Models of Adjunction in Minimalist Grammars

Thomas Graf

Department of Linguistics
Stony Brook University
mail@thomasgraf.net
http://thomasgraf.net

Abstract. Three closely related proposals for adding (cyclic) adjunction to Minimalist grammars are given model-theoretic definitions and investigated with respect to their linguistic and formal properties. While they differ with respect to their linguistic adequacy, they behave largely the same on a computational level. Weak generative capacity is not affected, and while subregular complexity varies betweeen the three proposals, it does not exceed the complexity imposed by Move. The closure of Minimalist derivation tree languages under intersection with regular tree languages, however, is lost.

Keywords: Minimalist grammars, adjunction, derivation trees, subregular tree languages, closure properties

Introduction

The distinction between arguments and adjuncts is recognized by a variety of linguistic formalisms. Although a number of empirical properties have been identified — for instance, adjuncts can be freely iterated and dropped from sentences — there is little consensus as to how adjuncts should be implemented. In the case of Minimalist grammars (MGs; [12]), a formalization of contemporary Chomskyan syntax, at least three different mechanisms have been proposed: adjunction as category-preserving selection, adjunction as asymmetric feature checking [3], and adjunction without feature checking [2].

This paper evaluates these three proposals with respect to their formal properties and linguistic adequacy. Building on [5, 6], I give a model-theoretic definition of each system in terms of constraints on Minimalist derivation trees and a mapping from these derivations to derived trees. The linguistic adequacy of these proposals is then evaluated with respect to a number of fundamental properties of adjuncts such as optionality and iterability, extending previous observations by Fowlie [2]. On the formal side, I compare Graf's results on subregular complexity of the derivation tree languages of standard MGs [5] to that of MGs with adjunction.

As summarized in Tab. 1 and 2 at the end of the paper, only the implementation without feature checking satisfies all linguistic criteria, but it is also the most complex, which is reflected by its higher subregular complexity. Furthermore, the closure under intersection with regular tree languages enjoyed by standard MGs [4, 9] is lost with all three variants of adjunction. Intuitively, this is due to the optionality and iterability of adjuncts, which each implementation needs to capture.

I proceed as follows: Section 1.1 recapitulates the constraint-based definition of MGs in [5], while Sec. 1.2 lists some formal properties of standard MGs. Sections 2.1–2.3 then look at each implementation of adjunction in greater detail. Even though the discussion is not particularly technical, the reader is expected to already be familiar with MGs and their constraint-based definition. All relevant details can be found in [4–6].

1 Minimalist Grammars

1.1 Definition in Terms of Derivation Tree Languages

I follow [5, 6] in defining MGs in terms of their Minimalist derivation tree languages (MDTLs) and a mapping from derivation trees to derived trees. This perspective will make it a lot easier later on to add adjunction operations to MGs and reason about their generative capacity, "derivational" complexity, and linguistic adequacy.

The definition of MDTLs is rather intuitive. Each lexical item (LI) is translated into a tree that corresponds to the contiguous subpart of the derivation that the LI controls via its positive polarity features. These trees are called *slices*. An MDTL is the largest set of trees that can be assembled from a finite number of slices without violating any constraints imposed by the MG feature calculus. As we will see in the next section, adding adjunction amounts to the introduction of new slices and modifying the constraints that regulate their distribution.

We start by defining features in a modular way.

Definition 1. *Let* BASE *be a non-empty, finite set of* feature names. *Furthermore,* OP $:= \{merge, move\}$ *and* POLARITY $:= \{+, -\}$ *are the sets of* operations *and* polarities*, respectively. A* feature system *is a non-empty set* $Feat \subseteq$ BASE $\times$ OP $\times$ POLARITY.

Negative Merge features are called *category features* (denoted f), positive Merge feature *selector features* ($=$f), negative Move features *licensee features* ($-f$), and positive Move features *licensor features* ($+f$). In the following, $\nu(f)$, $\omega(f)$ and $\pi(f)$ denote the name, operation, and polarity of f, respectively.

Definition 2. *Given a string alphabet Σ and feature system Feat, a* $(\Sigma, Feat)$-lexicon *is a finite subset of* $\Sigma \times \{::\} \times \{f \mid \pi(f) = +\}^* \times \{f \mid \omega(f) = merge, \pi(f) = -\} \times \{f \mid \omega(f) = move, \pi(f) = -\}^*$.

The ordering restriction on features is actually a corollary of the Minimalist feature calculus (see [4, 9]) and thus usually omitted. In anticipation of the modifications brought about by adjunction in the next section, though, I opt for explicitness over succinctness.

Next LIs are converted into slices in a top-down fashion:

Definition 3. *Let Lex be a* (Σ, Feat)*-lexicon and* $\mathit{Lex}_\star := \{\sigma :: f_1 \cdots f_n \star \mid \sigma :: f_1 \cdots f_n \in \mathit{Lex}\}$*. Then the* slice lexicon *of Lex is* $\text{slice}(\mathit{Lex}) := \{\zeta(l) \mid l \in \mathit{Lex}_\star\}$*, where* ζ *is given by*

$$\zeta(\sigma :: f_1 \cdots f_i \star f_{i+1} \cdots f_n) := \begin{cases} \sigma :: f_1 \cdots f_n \\ \qquad\qquad \text{if } f_1 \cdots f_i = \varepsilon \\ \zeta(\sigma :: f_1 \cdots f_{i-1} \star f_i \cdots f_n) \\ \qquad\qquad \text{if } \pi(f_i) = - \\ \mathit{move}(\zeta(\sigma :: f_1 \cdots f_{i-1} \star f_i \cdots f_n)) \\ \qquad\qquad \text{if } \tau(f_i) = \mathit{move} \text{ and } \pi(f_i) = + \\ \mathit{merge}(\Box, \zeta(\sigma :: f_1 \cdots f_{i-1} \star f_i \cdots f_n)) \\ \qquad\qquad \text{if } \tau(f_i) = \mathit{merge} \text{ and } \pi(f_i) = + \end{cases}$$

This definition uses the functional representation of trees, i.e. $f(t_1, \ldots, t_n)$ denotes the tree whose root is labeled f and whose i-th daughter is the root of tree t_i, $1 \leq i \leq n$. The symbol $\Box$ indicates a possible tree substitution site: For node u of tree s, $s[u \leftarrow t]$ is the result of substituting t for the subtree in s that is rooted in u. This is also called a *concatentation of s and t*. For slices s and t, $s[u \leftarrow t]$ is defined iff u is labeled $\Box$.

Given a slice lexicon slice(*Lex*), the *free slice language* FSL(slice(*Lex*)) is the smallest set that contains every tree t that is the result of concatenating finitely many $s \in \text{slice}(\mathit{Lex})$. The set of well-formed derivations is the largest subset of the free slice language whose trees obey certain tree-geometric conditions. These conditions correspond to constraints imposed by the Minimalist feature calculus.

An interior node m of $\zeta(l)$ is *associated to feature f_i on l* iff m is the root of $\zeta(\sigma :: f_1 \cdots f_i \star f_{i+1} \cdots f_n)$. Two features f and g *match* iff they have identical names and operations but opposite feature polarities. An interior node m matches a feature g iff the feature m is associated to matches g. Finally, the *slice root* of LI $l := \sigma :: f_1 \cdots f_n$ is the unique node of $\zeta(l)$ reflexively dominating every node in $\zeta(l)$.

Merge For every $t \in \text{FSL}(\text{slice}(\mathit{Lex}))$ and node m of t, if m is associated to selector feature $= f$, then its left daughter is the slice root of an LI with category feature f.

More succinctly, the selector features of an LI must be checked by an LI with a matching category feature.

Final Let $F \subseteq$ Base be a distinguished set of *final categories*. For every $t \in \text{FSL}(\text{slice}(\mathit{Lex}))$ and LI l, if the slice root of l is also the root of t, then the category feature c of l is a final category, i.e. $\nu(c) \in F$.

The conditions on *move* are only of ancillary importance to this paper. Consequently, I content myself with the bare definitions and do not further explore the reasoning behind them; the interested reader is referred to [5]. For every $t \in \text{FSL}(\text{slice}(\mathit{Lex}))$ and LI l in t with string $-f_1 \cdots - f_n$ of licensee features, $n \geq 0$, the *occurrences* of l in t are defined as follows:

- $occ_0(l)$ is the mother of the slice root of l in t (if it exists).
- $occ_i(l)$ is the unique node m of t labeled *move* such that m matches $-f_i$, properly dominates occ_{i-1}, and there is no node n in t that matches $-f_i$, properly dominates occ_{i-1}, and is properly dominated by m.

Intuitively, node m is an occurrence of LI l iff it denotes an operation that checks one of l's negative polarity features.

For t and l as before, and every node m of t:

Move There exist distinct nodes $m_1, \ldots, m_n$ such that m_i (and no other node of t) is the i^{th} occurrence of l, $1 \leq i \leq n$.
SMC If m is labeled *move*, there is exactly one LI for which m is an occurrence.

With these four constraints we can finally define MDTLs: Given an MG G with lexicon *Lex*, the MDTL of G is the largest $L \subseteq \mathrm{FSL}(Lex)$ such that every tree in L satisfies the four constraints above. Equivalently, each constraint can be taken to be the largest set of trees that satisfy the respective constraint, so that G's MDTL is the intersection of G's free slice language and the tree languages defined by **Final**, **Merge**, **Move** and **SMC**. Since all these tree languages are regular, it follows that MDTLs are too.

As mentioned at the beginning of this section, MDTLs must be mapped to derived tree languages. With the exception of *move*, this mapping is rather simple. The following is an informal description of the standard translation from derivation trees to multi-dominance trees (directed acyclic graphs), where the *phrasal root* of an LI is the same as its slice root:

1. **Linearize.** Switch the order of siblings l and n if l is an LI whose mother belongs to the slice of l.
2. **Project.** If n is a Merge node whose left daughter is an LI with at least one selector feature, relabel it $<$. All other interior nodes are labeled $>$.
3. **Add branches.** For every LI l, add branches from the phrasal root of l to each $occ_i(l)$, $i \geq 1$.
4. **Delete features.** Relabel every LI $\sigma :: f_1 f_2 \cdots f_n$ by σ.

These steps can be carried out by a tree-to-graph transduction Φ_{gr} that is definable in monadic second-order logic (MSO). The mapping Φ_{tr} from derivations to derived trees with traces is also MSO-definable. See [6, 11] for details.

Definition 4. *A* Minimalist Grammar *is a 5-tuple* $G := \langle \Sigma, Feat, Lex, F, \mathcal{R} \rangle$ *such that*

- *Lex is a* $(\Sigma, Feat)$*-lexicon, and*
- $F \subseteq$ Base *is the set of final features, and*
- $\mathcal{R}$ *is the set of regular tree languages* ***Final, Merge, Move, SMC****.*

The MDTL of G is $\mathrm{FSL}(\mathrm{slice}(Lex)) \cap \bigcap_{R \in \mathcal{R}} R$*. The tree language* $L(G)$ *generated by G is the image of its MDTL under the MSO transduction* Φ_{tr} *of [6]. Its string language is the string yield of* $L(G)$*.*

1.2 Formal Properties of MGs

MGs are weakly equivalent to MCFGs [10]. That MGs are at most as powerful as MCFGs actually follows from the fact that their string languages are the string yield of the image of regular tree languages under an MSO-definable transduction [11]. For the purpose of adding adjunction, this means that weak generative capacity is preserved as long as MDTLs are still regular and the mapping to derived trees is MSO-definable.

In [5], it is shown that MDTLs are actually subregular. They are definable in first-order logic with proper dominance (abbreviated FO[$<$]), and *homogeneous.*

Definition 5 (Homogeneity). *For s_1, s_2, t_1, t_2 arbitrary trees and L a regular tree language, L is* homogeneous *iff $t[u \leftarrow a(t_1, t_2)] \in L$, $t[u \leftarrow a(s_1, t_2)] \in L$, and $t[u \leftarrow a(t_1, s_2)] \in L$ jointly imply $t[u \leftarrow a(s_1, s_2)] \in L$.*

For MGs without movement (i.e. no LI with licensee features occurs in a well-formed derivation), the MDTLs are also definable in first-order logic with immediate dominance (FO[S]), closed under k-guarded vertical swap, and strictly local.

Definition 6 (Vertical swap). *Let t be the concatenation of trees t_1, t_2, t_3, t_4, t_5 such that for $1 < i \leq 5$, the root of t_i is immediately dominated by some node n_{i-1} of t_{i-1}. The vertical swap of t between t_2 and t_4 is the result of switching t_2 and t_4 in t such that the roots of t_2, t_3, t_4, t_5 are immediately dominated by n_3, n_4, n_1, and n_2, respectively. The vertical swap is k-*guarded *iff it holds that t_2 and t_4 are identical up to depth k, and so are t_3 and t_5.*

Definition 7 (Strictly local). *Given a tree t over alphabet Σ, its k-*augment *is the result of adding nodes $n_1, \ldots, n_k$ above the root and below each leaf such that n_i immediately dominates n_{i+1}, $1 \leq i < k$ and each n_i has the distinguished label $\diamond \notin \Sigma$. A k-*factor *of t is a subtree of t that has been truncated at depth k. That is to say, if s is a subtree of t with m nodes $n_1, \ldots n_m$ that are labeled $l_1, \ldots, l_m$, respectively, and properly dominated by $k-1$ nodes, then $s[n_1 \leftarrow l_1] \cdots [n_m \leftarrow l_m]$ is a k-factor of t. The set of k-factors of t is denoted $F_k(t)$. A regular tree language L over alphabet Σ is* strictly local *iff there is some $k \in \mathbb{N}$ and finite set S of trees over $\Sigma \cup \{\diamond\}$ with depth at most k such that $t \in L$ iff $F_k(t') \subseteq S$, where t' is the k-augment of t.*

These properties are interesting because they tell us something about the complexity of the constraints imposed by the feature calculus. Closure under k-guarded vertical swaps implies local threshold testability, while homogeneity is equivalent to recognizability by a particular kind of deterministic top-down tree automaton. Strict locality tells us that no dependency is unbounded.

2 Three Models of Adjunction

2.1 Category Preserving Selection

The simplest model of adjunction is naturally one that does not require any modifications to the formalism. Taking inspiration from Categorial Grammar, adjuncts can be implemented as LIs whose category feature has the same feature name as their selector feature. For example, the VP-adjunct *quickly* would correspond to quickly :: $=$V V. Such cases of selection are *category preserving*. This model was first discussed in [2], but it has been part of the general "MG folklore" for a long time.

Linguistic Properties Despite its simplicity, this account captures several properties of adjuncts. Since an LI can only be selected once all its positive polarity features have been checked — which corresponds to projection in the derived tree — category preserving LIs may only select LIs that have already projected their full structure. Consequently, adjuncts are correctly predicted to adjoin only to maximal projections.

The category preserving nature of adjuncts in this model also entails that adjuncts are optional with respect to Merge. If an LI can select an adjunct a, then it can also directly select the LI l that a adjoins to as both have the same category feature. Category preservation also implies that multiple adjuncts can adjoin one after another, that is to say, adjuncts are iterable.[1]

Certain important aspects of adjuncts are missed, though. First of all, nothing in this system rules out the existence of adjuncts that adjoin to multiple phrases at the same time. Consider an entry with two identical selector features, e.g. quickly :: $=$V $=$V V. This LI could be used to generate a structure where two VPs are simultaneously modified by a single adverb occurring between them, yet no such structures are attested.

Another problem is posed by ordering effects. These can of course be handled via subcategorization in the familiar fashion: if adjunct a cannot precede adjunct b, then a does not select b, nor are there LIs $c_1, \ldots, c_n$ such that a selects c_1, c_i selects c_{i+1} ($1 \leq i < n$), and c_n selects b (cf. [2]). However, since these categories must be pair-wise distinct, the LIs are not category-preserving. Hence they are not necessarily optional or iterable. As was already pointed out by Fowlie [2], these properties can be enforced globally in the lexicon, but the cost is lexical redundancy that hides the important generalizations.

Movement is also challenging. Since the adjunct selects the phrase it supposedly adjoins to, it is not included in the phrase projected by the latter. So if the adjoinee undergoes movement, the adjunct is left behind, just like a complement

[1] Notice that these properties break down as soon as movement is involved — removal of an adjunct that contains a licensee feature might render the derivation ill-formed, and having multiple instances of such an adjunct may trigger SMC violations. However, if movement out of adjuncts is prohibited by something like the Adjunct Island Constraint, for which there is a lot of empirical support, then problems arise only where the adjunct itself undergoes movement. Displacement of adjuncts, though, might not involve movement at all and may simply be an instance of a phrase or LI being base-generated in a different position than where it is interpreted. For this reason — and because all conceivable accounts of adjunction have similar problems reconciling movement with optionality [7] — I will ignore movement dependencies in the remainder of this paper as far as optionality and iterability are concerned.

DP that undergoes movement leaves behind the VP containing it. This can be fixed by I) instantiating the licensee features on lexical heads (suggested by an anonymous reviewer), or II) adding a limited form of pied-piping to the mapping from derivation trees to derived trees such that if l undergoes movement, the movement branches connect the occurrences of l to the slice root of the highest adjunct of l, rather than the slice root of l itself.

The first solution quickly leads to massive redundancy in the lexicon and once again misses generalizations. Suppose our grammar contains the LIs $l_1 := \mathrm{s} :: \mathrm{c}\ -\mathrm{f}$ and $l_2 := \mathrm{t} :: \mathrm{c}$. If licensee features are instantiated on empty heads instead, the lexicon would contain the LIs $l'_1 := \mathrm{s} :: \mathrm{c}$, $l'_2 := \mathrm{t} :: \mathrm{c}$, and $f := \varepsilon :: =\mathrm{c}\ \mathrm{c}\ -\mathrm{f}$. So if l_1 moves together with an adjunct a, this corresponds to l_1 being selected by a, which in turn is selected by the actual mover f. But keep in mind that l_2 should not be allowed to move, so the corresponding l'_2 must not be selected by f. This can only be guaranteed by changing the category of l'_2 to, say, c'. However, every LI except f that selects an LI of category c must still be able to select l'_2, so for each one of them we have to create a new variant whose selector feature is $=\mathrm{c}'$ instead of $=\mathrm{c}$. Not only does this unnecessarily increase the size of the lexicon, the fact that l_1 and l_2 have the same adjuncts is purely accidental in this revised grammar.

The second option avoids the lexical blow-up by enabling the transduction to pied-pipe adjuncts. But since adjuncts differ from non-adjuncts only in that they are category preserving, this may cause problems when non-adjuncts are also category preserving — for instance in the analysis of serial verb constructions in [1], where each verb except the most deeply embedded one has the feature component $=\mathrm{V}\ \mathrm{V}$. This actually highlights a more fundamental problem of this approach: adjuncts do not form a natural class since arguments may have the same feature make-up.

The inability to consistently distinguish arguments from adjuncts can also be seen in the case of recursive adjunction, i.e. adjunction to an adjunct. Without tricks, category-preserving selection cannot handle such configurations. Suppose that b adjoins to a, which is an XP-adjunct and therefore has category feature X. In order to adjoin to a, LI b must have selector feature $=\mathrm{X}$, and by category preservation it also has category X. But then b is an XP-adjunct just like a. So a phrase like *very red car* only has a structure where *very* modifies *car* rather than *red*.

As pointed out by an anonymous reviewer, this can be handled via empty heads, but the proposed solution serves only to highlight the fact that arguments cannot be separated from adjuncts in this system. Suppose that very :: adv and red :: a are selected by the empty head $\varepsilon :: =\mathrm{a}\ =\mathrm{adv}\ \mathrm{a}$, and this complex phrase is then combined with car :: n by the empty head $\varepsilon :: =\mathrm{n}\ =\mathrm{a}\ \mathrm{n}$. Then we obtain the structure [[very [ε red]] [ε car]], which is reasonably close to the intended [[very red] car]. But now consider cases where the adjunct follows the noun, e.g. *my cousin twice removed*. Here the empty head should be $\varepsilon :: =\mathrm{a}\ =\mathrm{n}\ \mathrm{n}$. This very feature template also arises in the standard DP-analysis of English possessor phrases such as *John's car*, where the possessive marker is given by 's :: $=$n $=$d d. Yet this structure is not assumed to involve adjuncts of any kind. We see, then, that adjuncts cannot be reliable identified with a specific type of feature component.

Formal Properties While modeling adjunction as category preserving selection has some important drawbacks from a linguistic perspective, it has the advantage of being compatible with the standard MG formalism. Consequently, all formal properties of MDTLs are unaffected. Only closure under intersection with regular tree languages can be lost under very specific assumptions. The proofs in [4, 9] rely on the ability to subscript category and selector features with specific states of a tree automaton. In many cases, an LI will have different state-suffixes on its selector and category features. As a result an LI may no longer be category preserving after its features have been suffixed with states. Hence, if suffixation must respect category preservation, then the class of MDTLs is no longer closed under intersection with regular tree languages. Or the other way round: the property of being an adjunct is not preserved under intersection with regular tree languages.

Example 1. Consider the regular tree language that includes a tree t iff t contains at most three LIs whose phonetic exponent is *quickly*. If the LI l for *quickly* must be category preserving, then irrespective of how its features are altered, one instance of l can always be selected by another one. Hence if the MDTL contains a tree with at least one instance of *quickly*, it also contains trees with more than three of them.

2.2 Asymmetric Feature Checking

A very different implementation of adjunction was presented by Frey and Gärtner [3]. Adjuncts are now formalized as LIs whose category feature c has been replaced by an adjunction feature $\approx$a. An LI l with feature $\approx$a can adjoin to any LI l' of category a. Crucially, the adjunction operation is asymmetric in that it only checks the adjunction feature of l, whereas the category feature on l persists, thereby allowing for multiple LIs to adjoin to it. This partial feature persistence of adjunction sets it apart from Merge, which always checks the relevant features of both the selector and the selectee. In addition, it is the phrase being adjoined to that projects, rather than the adjunct itself. An example derivation with the corresponding derived tree is given in Fig. 1.[2]

Definition Only a few things have to be altered in our definition of MDTLs to incoporate Frey and Gärtner's version of adjunction. First, adjunction features must be added to the feature system and restricted to be in complementary distribution with category features. To this end, Def. 1 is revised such that $\textsc{Op} := \{\mathit{merge}, \mathit{move}, \mathit{adjoin}\}$, and the defintion of LI is altered accordingly:

[2] A related system is developed by Hunter in [8], who uses the same feature checking mechanism but a different type of Minimalist derivations. As far as I can tell, my observations about Frey and Gärtner's system apply to Hunter's, too.

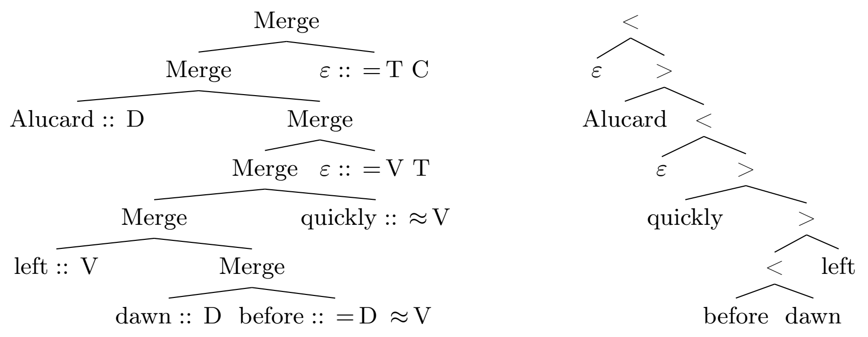

Fig. 1. Left: derivation tree with adjunction as asymmetric feature checking; Right: corresponding derived tree

Definition 8. *Given a string alphabet Σ and feature system Feat, a $(\Sigma, Feat)$-*lexicon *is a finite subset of* $\Sigma \times \{::\} \times \{f \mid \pi(f) = +\}^* \times \{f \mid \omega(f) \in \{merge, adjoin\}, \pi(f) = -\} \times \{f \mid \omega(f) = move, \pi(f) = -\}^*$.

The only change from Def. 2 is that the unique category feature may be replaced by an adjunction feature of negative polarity.

The next step is to assign adjuncts an interpretation in terms of derivation trees. That is to say, both the translation from LIs to slices and the well-formedness conditions on MDTLs need to be amended. Def. 3 is extended to cover one more case: $merge(\Box, \zeta(\sigma :: f_1 \cdots f_{i-1} \star f_i \cdots f_n))$ if $\tau(f_i) = adjoin$. Note that even though adjuncts do not project in the derived tree, in the derivation tree the adjunction step belongs to the slice of the LI with the adjunction feature. So from a derivational perspective adjunction looks very similar to Merge (we do not even introduce a new label to distinguish the two).

A simple modification of the constraint **Merge** suffices to regulate the distribution of adjuncts.

Merge For every $t \in \text{FSL}(\text{slice}(Lex))$ and node m of t, if m is associated to selector feature $=f$ or adjunction feature $\approx f$, then its left daughter is the slice root of an LI with category feature f or adjunction feature $\approx f$.

The only difference to the original definition is the presence of the two disjuncts "or adjunction feature $\approx f$". The first disjunct allows Merge to be triggered by adjunction features, too. The second one allows for configurations where a phrase has multiple adjuncts, as in Fig. 1. In this case, only the Merge node of the lowest adjunct is the mother of the slice root of an LI with category feature f. For a higher adjunct a, however, the slice root belongs to an LI b with an adjunction feature $\approx f$. As long as the two adjunction features are the same, though, it follows by induction that there is some LI further down that both a and b adjoin to.

This completely characterizes the adjunction operation on a derivational level, so it only remains for us to modify the mapping from derivations to derived trees. Two steps must be altered. **Linearize** now distinguishes two cases where siblings are switched around:

Linearize Switch the order of siblings l and n if
- l is an LI whose mother belongs to the slice of l, or
- the mother of l is a Merge node associated to an adjunction feature.

In the definition of **Add branches**, the phrasal root of LI l is now defined as either the slice root of the highest adjunct of l or the slice root of l if the former does not exist. Without this change, our model-theoretic definition would differ from Frey and Gärtner's in that adjuncts would be stranded if the phrase they adjoin to undergoes movement (a problem already encountered with the category preservation model).

The change to how phrasal roots are determined also solves a minor problem in the definition of **Final**, which is no longer adequate because the slice root of the head of a tree may no longer be the root of the tree. However, its phrasal root still is, so it suffices to replace "slice root" by "phrasal root" in **Final**.

Formal Properties Considering how little needs to be changed in the definitions, it is hardly surprising that most formal properties of MGs also hold after the introduction of a dedicated adjunction mechanism. The revised version of **Merge** only adds a few disjunctions, wherefore it is still MSO-definable and defines a regular tree language. The new clause in **Linearize** is also easily expressed in MSO, as is the new definition of phrasal root (cf. Sec. 2.2 of [6]). So both MDTLs and their mapping to derived trees are still MSO-definable, which entails that this variant of MGs generates at most MCFLs, and consequently adjunction has no effect on weak generative capacity.

Even the subregular complexity of MDTLs is unaffected. For MGs without Move, they are still strictly local since the domain for **Merge** comprises only two slices. This also implies that adding adjunction to MGs with movement does not negatively affect their definability in FO[$<$]. Finally, they are homogeneous and therefore can be recognized by lrDTDAs.

Lemma 1. *Let G be an MG with adjunction as asymmetric feature checking. Then G's MDTL is homogeneous.*

Proof. Recall that a tree language L is homogeneous iff $t[u \leftarrow a(t_1, t_2)] \in L$ and $t[u \leftarrow a(s_1, t_2)] \in L$ and $t[u \leftarrow a(t_1, s_2)] \in L$ jointly imply $t[u \leftarrow a(s_1, s_2)] \in L$. We are only interested in subtrees whose root a is a Merge node associated to an adjunction feature. All other cases are already covered by the homogeneity proof for standard MDTLs in [5].

Let $t[\approx f]$ and $t[= f]$ denote that the head of t has $\approx f$ and $= f$ as its first feature, respectively, whereas $t[f]$ indicates that there is an LI l in t whose first unchecked feature is f and every Merge node properly dominating the slice root of l is the slice root of an LI with feature $\approx f$. Then $t[u \leftarrow a(t_1, t_2)] \in L$ only if $t_i[f]$ and $t_j[\approx f]$ for $i \neq j \in \{1, 2\}$. Assume w.l.o.g. that $i = 1$ and $j = 2$. Then it must also be the case that $s_1[f]$, and $s_2[\approx f]$ or $s_2[= f]$, so that $a(s_1, s_2)$ is well-formed with respect to **Merge**. Furthermore, we know that s_1 and t_1 on the one hand and s_2 and t_2 on the other have the same unchecked licensee features since substituting one for the other preserves well-formedness. It follows that $t[u \leftarrow a(s_1, s_2)] \in L$.

As in the case of category-preserving selection, the status of closure under regular intersection is not entirely straight-forward. Simply applying the suffixation strategy of [4, 9] is insufficient. Once again this is illustrated by the regular tree language in example 1, which contains only trees with at most three occurrences of *quickly*. If one instance of *quickly* can adjoin to a given LI l, then arbitrarily many instances of it may adjoin to l. Hence there can be no MG G whose MDTL contains trees with up to three occurrences of a, but not more than that.

It is possible, however, to switch from asymmetric feature checking to standard symmetric Merge using category-preserving selection without changing the shape of the derivation tree — both are just instances of Merge. To the extent that this is a licit step, closure under regular intersection would once again hold if adjunct status need not be preserved. But even though the derivation trees would be identical, the derived structures are not: an adjunct in Frey and Gärtner's system does not project, whereas a category-preserving head does. And since we already saw in the discussion of the category-preservation account that not every instance of category-preserving selection constitutes adjunction, a derivation with adjunction cannot be uniquely recovered from an isomorphic one with selection. This illustrates once again that closure under regular intersection can be obtained only if LIs may lose their adjunct-status.

Linguistic Properties Just like the category preservation approach, the implementation of adjunction as asymmetric feature checking captures several core properties of adjuncts. Since category features are unaffected by adjunction, adjuncts are correctly predicted to be optional and iterable (*modulo* movement dependencies). In addition, adjunction features behave similar to selector features in that they are checked by category features. An LI therefore can be adjoined to only after it has discharged all its positive polarity features, i.e. projected a full phrase in the derived tree, which rules out X$'$-adjuncts.

In contrast to the category preservation approach, Frey and Gärtner's implementation also behaves correctly with respect to Move — an adjunct moves together with the phrase it is adjoined to. Admittedly this has to be explicitly stipulated in the definition of phrasal root, but the clear division between adjunction and selection in the feature system means that this modification does not bring about any unexpected side-effects. Another welcome property is that thanks to the limit to one adjunction feature, adjoining to multiple phrases at once is impossible.

But Frey and Gärtner's approach is not without shortcomings, either. Since adjunction features replace category features, adjuncts lack category features and thus cannot be adjoined to. So just like the category preservation implementation, asymmetric feature checking cannot assign the correct structure to *very red car*. Even empty heads only solve the problem if one treats the adjuncts as arguments of an unpronounced adjunct, which defeats the purpose of having an explicit argument-adjunct distinction in the system.

Ordering effects are also unaccounted for. Since adjunction to an LI l with category feature f can be triggered only by the feature $\approx f$, all adjuncts of l have said category feature. From this it follows immediately that these adjuncts may adjoin in any given order and **Merge** will still be satisfied. Fowlie [2] sketches a workaround based on a Cinque-style hierarchy of empty heads, each of which serves as an adjunction site for adjuncts of a particular type. Still, the order is enforced by selection rather than the adjunction mechanism itself, which reintroduces many problems of treating adjunction as selection, for instance regarding phrasal projection and the interaction with movement.

2.3 Adjunction Hierarchies

A third approach has been recently proposed by Fowlie in [2]. Fowlie's primary interest is to reconcile optionality with ordering effects. Technically this is an easy task if one uses standard selection: if LI l has selector feature $=\mathrm{c}$, and LIs of category c may be adjoined to by adjuncts $a_1, \ldots a_n$ as long as each a_i is structurally lower than a_j, $1 \leq i < j \leq n$, then we have n additional versions of l where $=\mathrm{c}$ has been replaced by $=\mathrm{x}$, where x is the category feature of some a_i, $1 \leq i \leq n$. But this solution comes at the price of a significantly larger lexicon.

Fowlie proposes to put a partial order R on the set of categories instead. Moreover, there are no adjunction features; adjunction of a to l has no effect on the feature make-up of either LI and may take place as long as

- for c_a and c_l the category features of a and l, respectively, and $\alpha : \textsc{Base} \rightarrow \wp(\textsc{Base})$ a map from categories to sets of categories that may adjoin to them, it must hold that $c_a \in \alpha(c_l)$, and
- $c_a \not R\, c_l$, and
- for every LI b of category c_b that adjoined to l before a, $c_a \not R\, c_b$.

Notice that $x \not R\, y$ is compatible with $y \not R\, x$ and $y\ R\ x$. This is used by Fowlie to make a distinction between order-insensitive and order-sensitive adjuncts, respectively, with the former being linearized to the right of the adjoinee and the latter to the left (the split is motivated by empirical observations). For the sake of simplicity I ignore this distinction in what is to follow.

Definition While a literal implementation of the formalism in [2] is tedious because of the way non-local information is passed around via feature pairs, this mechanism can safely be ignored for a model-theoretic definition. As a matter of fact, this is preferable from a formal perspective since local book-keeping of non-local information could obscure the subregular complexity of adjunction in derivation trees. Without Fowlie's feature pairs, the feature system is exactly the one of standard MGs. Consequently, all modifications take place on the level of derivation trees and the mapping to derived trees.

For derivation trees, a viable strategy is to insert Adjoin nodes at arbitrary points and then filter out the illicit derivations created this way. Technically, this is achieved by a minimal change in the definition for slice lexicons such that $\text{slice}(\mathit{Lex}) := \{\zeta(l) \mid l \in \mathit{Lex}_*\} \cup \{\mathit{adjoin}(\Box_1, \Box_2)\}$. As a result, the free slice language not only contains combinations of lexical slices, but also trees where binary branching *adjoin* nodes occur in random positions (but not within lexical slices).

For movement, the presence of adjunction nodes causes no problems because the constraints **Move** and **SMC** are non-local. But **Merge** was stated under the assumption that the left daughter is the slice root of an LI with the matching category feature. Adjunction destroys this local relation. In the system of Frey and Gärtner, this could still be worked around due to the feature-driven nature of adjunction. An LI with feature $\approx f$ could only adjoin to LIs of category f. But this isn't necessarily the case in this system, where α^{-1} — which determines for every category the categories it may adjoin to — is not guaranteed to be a function. Prepositional phrases, for example, can adjoin to both nouns and verbs. So if a Merge node is associated to feature $=\text{N}$ and its left daughter is a node indicating adjunction of a PP, it is still unclear whether a matching feature N can be found further down the tree. But suppose that we always interpret *adjoin* nodes in a fashion such that the adjoinee is found along the left branch and the adjunct along the right branch. Suppose m *left-dominates* n iff m properly dominates n and there is no z such that z is properly dominated by m, reflexively dominates n, and has a left sibling. Then the following definition will do the trick:

Merge For every $t \in \text{FSL}(\text{slice}(\mathit{Lex}))$ and node m of t, if m is associated to selector feature $=f$, then the highest node that is left-dominated by m and not labeled *adjoin* is the slice root of an LI with category feature f.

This takes care of Merge nodes, but it still remains for us to regulate the distribution of *adjoin* nodes. The first step is to determine the arguments of each adjunction step, i.e. the adjoining phrase and the phrase being adjoined to. If adjuncts cannot be adjoined to, this is very easy. For then the right daughter is the slice root of the adjunct, and the slice root of the phrase being adjoined to is the highest Merge node that is left-dominated by the adjoin node (once again left-dominance is used to account for the fact that there may be other adjoin nodes along the path). Somewhat surprisingly, though, allowing adjuncts to be adjoined to is an easy process once left-dominance has been defined.

First, an adjunction node m is an *adjunction occurrence* of LI l in derivation tree t iff m is the lowest node in t that properly dominates the slice root of l but does not left-dominate the slice root of l. Second, an adjunction node m is associated to category feature c iff m is an adjunction occurrence of LI l with category feature c.

Adjoin For every $t \in \text{FSL}(\text{slice}(\mathit{Lex}))$ and node m of t, if m is associated to category feature c_m, then

- the highest Merge node in t left-dominated by m is the slice root of an LI l with category feature c_l, and

- $c_m \in \alpha(c_l)$, and
- for every node n that is properly dominated by m, reflexively dominates l, and is associated to category feature c_n, it holds that $c_m \not R\, c_n$.

As with Frey and Gärtner's system, we also have to make changes to the mapping from derivations to derived trees.

Linearize Switch the order of siblings l and n if
- l is an LI whose mother belongs to the slice of l, or
- the mother of l is an adjunction node.

The phrasal root of an LI l (referenced in **Add Branches** is now the highest adjunction node n such that n properly dominates the slice root of l and no Merge properly dominated by n properly dominates the slice root of l. Once again **Final** is easily adapted to the new system by replacing "slice root" by "phrasal root".

Figure 2 gives an example of what derivations with multiple adjunctions look like in this system, and what kind of structures they yield.

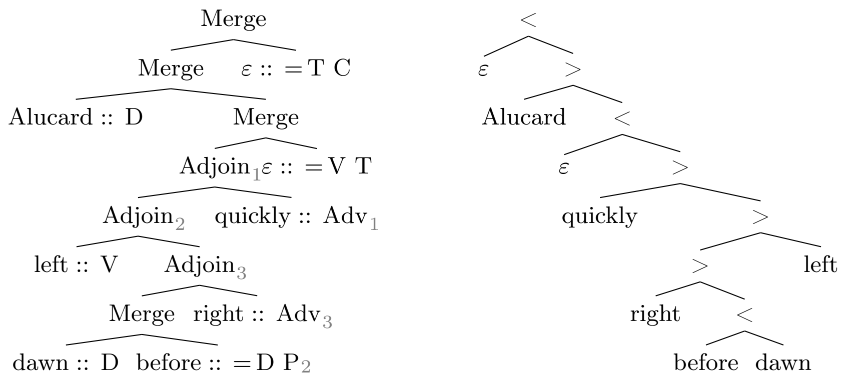

Fig. 2. Left:derivation tree with recursive adjunction, suffixes indicate adjunction occurrences; Right: corresponding derived tree

Formal Properties As with the previous model the changes in the definitions are innocuous enough to see that the formal properties of MDTLs are mostly unaffected. The major change is the introduction of left-dominance, which can easily be defined in first-order logic with proper dominance. The function α and the relation R are both finite by virtue of Minimalist lexicons being finite, so they, too, are first-order definable. From all this it follows that both the MDTLs and their mapping to derived trees are MSO-definable and weak generative capacity is not increased.

The definability of MDTLs in FO[$<$] is not endangered either, because the non-local dependencies established by adjunction are no more complex than those regulating Move. For MGs without movement, however, adjunction does increase subregular complexity significantly. First of all, the dependence on left-dominance, an unbounded relation, means that MDTLs are no longer strictly local. This can only be avoided by banning adjunction to adjuncts, which allows for an unbounded number of adjoin nodes to occur between an LI and its adjunction occurrence. But even then α and R have to be chosen very carefully to ensure that MDTLs are strictly local: the category of a phrase must be predictable from the categories of its adjuncts, which is not the case for natural language (*very*, for example, is freely iterable and may be an adjunct of adjectives or adverbs).

Without strong restrictions on α and R, MDTLs are not even homogeneous or closed under k-guarded vertical swap.

Lemma 2. *The MDTLs of MGs with hierarchical adjunction are not closed under k-guarded vertical swap.*

Proof. Consider a grammar containing (at least) the following items:

$$\text{b} :: \text{b} \qquad \text{a} :: =\text{b a} \qquad \text{b} :: =\text{b b}$$

Furthermore, $F := \{a, b\}$, $\alpha(a) = \alpha(b) = \{b\}$, and $b \not R\, b$. Let d be the derivation tree $\mathit{merge}(\mathit{merge}(\text{b} :: \text{b}, \text{b} :: =\text{b b}), \text{a} :: =\text{b a})$, and let d^n be the derivation where n instances of b :: b adjoin to each LI in d. Then for every $k \in \mathbb{N}$ there is some n such that d^n can be factored into t_1, t_2, t_3, t_4, t_5, where every subtree consists of adjunction nodes and instances of b :: b, and t_2 contains a :: =b b and the corresponding Merge node at some depth $f > k$, t_3 contains b :: =b b and the corresponding Merge node at some depth $g > k$, and t_4 contains the original b :: b at some depth $h > k$. Removing the adjuncts from the k-guarded vertical swap of d^n between t_2 and t_4 yields the ill-formed derivation tree $\mathit{merge}(\mathit{merge}(\text{b} :: \text{b}, \text{a} :: =\text{b a}), \text{b} :: =\text{b b})$, whence the k-guarded vertical swap is not contained in the grammar's MDTL.

Lemma 3. *The MDTLs of MGs with hierarchical adjunction are not homogeneous.*

Proof. Consider the following grammar:

$$\text{a} :: \text{a} \qquad \text{b} :: \text{b} \qquad \text{c} :: \text{c} \qquad \text{d} :: \text{d}$$

Suppose $F := \{a, b\}$, $\alpha(a) := \{c, d\}$ and $\alpha(b) := \{c\}$. Now let $a =$ adjoin, $t_1 :=$ a :: a, $t_2 :=$ c :: c, $s_1 :=$ b :: b, and $s_2 :=$ d :: d. Then $a(t_1, t_2)$, $a(t_1, s_2)$, and $a(s_1, t_2)$ are all well-formed derivations, but $a(s_1, s_2)$ is not.

Closure under intersection with regular tree languages is also lost. Admittedly our standard example — the regular language of trees that contain at most three instances of *quickly* — can be accommodated in this system (use three different categories c_i for *quickly*, such that $1 \leq i \leq 3$ and c_j may adjoin to c_i iff $j > i$). But we still run into problems with its dual, the regular language of trees that contain at least three instances of *quickly*. Since adjuncts are completely decoupled from the feature checking mechanism, they are not required in order for a derivation to be well-formed, wherefore the presence of even one instance of *quickly* cannot be enforced.

Linguistic Properties The hierarchical approach improves significantly on both previous proposals, to the extent where it passes all criteria discussed in this paper: optionality, iterability, recursive adjunction, correct behavior with respect to Move, the ability to capture ordering effects, and the prohibition against X′-level adjunction. It should be noted, though, that some properties do not fall out naturally under the model-theoretic perspective and are simply a matter of how we phrase our definitions. The interaction with Move, for example, depends purely on the how *phrasal root* is defined, and hence could easily be altered. It would also be a rather easy technical exercise to allow for X′-level adjunction. In addition, some properties depend on the choice of R. If R is reflexive, for instance, then adjuncts cannot be iterated because the condition $c_m \not R\, c_n$ in **Adjoin** would be trivially violated. Optionality, however, is a robust property of this system thanks to the decoupling of adjunction and feature checking.

Conclusion

An overview of the formal and linguistic properties of the three models of adjunction are given in Tab. 1 and 2. The emerging picture is that all accounts capture the most basic properties of adjuncts — optionality, iterability, the lack of X′ adjuncts — but diverge once one considers other aspects such as adjunction to adjuncts, the interaction with Move, and ordering effects. Only Fowlie's hierarchical approach performs well across the board, but does so at the expense of increasing the subregular complexity of MDTLs, even with respect to standard MGs. Loss of homogeneity and closure under k-guarded vertical swap, however, are unavoidable in any implementation of adjunction where adjuncts are iterable and can adjoin to phrases with different categories, both of which seem to be empirical necessities. Similarly, closure under intersection with regular tree languages is incompatible with the optionality of adjuncts.

Bibliography

[1] Collins, C.: Argument sharing in serial verb constructions. Linguistic Inquiry 28, 461–497 (1997)
[2] Fowlie, M.: Order and optionality: Minimalist grammars with adjunction. In: Kornai, A., Kuhlmann, M. (eds.) Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13). pp. 12–20 (2013)
[3] Frey, W., Gärtner, H.M.: On the treatment of scrambling and adjunction in minimalist grammars. In: Proceedings of the Conference on Formal Grammar (FGTrento). pp. 41–52. Trento (2002)

	Cat. Preserv.	Asymm. Checking	Hierarchical
no X$'$ adjuncts	✓	✓	✓
optional	✓	✓	✓
iterable	✓	✓	✓
recursive	$\sim$	$\sim$	✓
no double adjunction		✓	✓
ordering effects	✓	$\sim$	✓
correct projection		✓	✓

Table 1. Linguistic properties of adjunction implementations

	Cat. Preserv.	Asymm. Checking	Hierarchical	Move
strictly local	✓	✓		
vertical swap	✓	✓		
homogeneous	✓	✓		✓
FO[S]	✓	✓		
FO[$<$]	✓	✓	✓	✓
reg $\cap$				✓
preserves gen. capacity	✓	✓	✓	NA

Table 2. Formal properties of adjunction implementations (without Move)

[4] Graf, T.: Closure properties of minimalist derivation tree languages. In: Pogodalla, S., Prost, J.P. (eds.) LACL 2011. Lecture Notes in Artificial Intelligence, vol. 6736, pp. 96–111. Springer, Heidelberg (2011)

[5] Graf, T.: Locality and the complexity of minimalist derivation tree languages. In: de Groot, P., Nederhof, M.J. (eds.) Formal Grammar 2010/2011. Lecture Notes in Computer Science, vol. 7395, pp. 208–227. Springer, Heidelberg (2012)

[6] Graf, T.: Local and Transderivational Constraints in Syntax and Semantics. Ph.D. thesis, UCLA (2013)

[7] Graf, T.: The syntactic algebra of adjuncts. In: Proceedings of CLS49 (to appear)

[8] Hunter, T.: Deriving syntactic properties of arguments and adjuncts from neo-davidsonian semantics. In: Ebert, C., Jäger, G., Michaelis, J. (eds.) The Mathematics of Language. Lecture Notes in Computer Science, vol. 6149, pp. 103–116. Springer, Heidelberg (2010)

[9] Kobele, G.M.: Minimalist tree languages are closed under intersection with recognizable tree languages. In: Pogodalla, S., Prost, J.P. (eds.) LACL 2011. Lecture Notes in Artificial Intelligence, vol. 6736, pp. 129–144 (2011)

[10] Michaelis, J.: Transforming linear context-free rewriting systems into minimalist grammars. Lecture Notes in Artificial Intelligence 2099, 228–244 (2001)

[11] Mönnich, U.: Grammar morphisms (2006), ms. University of Tübingen

[12] Stabler, E.P.: Derivational minimalism. In: Retoré, C. (ed.) Logical Aspects of Computational Linguistics, Lecture Notes in Computer Science, vol. 1328, pp. 68–95. Springer, Berlin (1997)