


Notice that every step leaves chain-precedence relations (indicated by the arrows) unchanged, but the final step makes the chain-precedence relation between the links of the DP chain relate positions that do not stand in a c-command relation. After the final step, the structure has no unattached links, and no outstanding requirements `=XP, +f, -f`, so it is a "complete" WP. Usually we assume that the "start category" of our grammars is CP – a clause – but if WP were the start category of this grammar, then this would count as a successful derivation of the string `believe it`.

2.2 Example grammar 2

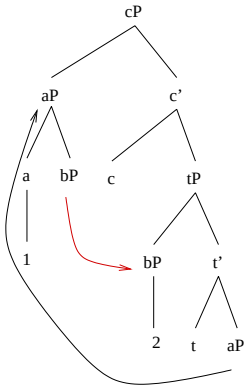
The structure derived in the previous section has one moved constituent that does not c-command its trace, but we saw in the introduction that there can be many of these. The following simple grammar shows that we can derive arbitrarily many non-c-commanding antecedent trace relations very easily. This grammar has the pronounced vocabulary $\{1, 2\}$, and with start category CP, it generates the string language $\{1^n 2^n \mid n \geq 0\}$ from these 7 lexical items (the first of which is just an empty CP, to generate the empty string):



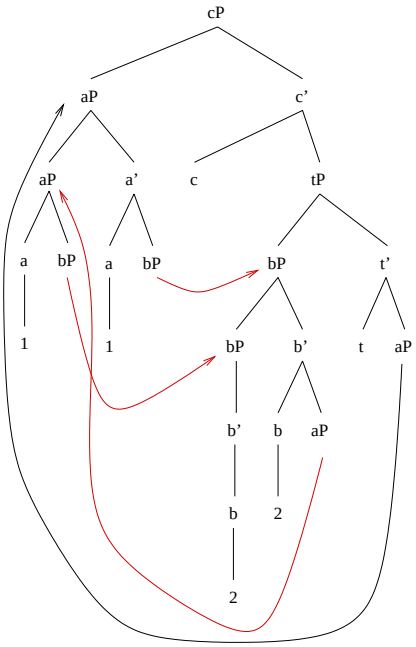


In the CPs that can be derived from this grammar (the ones with no unconnected links and no outstanding requirements), the 1's appear in aPs. The lexical requirements are such that whenever an aP is introduced into the derivation, the aP embedded under it (if any) moves to its specifier. The 2's appear in bPs and they similarly attract any lower bPs to specifier position. Finally, the bPs all move to the specifier of tP, and the aPs move to the specifier of cP to complete the "clause."

This grammar trivially derives the empty string, since it is in the lexicon. This grammar derives the string 12 with two movements, to yield one non-c-commanding antecedent-trace relation:



And it derives the string 1122 with four movements, to yield three non-c-commanding antecedent-trace relations:



As the derived strings get longer, the number of non-c-commanding antecedent-trace relations increases without bound.

It is a familiar fact that a simple context free grammar can derive 1^n2^n. The grammar provided here, though, extends easily to grammars in which parts of the aPs and bPs are left stranded to the right of the 1^n2^n cluster, yielding a range of structures that has been used in the analysis of Hungarian verbal complexes (Koopman and Szabolcsi, 2000). Furthermore, this movement grammar for 1^n2^n extends easily to the non-context free language $1^n2^n3^n$ and also to the non-tree-adjoining language $1^n2^n3^n4^n5^n$, as was pointed out in Stabler (1999).

3 Previous work

Although the conventional depictions of MG structures look rather bulky, we can specify the structures by listing the rules that applied to derive them, and then from this sequence we can eliminate all but a specification of the choices made in the course of the derivation, to yield a very succinct representation (Stabler, 1999).

The way that these MGs factor dependencies is clearly unlike TAGs, and they are only slightly more similar to set-local multi-component TAGS (MC-TAGs) (Joshi, Levy, and Takahashi, 1975; Weir, 1988), where the components to be combined are sets of trees. D-tree grammars (DTGs) and some related proposals come a little closer, using ordered sets with elements whose attachment sites must respect the ordering (Rambow, Vijay-Shanker, and

Weir, 1995a; Rambow, Vijay-Shanker, and Weir, 1995b; Frank, Kulick, and Vijay-Shanker, 1999; Kulick, 1998).

Michaelis (1998) showed that, given any MG, we can construct a MCFG that generates the same language, so we know MLs $\subseteq$ MCFLs. Furthermore, the MCFLs are exactly the languages generated by linear context free rewrite systems (Seki et al., 1991; Weir, 1988), and these systems are also expressively equivalent to Boullier's (1998) simple positive range concatenation grammars. However, the construction that Michaelis uses to establish his result is complex, and so it has been difficult to approach questions like the following:

Open 1: Can MCFG recognition algorithms be adapted to yield a natural MG parser?

Open 2: Certain elaborations of MGs are needed (successive cyclic movements, multiple extractions, etc.). Is the set of languages definable with these elaborations still contained in the MCFLs?

Open 3: MCFLs $\subseteq$ MLs ?

We consider the first question here. Hopefully, our approach to this first question may be a step toward addressing the other questions as well.

4 Recognition

It is clear that simple trace passing recognition methods of the familiar kind cannot recover derivations in the MG framework. This is informally suggested by the structure of chains noted in the introduction, where we saw that a trace can appear to the left and higher than its antecedent. Taking linear indexed grammars as a kind of formalization of simple trace passing grammars, the suspicion that these are not adequate is clinched by the formal results showing that MGs can define languages which are simply beyond the expressive power of linear indexed grammars (Stabler 1997, 1999; Michaelis 1998). However, there is work on recognition in the TAG tradition that provides exactly what we need: recognition methods for linear context free rewrite systems, multiple context free grammars, and related formalisms are easily adapted to MGs (Boullier 1998, 1999; Groenink 1997; Rambow, Vijay-Shanker and Weir 1995a; Seki et al. 1991).Rambow, Vijay-Shanker, and Weir (1995b) These methods can be regarded as being based on Pollard's (1984) insight that the expressive power of context-free-like grammars can be enhanced by marking one or more positions in a string where further insertions can take place, and that this is indeed the primary roles of the tree structures in TAGs and in transformational grammars. The recognition methods for tree grammars need only represent the positions of the substrings of each constituent. Recent work elegantly separates the concatenation conditions that define the linear order of pronounced elements from the other structural conditions imposed by the grammar. From

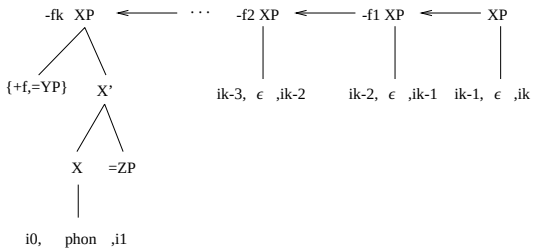
this perspective, it is no surprise that the stage is set for a grammars like the ones defined above in which discontinuous constituents are the rule.

In fact, a simple CKY-like parsing algorithm suffices, one that stays very close to the operations of the grammar itself. Notice that each link has exactly one (possibly empty) terminal string "anchor," and so a chain will be introduced into the parse wherever all of its links can be anchored. Then, the lexical items can be combined in all the ways allowed by the grammar, so long as the adjacency relations are respected within any given link. That is, linked trees need not be adjacent, but only left-adjacent structures can be attached to specifier position, and only right-adjacent items can be attached in complement position.

To represent the input to our recognition method, we can number the positions in the input string from 0 to the length of the string, as usual, or we can regard the position numbers as states in an arbitrary finite automaton.[5]

We will specify the CKY-like method in a deductive style familiar from Shieber, Schabes, and Pereira (1993), Sikkel and Nijholt (1997) and similar work. The chart computation is represented as the closure of a set of lexical axioms under some inference rules:

1. **Lexical axioms:** For each chain $(T_1, T_2, \ldots T_n)$ in the lexicon with lexical leaves $(w_1, w_2, \ldots w_n)$ (where w may be the empty string), if each w_i $(1 \leq i \leq n)$ occurs between positions (or states) i_{i-1} and i_i, then we have as an axiom the positioned "lexical chain" $(i_0, T_1, i_1), (i_1, T_2, i_2), \ldots (i_{n-1}, T_n, i_n)$. The form of a positioned chain then can be depicted as follows:



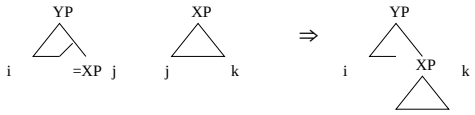
   The accessibility relations of the nodes of each chain, defined earlier, is extended to these positioned chains.

2. **Inference rule: merge complement.**

   Given expressions E_1, E_2, where the first feature on an accessible node n_1 in a complement of a tree (i, T_1, j) in E_1 is $= f$ for some feature f, and where the first feature of an accessible node n_2 of a tree (j, T_2, k)

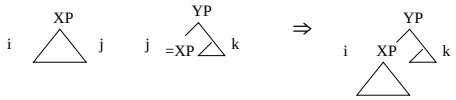
[5] Parsers that can take regular sets as inputs like this have a big advantage, since regular characterizations of phonological and prosodic structure are often used. See for example Billot and Lang (1989). This framework has the potential for capturing the sometimes rather subtle interactions between syntactic and prosodic structure (Hayes, 1989), and possibly even the the OT-based models of morphology and phonology found in Eisner (1997), Albro (1997) and elsewhere.

in E_2 is f, merge can apply to attach these two expressions, yielding an expression E_3 which is the result of deleting n_1 and attaching the parent of n_1 to n_2, in a tree (i, T, k):



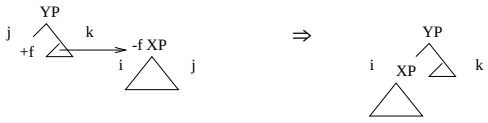
The positions of T_1, T_2 in their respective chains is unaltered by this merge operation. So, once this operation has applied, the next accessible node (if any) in each chain is determined in the way discussed earlier. So the resulting expression $E3$ may well have more than one accessible node.

3. **Inference rule: merge specifier.**
   Given expressions E_1, E_2, where the first feature on an accessible node n_1 in a specifier of a tree (j, T_1, k) of E_1 is $= f$ for some feature f, and where the first feature of an accessible node n_2 of a tree (i, T_2, j) in E_2 is f, merge can apply to attach these two expressions, yielding an expression E_3 which is the result of deleting n_1 and attaching the parent of n_1 to n_2, in a tree (i, T, k):



   The positions of T_1, T_2 in their respective chains is unaltered by this merge operation. So, once this operation has applied, the next accessible node (if any) in each chain is determined in the way discussed earlier. So the resulting expression $E3$ may well have more than one accessible node.

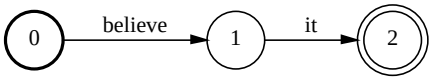
4. **Inference rule: move.**
   Given an expression E_1, where the first feature on an accessible node n_1 in a specifier of a tree (j, T_1, k) in E_1 is $+f$ for some feature f, and where the first feature of exactly one accessible node n_2 of another tree (i, T_2, j) in E_1 is f, move can apply to attach these two trees, yielding an expression E_3 which is the result of deleting n_1 and attaching the parent of n_1 to n_2, in a tree (i, T, k):



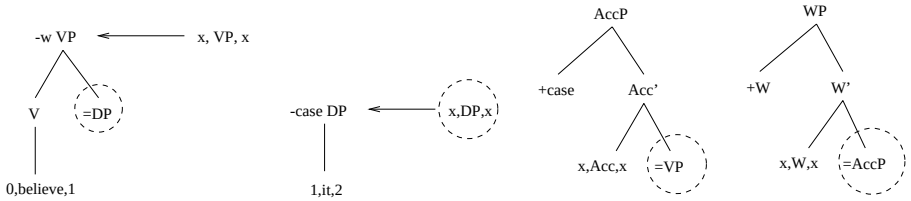
   The arrow between the two involved subtrees here is intended to indicate just that, since these two trees are part of the same expression, they must be connected somehow by chain relations.

4.1 Example 1

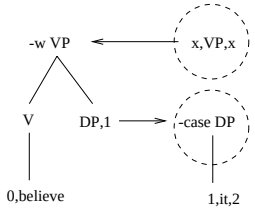
Let's step through the recognition of the WP *believe it* using the grammar presented in section 2.1, above. We assume that the input is given as a finite state automaton – in this case a trivial one:



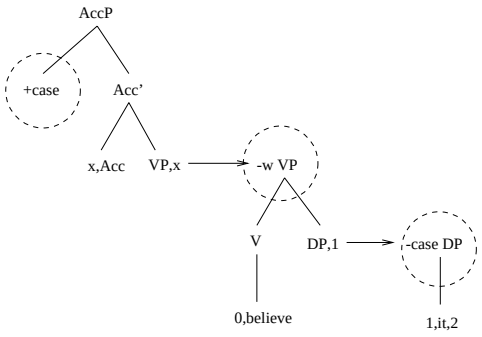
The lexical axioms are the following. For all $x \in \{0, 1, 2\}$ we have these positioned chains (in which I have not drawn the ϵ yields of Acc, W and the first links of the VP and DP, but simply labeled these trees as positioned between any positions x and x):

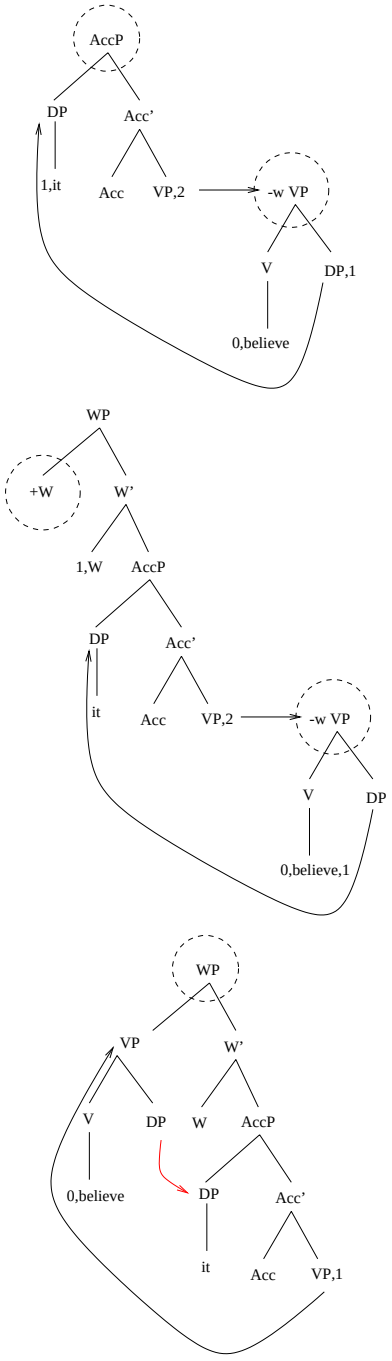


The recognition steps then recapitulate the derivation steps shown earlier, but now with string positions calculated according to the CKY rules given above. To begin, the first two lexical items can merge to yield, for $x \in \{0, 1, 2\}$:



Notice that, in the result of this first step, I have indicated the string position of the tree from 0 to 1 by placing the 0 before the first lexical leaf and the 1 after the last lexical leaf. Following this labeling convention, we have the following steps:





We obtain the completed singleton chain that has no outstanding features except the root category, spanning the whole input from position 0 to position 2.

4.2 Implementation

This CKY-like recognition strategy, presented as an method for assembling "positioned chains," is intended to be relatively intuitive and natural for linguists accustomed to reasoning about trees.[6] But to define the recognition calculation, we can reduce these tree structures to the bare essentials. To do this, we represent each chain just by the sequence of syntactic features in order of accessibility (and we can simply write d for DP, v for VP, etc.). We can omit the positions of all but the lexical leaves of the last tree in each chain, since non-final trees in a chain are always empty and can take any position. And we distinguish chains with unfilled complements from chains with filled complements by the types : and ::, respectively. Then, representing the possible feature sequences with a regular expression, each chain positioned in a finite state machine with states $N = \{0, 1, \ldots, n\}$ can be represented by an element of

$$(N \times N) \times \{:,::\} \times ((\text{selectors}(\text{selectors} \cup \text{triggers}))\text{categories}(\text{requirements}^*)).$$

And each expression can be represented by a nonempty sequence of chains. So, for example, the lexical axioms of the example calculation in the previous section are these, for all $x \in \{0, 1, 2\}$:

1. (0,1)::=d v -w
2. (1,2)::d -case
3. (x,x)::=v +case acc
4. (x,x)::=acc +w w

[6] Drawing from the recent work of (Harkema, 2000), the implementation sketched here is improved from the one originally presented at Bad Teinach.

We define the CKY inference rules for any positions $a, b, c, d \in N$, any type $\cdot \in \{:, ::\}$, any $f \in$ categories, any $+g \in$ triggers, any (possibly empty) sequence of features γ, any nonempty sequence of features δ, and any chains $\alpha_1, \ldots, \alpha_k, \iota_1, \ldots, \iota_l$ $(0 \leq k, l)$:

$$\frac{(a,b) :: =f\,\gamma \qquad (b,c) \cdot f, \alpha_1, \ldots, \alpha_k}{(a,c) : \gamma, \alpha_1, \ldots, \alpha_k}\ \text{merge1}$$

$$\frac{(b,c) : =f\,\gamma, \alpha_1, \ldots, \alpha_k \qquad (a,b) \cdot f, \iota_1, \ldots, \iota_l}{(a,c) : \gamma, \alpha_1, \ldots, \alpha_k, \iota_1, \ldots, \iota_l}\ \text{merge2}$$

$$\frac{(a,b) \cdot =f\,\gamma, \alpha_1, \ldots, \alpha_k \qquad (c,d) \cdot f\,\delta, \iota_1, \ldots, \iota_l}{(a,b) : \gamma, \alpha_1, \ldots, \alpha_k, (c,d) : \delta, \iota_1, \ldots, \iota_l}\ \text{merge3}$$

$$\frac{(b,c) : +g\,\gamma, \alpha_1, \ldots, \alpha_{i-1}, (a,b) : -g, \alpha_{i+1}, \ldots, \alpha_k}{(a,c) : \gamma, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k}\ \text{move1}$$

$$\frac{(a,b) : +g\,\gamma, \alpha_1, \ldots, \alpha_{i-1}, (c,d) : -g\,\delta, \alpha_{i+1}, \ldots, \alpha_k}{(a,b) : \gamma, \alpha_1, \ldots, \alpha_{i-1}, (c,d) : \delta, \alpha_{i+1}, \ldots, \alpha_k}\ \text{move2}$$

To enforce the "shortest move constraint" (SMC) in the move operations, we require that none of $\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k$ has $-f$ as its first feature.

With these rules, the recognition steps depicted in the previous section are represented as follows:

```
1.  (0,1)::=d v -w                              lexical
2.  (1,2)::d -case                              lexical
3.  (x,x)::=v +case acc                         lexical
4.  (x,x)::=acc +w w                            lexical
5.  (0,1):v -w,(1,2):-case                      merge3(1,2)
6.  (x,x):+case acc,(0,1):-w,(1,2):-case        merge3(3,5)
7.  (1,2):acc,(0,1):-w                          move1(6)
8.  (1,2):+w w,(0,1):-w                         merge1(4,7)
9.  (0,2):w                                     move1(8)
```

This method is not ideal for linguists who want to visualize each step, but accomplishes exactly the same thing as the chain assembly rules presented above. This method is shown to be sound, complete and efficient in Harkema (2000). The parsing rules given here can be dropped into any "chart-based" method for computing closures, such as the one presented in Shieber, Schabes, and Pereira (1993), to obtain running version of this method. An alternative, constraint propagation implementation of the same method is provided in (Morawietz, 1999).

Using the basic strategy from Billot and Lang (1989) and others, we can extend this method to build a "packed forest" representation of derivation trees, from which a tree can be extracted efficiently.

5 Conclusions

We have provided a recognition method for a simple formalization of some basic ideas from recent transformational syntax, a kind of "minimalist grammar" (MG). These grammars allow "remnant movements" of the kind found in many recent proposals, and, as has been observed before, they consequently can define languages that are beyond the expressive power of simple tree adjoining grammars. However, the definable languages are in the class of "multiple context free languages" and "multicomponent tree adjoining languages," and consequently it is not difficult to adapt recognition methods for those grammars to MGs. This step brings our understanding of MGs closer to these other grammars. The convergence of different traditions on these "mildly context sensitive" grammars provides some further support for the idea that we these grammars may have roughly the right expressive power for linguistic theory, though there are some challenges to this claim (Michaelis and Kracht, 1997).

References

Albro, Daniel M. 1997. Evaluation, implementation, and extension of primitive optimality theory. M.A. thesis, UCLA.

Albro, Daniel M. 1998. Three formal extensions to primitive optimality theory. 1998 ACL Meeting.

Billot, Sylvie and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of the 1989 Meeting of the Association for Computational Linguistics*.

Boullier, Pierre. 1998. Proposal for a natural language processing syntactic backbone. Technical Report 3242, Projet Atoll, INRIA, Rocquencourt.

Boullier, Pierre. 1999. On TAG and multicomponent TAG parsing. Technical Report 3668, Projet Atoll, INRIA, Rocquencourt.

Chomsky, Noam. 1995. *The Minimalist Program*. MIT Press, Cambridge, Massachusetts.

Chomsky, Noam. 1998. Minimalist inquiries: the framework. MIT. Forthcoming.

Cornell, Thomas L. 1998. Derivational and representational views of minimalist transformational grammar. In *Logical Aspects of Computational Linguistics 2*. Springer-Verlag, NY. Forthcoming.

Eisner, Jason. 1997a. Decomposing FootForm: Primitive constraints in OT. In *Proceedings of Student Conference in Linguistics, SCLI VIII*, MIT Working Papers in Linguistics.

Eisner, Jason. 1997b. Efficient generation in Primitive Optimality Theory. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*.

Eisner, Jason. 1997c. What constraints should OT allow? Presented at the Annual Meeting of the Linguistic Society of America, Chicago. Available at http://ruccs.rutgers.edu/roa.html, January.

Frank, Robert, Seth Kulick, and K. Vijay-Shanker. 1999. C-command and extraction in tree adjoining grammar. Johns Hopkins University, University of Pennsylvania, University of Delaware.

Groenink, Annius. 1997. *Surface without structure: Word order and tractability issues in natural language processing*. Ph.D. thesis, Utrecht University.

Harkema, Henk. 2000. A recognizer for minimalist grammars. In *Sixth International Workshop on Parsing Technologies, IWPT'2000*.

Harley, Heidi. 1995. *Subjects, Events and Licensing*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Hayes, Bruce. 1989. The prosodic hierarchy in meter. In P. Kiparsky and G. Youmans, editors, *Rhythm and Meter*. Academic, NY.

Joshi, Aravind. 1985. How much context-sensitivity is necessary for characterizing structural descriptions. In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Processing: Theoretical, Computational and Psychological Perspectives*. Cambridge University Press, NY, pages 206–250.

Joshi, Aravind K., Leon S. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10:136–163.

Joshi, Aravind K., K. Vijay-Shanker, and David Weir. 1991. The convergence of mildly context sensitive grammar formalisms. In Peter Sells, Stuart Shieber, and Thomas Wasow, editors, *Foundational Issues in Natural Language Processing*. MIT Press, Cambridge, Massachusetts, pages 31–81.

Kallmeyer, Laura. 1999. *Tree Description Grammars and Underspecified Representations*. Ph.D. thesis, Universität Tübingen.

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

Kayne, Richard. 1999. A note on prepositions and complementizers. In *A Celebration*. MIT Press, Cambridge, Massachusetts. Currently available at http://mitpress.mit.edu/chomskydisc/Kayne.html.

Kayne, Richard and Jean-Yves Pollock. 1999. New thoughts on stylistic inversion. Manuscript, New York University and CNRS, Lyon.

Keenan, Edward L. and Edward P. Stabler. 1996. Abstract syntax. In Anne-Marie DiSciullo, editor, *Configurations: Essays on Structure and Interpretation*, pages 329–344, Somerville, Massachusetts. Cascadilla Press.

Koopman, Hilda and Anna Szabolcsi. 2000. *Verbal Complexes*. MIT Press.

Kracht, Marcus. 1998. Adjunction structures and syntactic domains. In Hans-Peter Kolb and Uwe Mönnich, editors, *The Mathematics of Syntactic Structure: Trees and their Logics*. Mouton-de Gruyter, Berlin.

Kulick, Seth. 1998. Constrained non-locality in syntax: Long-distance dependencies in tree adjoining grammar. Ph.D. Dissertation Proposal.

Kural, Murat. 1996. *Verb incorporation and elementary predicates*. Ph.D. thesis, University of California, Los Angeles.

Lang, Bernard. 1994. Recognition can be harder than parsing. *Computational Intelligence*, 10.

Larson, Richard K. 1988. On the double object construction. *Linguistic Inquiry*, 19:335–391.

Michaelis, Jens. 1998. Derivational minimalism is mildly context-sensitive. In *Proceedings, Logical Aspects of Computational Linguistics, LACL'98*, Grenoble. Available at http://www.ling.uni-potsdam.de/ michael/papers.html.

Michaelis, Jens and Marcus Kracht. 1997. Semilinearity as a syntactic invariant. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics*, pages 37–40, NY. Springer-Verlag (Lecture Notes in Computer Science 1328).

Mönnich, Uwe. 1997. Adjunction as substitution. In *Formal Grammar '97, Proceedings of the Conference*.

Morawietz, Frank. 1999. Parsing as constraint propagation. Technical report, University of Tübingen.

Pesetsky, David. 1995. *Zero Syntax: Experiencers and Cascades*. MIT Press, Cambridge, Massachusetts.

Pollard, Carl. 1984. *Generalized phrase structure grammars, head grammars and natural language*. Ph.D. thesis, Standford University.

Rambow, Owen, K. Vijay-Shanker, and David Weir. 1995a. D-tree grammars. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*.

Rambow, Owen, K. Vijay-Shanker, and David Weir. 1995b. Parsing D-tree grammars. In *Proceedings of the International Workshop on Parsing Technologies*.

Schacter, Paul. 1985. Focus and relativization. *Language*, 61:523–568.

Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1993. Principles and implementation of deductive parsing. Technical Report CRCT TR-11-94, Computer Science Department, Harvard University, Cambridge, Massachusetts. Available at `http://arXiv.org/`.

Sikkel, Klaas and Anton Nijholt. 1997. Parsing of context free languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Volume 2: Linear Modeling.* Springer, NY, pages 61–100.

Sportiche, Dominique. 1999. Reconstruction, constituency and morphology. GLOW, Berlin.

Stabler, Edward P. 1997. Derivational minimalism. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics*. Springer-Verlag (Lecture Notes in Computer Science 1328), NY, pages 68–95.

Stabler, Edward P. 1999. Remnant movement and complexity. In Gosse Bouma, Erhard Hinrichs, Geert-Jan Kruijff, and Dick Oehrle, editors, *Constraints and Resources in Natural Language Syntax and Semantics*. CSLI, Stanford, California, pages 299–326.

Vergnaud, Jean-Roger. 1982. *Dépendances et Niveaux de Représentation en Syntaxe*. Ph.D. thesis, Université de Paris VII.

Weir, David. 1988. *Characterizing mildly context-sensitive grammar formalisms*. Ph.D. thesis, University of Pennsylvania, Philadelphia.