Adjunction in (T)SL Syntax

Kenneth Hanson
Department of Linguistics
Stony Brook University
mail@kennethhanson.net

Abstract

Adjunction is intuitively a local operation, yet its subregular complexity is dependent on both the geometry of the syntactic representation as well as the specific model of adjunction assumed. Here, I propose a model of adjunction which is strictly local (SL) over Minimalist Grammar (MG) dependency trees, and which incorporates the core properties of optionality, iteration, invisibility to selection, and adjunct ordering restrictions. Non-locality is avoided in cases of recursive adjunction, and an interesting treatment of several other formal properties of adjunction is made possible.

1 Introduction

In the last several years, a two-level classification of the computational complexity of syntax has emerged: local dependencies such as selection are *strictly local* (SL) over trees, while non-local dependencies such as movement, agreement, and case assignment are *tier-based strictly local* (TSL), a straightforward generalization of SL in which a subset of non-salient elements are ignored (Graf, 2018, 2022b; Hanson, 2023b, 2025; Vu et al., 2019). This closely matches past results on local and non-local phonological dependencies, which are predominantly SL and TSL over strings, respectively (Heinz, 2018), providing evidence of cognitive parallelism across linguistic domains (Graf, 2022a).

The placement of adjunction within this scheme, however, has remained unclear, as formal models of adjunction vary in their subregular complexity (Graf, 2014). Furthermore, the complexity of adjunction interacts with that of selection: in the derivation tree language for a Minimalist Grammar (MG) with recursive adjunction, the complexity of selection is increased to TSL (Graf, 2018). This is not a terrible state of affairs, as it would mean that the overall complexity of much of syntax is quite low, and uniform across operations. At the same time, selection is typically considered to be highly local. For example, a verb may select the category of its complement, but not the complement of its complement, let alone more distant items, yet this is exactly what we would predict if selection was TSL. Similarly, most of the key properties of adjunction require only a SL grammar (Hanson, 2023a). We therefore ask: can the non-locality of adjunction, and by extension selection, be eliminated?

The answer is affirmative. With minor adjustments, the *MG dependency tree model* defined in Graf and Kostyszyn (2021) can easily accommodate a linguistically satisfactory SL model of adjunction, which includes the core properties of optionality, iteration, invisibility to selection, and ordering restrictions among adjuncts. The primary change required is to generalize the model to *unranked* trees, which have no maximum branching factor. This is highly natural from a mathematical perspective, and brings several added benefits. Selection remains SL, as does the combined grammar for selection and adjunction, even allowing for a degree of variation in the position of adjunction. The model also provides an interesting perspective on the distinction between left and right adjuncts which suggests doubling down on separation between dependency structure and constituency structure, relegating the latter to the post-syntactic map.

The remainder of this paper proceeds as follows. First, I introduce the necessary background on adjunction, MG dependency trees, and strictly local string and tree languages (2). Next, I implement a strictly local grammar for MG dependency trees which includes selection as well as adjunction in the style of Frey and Gärtner (2002) (3). From there, I refine the system to incorporate recursive adjunction (4) and adjunct ordering restrictions (5), building on insights from Graf (2018) and Fowlie (2013). Finally, I address some alternatives and potential complications for the proposed model, and directions for future research (6).

Proceedings of the Society for Computation in Linguistics (SCiL) 2025, pages 133-143.
Eugene, Oregon, July 18-20, 2025

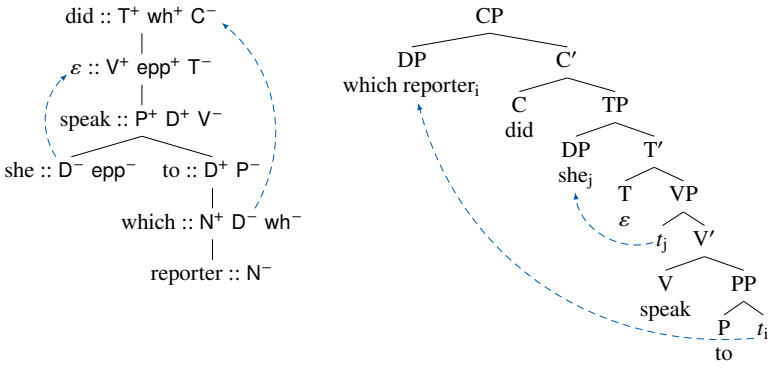


Figure 1: MG dependency tree (left) and phrase structure tree (right) for *Which reporter did she speak to?*

2 Background and model

This section briefly describes the properties of adjunction that we aim to capture, the MG dependency tree model, and SL grammars over strings, ranked trees, and unranked trees. More complex formal languages play no role in the core analysis, though several TSL and M[ulti]TSL string languages appear in 6; see Appendix A for a brief overview and example grammars.

2.1 Properties of adjunction

We are concerned primarily with the following properties of adjunction:

1. **Optionality** – an adjunct may be added or removed without affecting wellformedness
2. **Iteration** – if one adjunct may be added in some context, then any number may be added
3. **Ordering restrictions** – when two or more phrases adjoin to the same head, there may be restrictions on their order
4. **Invisibility for selection** – the properties of a phrase are determined by the those of its head, not those of any adjunct

Some simple examples of adjectival modification are provided below. (1a) demonstrates optionality and iteration: any combination of adjectives denoting size, color, and material can be used, as long as they occur in that order. The remaining examples show that other logically possible orders are degraded.

(1) a. a (big) (blue) (wooden) house
b. ?? a blue big house
c. ?? a wooden big house
d. ?? a wooden blue house

Property #4 is more subtle. Empirically, it means that every phrase represented by (1a) has the same external distribution. Theoretically, it means that the features of the noun 'house' project, not those of the adjectives. This is easily lost in models of adjunction in which the adjective selects the noun, and can interfere with the locality of selection.

There are other properties of adjunction that we might also want to treat, but these four will be our focus, since they are directly related to the subregular complexity of adjunction. In 6.3, we will briefly touch on another structural property: the c-command paradox for right adjuncts.

2.2 MG dependency trees

Here, we briefly outline the MG (Minimalist Grammar) dependency tree model as defined in Graf and Kostyszyn (2021).[1]

In MG (Stabler, 1997, 2011), lexical items pair a phonetic exponent with a string of *features* which control how they may combine in a syntactic derivation. Standard MGs have two types of binary features, controlling the operations Merge and Move. For Merge, we have *selector* features (F^+) and *category* features (F^-). For example, the determiner *the* has features $\mathsf{N}^+\ \mathsf{D}^-$. For Move we have *licensor* features (f^+), marking the landing site of movement, and *licensee* features, marking the head of the mover (f^-). For *wh*-movement, the landing site bears wh^+ and the mover bears wh^-. Additional operations require adding further feature types; we will do this for adjunction momentarily.

MGs generate a language of *derivation trees*, which encode the sequence of operations of Merge/Move/etc. Several variants exist; here we use *dependency trees*, in which all nodes are lexical

[1]The model first appears in Graf and Shafiei (2019). A nearly identical framework can also be found in Kobele (2012).

items (traditional derivation trees will be revisited in 4). Figure 1 shows a dependency tree for a simple sentence with *wh*-movement along with the corresponding phrase structure tree. The daughters of each node are its arguments, ordered by asymmetric c-command (that is, reverse order of selection). Movement is represented only via features; arrows are provided for visual convenience only.

When we say that selection is SL, we mean that licit and illicit arrangements of selector and category features can be distinguished using a SL tree grammar. Importantly, the complexity of selection itself could change if other operations are included in the tree language. We show that this does not occur in the dependency tree model: not only does adding movement not matter (as is well established) but adjunction can safely be added as well.

2.3 SL languages and grammars

SL string languages and grammars are defined in terms of *k-factors*, which are substrings of a string augmented with edge markers. For example, the 3-factors of the string *abc* are:

$$\{\rtimes\rtimes a,\ \rtimes ab,\ abc,\ bc\ltimes,\ c\ltimes\ltimes\}$$

A positive SL-k grammar is a set of *permitted* k-factors, while a negative SL-k grammar is a set of *forbidden* k-factors. Here, we make use of positive grammars (interconversion is always possible). For example, a positive SL-3 grammar consisting of just the above factors would generate the string *abc* and no others. If we add the factors {cab, bca}, then we can also generate *abcabc*, *abcabcabc*, etc. By further adding {abb, bbc}, we can optionally double the *b* to produce *abbc*, *abcabbc*, etc.

A string language is strictly k-local (SL-k) iff it can be described using a positive or negative SL-k grammar. As a regular expression, the language of the above example is $(ab(b)c)^+$. See Rogers et al. (2013) for a formal definition and further context.

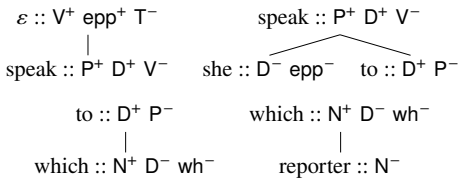
SL languages/grammars are easily extended to *ranked trees*, which have a fixed maximum branching factor. They can be further extended to *unranked trees*, which have no such restriction, by associating each node with an SL string language that constrains its string of daughters.[2] We consider each of these cases in turn.

[2]Such a tree language cannot be implemented with a standard bottom-up deterministic tree automaton (BDTA). Instead, the states of the daughters are processed by a finite state string automaton, and final state of the string automaton is combined with the mother node's label to determine its state. See Comon et al. (2008) for details.

2.4 Ranked trees, selection

Traditionally, regular and subregular tree languages are defined over *ranked trees*, in which each element has a fixed number of daughters, known as its rank. The maximum branching factor of a tree is therefore bounded by the highest ranked element it contains. For such trees, a SL-k tree grammar is just a set of permitted/forbidden subtrees of height k (Rogers, 1997). For the grammar which generated the example in Figure 1, these include the following, among others:

(2) Some permitted subtrees of height 2



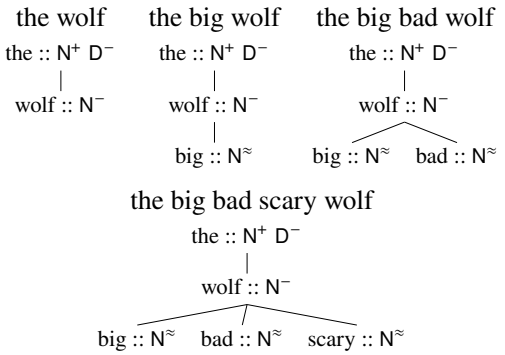
Of course, this can and should be condensed into a format which encodes the relevant generalizations, e.g., every verb with the selector features P^+ D^+ should have exactly two daughters, bearing D^- and P^-, in that order. We will do this in the next section. For now, we note that because the largest portion of the tree we need to examine is of height 2 and the number of possible subtrees is finite, we can list all licit/illicit subtrees, so selection is SL-2.

2.5 Unranked trees, adjunction

We base our system on the work of Frey and Gärtner (2002), who treat adjunction as *asymmetric feature checking*. We add a new class of *adjunction features*, notated $\mathsf{F}^{\approx}$. Modifying adjectives, for example, bear $\mathsf{N}^{\approx}$, since they adjoin to NPs. Adjunction features must be checked against a matching category feature, but the category feature of the head remains unchecked. This contrasts with Merge and Move, whose positive features must be checked against negative features in a one-to-one manner. Adjunction is therefore optional, and may also iterate.

In the MG dependency tree model, it is extremely natural to treat adjuncts as dependents of their heads, preceding all specifiers and complements. This is implicitly assumed by Shafiei and Graf (2020) in their model of adjunct islands, and I do the same in Hanson (2023a) to handle adjunct ordering. However, neither work formalizes this, nor do they treat recursive adjunction. Below are dependency trees for DPs with 0, 1, 2, and 3 NP adjuncts, respectively.

(3) Adjuncts as dependents of the head



There are two key things to notice here. First, the noun and its selector remain adjacent, as does the string of adjuncts and their head. This means that adjunction to XP is invisible to selection of XP by another head Y, as desired. Second, there is no finite bound on the number of daughters of a node. We therefore require *unranked* trees, in which the daughters of a node no longer form a tuple, but a string. Rather than exhaustively listing licit subtrees, the label of each node is mapped to a *daughter string language*, which may be infinite; many examples are given in the following sections. As for the formal implementation, the definition of an MG dependency tree language needs to be adjusted slightly, though we do not do this here.[3]

In the next section, we construct a generalized SL grammar for unranked trees which handles both selection and adjunction, and show that it works for the above structures, among others. In the following sections, we make some minor adjustments in order to incorporate recursive adjunction and adjunct ordering hierarchies.

2.6 Classes of tree grammars

The computational complexity of a tree language need not be uniform in both the vertical and horizontal dimensions. Adapting the terminology of Graf and Kostyszyn (2021), a SL-i[SL-j] tree grammar has a window of i in the vertical dimension and j in the horizontal dimension, the latter corresponding to the daughter string languages. It is also possible to use more a more powerful mechanism in one or both dimensions. For example, the analysis of movement in Graf (2022b) is TSL with a window of 2 in both dimensions, making it TSL-2[TSL-2], while the analysis of case in Hanson (2023b) is MTSL-2[TSL-2], as it involves multiple tree tiers. For present purposes, the window in the vertical dimension will never vary (it is not obvious how a window larger than size 2 would even work), but the window of the daughter string languages may vary depending on the number of arguments. When the window in the horizontal varies by daughter string language, we take the upper bound as representative.

3 Adjunction without non-locality

We begin by constructing a SL grammar which covers selection and adjunction for unranked trees, implementing the system from the previous section. We then augment the system to include recursive adjunction and adjunct ordering restrictions. The approach is closely mirrors the use of TSL tree grammars in Graf (2018) and subsequent work except that the tier projection step, needed only for long-distance dependencies, is omitted.

For now, we make no distinction between left and right adjuncts: their position in the dependency tree represents only their structural (=scopal) position. We present a potential problem with this assumption, as well as a solution, in 6.3.

3.1 Selection

First, consider the case where a node has only arguments or adjuncts among its daughters, but not both. The rules for selection and adjunction in isolation are exceedingly simple, being finite and SL, respectively. We begin with selection.

(4) **Select:** If a node bears the sequence of selector features $\mathsf{X}_1^+, \ldots, \mathsf{X}_n^+$, then its ith daughter from the right must bear category feature X_i^-, for all $1 \leq i \leq n$.[4]

For example, *devour* is an obligatorily transitive verb, with selector features D^+ D^+. Therefore, its daughter string language consists of all strings of length two in which the category of each item is D^-. There is a finite number of selector features on any given lexical item, and the lexicon itself is finite, so the daughter string language of each node is finite, and therefore also strictly local. Specifically, if the number of arguments is n, the daughter string language is SL-$(n+1)$. In the case of *devour*:

[3] The first order constraints in Graf and Kostyszyn (2021) are meant to be combined with an appropriate axiomatization for the class of ranked finite trees; our modified version should be instead be combined with the class of unranked trees. Backofen et al. (1995) provide first-order theories of both ranked and unranked trees which are minimally different and have the desired properties, though infinite trees are not ruled out, as this requires at least monadic second order logic.

[4] Recall that the arguments of a node appear in reverse merge order.

(5) Selection grammar for *devour* (SL-3)
$G^+ = \{\rtimes\rtimes D^-, \rtimes D^- D^-, D^- D^- \ltimes, D^- \ltimes\ltimes\}$

The complete grammar is a map from the label of the mother to the grammar for its daughter string, based only on its selector features. If the maximum number of selector features is n, then in the classification introduced in 2.6, the complexity of the tree grammar is SL-2[SL-$(n+1)$], since we make use of a window which is of height 2 and width $(n+1)$.

3.2 Adjunction

Next, we introduce our adjunction rule.

(6) **Adjoin:** If a node bears category X^-, then it may bear zero or more daughters bearing $X^{\approx}$. No other daughters with adjunction features are allowed.

For example, *wolf* bears N^-, so it may have zero or more daughters bearing $N^{\approx}$. If we map the label of each node to just its adjunction feature, the daughter string language for each category X can be described with the positive grammar $\{\rtimes\ltimes, \rtimes X^{\approx}, X^{\approx}X^{\approx}, X^{\approx}\ltimes\}$, and is therefore SL-2. Since *devour* and most other verbs have at least one argument, we provide a concrete example for *wolf* instead:

(7) Adjunction grammar for *wolf* (SL-2)
$G^+ = \{\rtimes\ltimes, \rtimes N^{\approx}, N^{\approx}N^{\approx}, N^{\approx}\ltimes\}$

As stated, neither of the above rules works for nodes with both arguments and adjuncts among its daughters. Now we combine the two cases.

3.3 Combining the constraints

Recall that we assume all adjuncts to precede all arguments. Therefore, the combined daughter string language template is the concatenation of the two.

(8) **Select + Adjoin:** If a node bears the sequence of selector features $X_1{}^+, \ldots, X_n{}^+$ and category feature Y^-, then its daughter string consists of zero or more daughters bearing $Y^{\approx}$ followed by n daughters bearing category feature $X_i{}^-$, from right to left, for all $1 \leq i \leq n$.

SL languages are not in general closed under concatenation, so we must show that concatenation is possible in this case. Specifically, we show that the combined daughter string language schema has a factor width equal to the higher of the two source grammars: if n is the maximum number of selector features, then the combined grammar is SL-2[SL-k], where k is the greater of $\{2, (n+1)\}$.

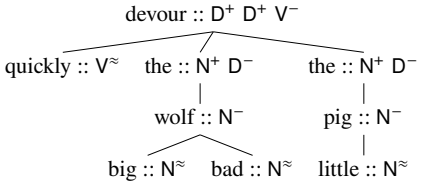
The construction is as follows. First, we convert the SL-2 adjunction grammar to SL-$(n+1)$ by padding its factors, and also remove any factors that allow a string to end without any arguments. Second, we add these to the factors of the selection grammar. Finally, we add any factors needed to transition from an adjunct to the highest argument.

A concrete example for *devour* is shown below. As before, we map each node label to just its adjunction or category feature for brevity.

(9) Combined grammar for *devour* (SL-3)
$G^+ = \{\rtimes\rtimes D^-, \rtimes D^- D^-, D^- D^- \ltimes, D^- \ltimes\ltimes, \rtimes\rtimes V^{\approx}, \rtimes V^{\approx} V^{\approx}, V^{\approx} V^{\approx} V^{\approx}, \rtimes V^{\approx} D^-, V^{\approx} V^{\approx} D^-, V^{\approx} D^- D^-\}$

Let us apply this grammar to the node *devour* in the dependency tree for the sentence *The big bad wolf quickly devoured the little pig*, shown below. For simplicity, we truncate the tree at the VP level and omit movement features. The reader may confirm that all 3-factors of the daughters of *devour* are licit. To ensure that the entire tree is licit, we repeat this procedure for every node.

(10) a. Dependency tree:



b. DS of *devour*: $V^{\approx}\ D^-\ D^-$

c. 3-factors of DS: $\{\rtimes\rtimes V^{\approx}, \rtimes V^{\approx} D^-, V^{\approx} D^- D^-, D^- D^- \ltimes, D^- \ltimes\ltimes\}$

The construction is essentially identical for items with three or more arguments. For those with just one, the selection grammar is already SL-2, so no padding of the adjunction factors is required. For items with no arguments (including the verb *rain* and many nouns), we are back to the plain adjunction grammar, which remains SL-2.

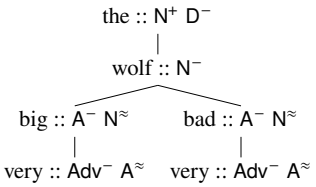
To briefly review, we achieved a combined SL model of selection and adjunction over unranked trees, whose grammar is a mapping from node labels to daughter string languages, each of which is SL, for a combined complexity of SL-2[SL-k], with $k \geq 2$. Now, we introduce recursive adjunction.

4 Recursive adjunction

We follow the lead of Graf (2018) by reintroducing category features on adjuncts. For example, modifying adjectives carry $A^-\ N^{\approx}$, and adverbs carry

either $\text{Adv}^- \text{A}^{\approx}$ or $\text{Adv}^- \text{V}^{\approx}$. Below is an example adverbial modification of adjectives, which in turn modify a noun.

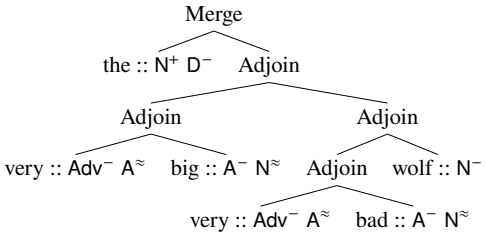
(11) the very big very bad wolf



Locality is clearly preserved in the dependency tree model, since adding an adverb under an adjective does not interrupt adjacency between the adjective and the head noun, just as adding an adjective below a noun does not affect the relation with the selecting determiner. Furthermore, although some category features are no longer checked with a corresponding selector feature, this can be determined just from the label of the node in question, so we do not even need to change the SL tree grammar. We continue to distinguish items with and without a final adjunction feature, just as before.

At this point, I should briefly describe the problem that occurs with recursive adjunction in traditional MG derivation trees. In this system, internal nodes represent the Merge/Move/Adjoin operations, and all leaves are lexical items. The derivation tree for the current example is shown below.

(12) the very big very bad wolf



Here, an adjunct and its head are not necessarily adjacent, and the distance grows without bound if the adjunct itself serves an an adjunction site. As a consequence, adjunction is not SL for any window size. Furthermore, selection is not SL either, since the distance between the D head and the N head grows without bound as adjuncts are added. If not for recursive adjunction, strict locality of selection could be rescued via a chain analysis (e.g. D licenses A, which licenses A, which licenses N). But with recursive adjunction, the intervening A heads themselves are not guaranteed to lie within any finite window.

According to Graf (2018), Merge and Move are *structure-sensitive TSL* (SS-TSL) over derivation trees (see De Santo and Graf 2019 for the string case); the complexity of Adjoin is left open, though it is clearly not SL. Several phonological phenomena are SS-TSL over strings (Graf and Mayer, 2018; Mayer and Major, 2018), so this is not a catastrophe. Additionally, as Graf notes, it would mean that Merge and Move are extremely closely related in formal terms, mirroring the view in Chomsky (2004). However, as the evidence accumulates that SL and TSL are sufficient for most syntactic phenomena under the dependency tree model (Graf, 2022b; Hanson, 2023b, 2025), one gets the impression that the need for SS-TSL is an artifact of the derivation tree representation.

This requires elaboration since, as a reviewer remarks, there is an inherent trade-off between representational and computational complexity, such that one can often be reduced by increasing the other. In this case, the information in each representation is comparable, with sister order in the dependency replacing the extra nodes of the derivation tree, but the computational complexity of the former is lower. Furthermore, the range of patterns which SL/TSL can produce have wide empirical support, while SS-TSL serves primarily to factor out the extra nodes of the derivation tree. An exception can be found in Principle B of the binding theory, which seems to require SS-TSL (Graf and Shafiei, 2019), mirroring the occasional SS-TSL pattern in phonology, but this does not seem to be needed for most operations. In summary, the dependency tree model allows us to minimize the overall complexity of the system while also providing the best fit to the known typology.

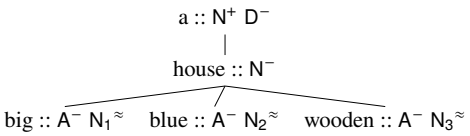
5 Adjunct ordering restrictions

The adjustment we made for recursive adjunction also lays the groundwork for encoding adjunct ordering restrictions. The basic insight by Fowlie (2013) is that a principled treatment of adjunction ordering requires a *pair of features* rather than a single adjunction feature. By tracking both the position in the hierarchy and the adjunction target simultaneously, we can avoid resorting to low level tricks such as adding unmotivated empty categories or exploding the lexicon.

Rather than implementing her exact system, we make use of the pairing of category and adjunction features already in play. Specifically, we split

our adjunction features by adding an index corresponding to the position the item must take in the relevant ordering hierarchy. The primary difference between our approach and Fowlie's is that while she uses paired features primarily to label adjunction nodes in the derivation tree, we label the adjuncts themselves. Consider again the example from (1), repeated below with its dependency tree.

(13) a big blue wooden house



In this case, we included only three indices, but we can include as many as we need as long as the number of positions is finite. In the style of the preceding examples, the rule is as follows:

(14) **Ordered adjunction:** If a node bears category feature X which has n positions in its adjunction hierarchy, then any pair of daughters d_i, d_j bearing $X_i^{\approx}$ and $X_j^{\approx}$, where $1 \leq i < j \leq n$, must be ordered such that d_i precedes d_j.

As discussed by Hanson (2023a), ordered adjunction is SL-2, just like simple adjunction, even allowing for iteration, e.g. *the big big big blue house*. Viewed as a finite state automaton, the daughter string language is just the reflexive transitive closure of the order of adjunction categories. Rather than clutter the above definition, we proceed directly to the template which covers all cases:

(15) Adjunction grammar for category N (SL-2)
$G^+ = \{\rtimes\ltimes,\ \rtimes N_1^{\approx},\ \rtimes N_2^{\approx},\ \rtimes N_3^{\approx},\ N_1^{\approx}N_2^{\approx},\ N_2^{\approx}N_3^{\approx},\ N_1^{\approx}N_1^{\approx},\ N_2^{\approx}N_2^{\approx},\ N_3^{\approx}N_3^{\approx},\ N_1^{\approx}\ltimes,\ N_2^{\approx}\ltimes,\ N_3^{\approx}\ltimes\}$

This grammar can then be combined with the selection grammar as before.

It is natural to ask whether it might be better to split the category feature of the adjunct rather than the adjunction feature. This would also presumably work, and would in fact be more faithful to Fowlie's system. One small downside is that increases the size of the lexicon somewhat. For example, if we split category A into S(ize)/C(olor)/M(aterial)/etc. then predicational use of adjectives will require duplicate lexical entries for all selecting heads (*be*, *seem*, etc.). The same is true of adjective modifiers such as *very*. While Fowlie presents some potential solutions, the present approach sidesteps these problems altogether, as the effects of the split are isolated to just the context where they are desired. We further discuss this alternative in Section 6.4.

6 Extensions and alternatives

At this point, we have achieved what we set out to do: we have constructed a simple SL model of adjunction which handles all of the properties specified at the outset, and which avoids non-locality in cases of recursive adjunction. Now, we address some other issues which have not been our focus, some possible extensions of the current system, and how some other systems compare. For brevity, some of the string languages in this section are defined using regular expressions, with the grammars relegated to Appendix A.

6.1 Ordered and unordered adjuncts

So far, we have mostly ignored adjuncts without ordering hierarchies, which traditionally include PPs. To a certain degree, there is not much to say about them since, if they are indeed unordered with respect to each other, then the simple adjunction grammar from Section 3 will do the job. The fact that they are linearized to the right in English can be seen as a part of the mapping to the surface, independent of the dependency tree.

However, there is potential danger to the SL analysis when we consider both ordered and unordered adjuncts together. Suppose for the sake of argument that PPs can be interspersed among adjectives or adverbs (as determined by scope), and that they can also iterate in each position. This would yield a daughter string language along the following lines:

(16) Ordered APs and unordered PPs
$P^* \cdot A_1^* \cdot P^* \cdot A_2^* \cdot P^* \cdot A_3^* \cdot P^*$

Such a language is not SL, since we need adjacent items in the adjective hierarchy to appear in the same window yet there is no limit to the number of P heads which may intervene. It is uncertain whether this scenario is actually realistic, but if so, then the daughter string languages for selection and each type of adjunction become TSL, and the combined language would be Multi-TSL (MTSL; see De Santo and Graf 2019), since the tiers for each would be different. Even if such constructions exist, it could be that left and right adjuncts are not actually interspersed in the dependency tree, in which the daughter string language remains SL-2. We consider this possibility in Section 6.3.

6.2 The position of adjunction

In the above analysis, we assumed that all adjuncts precede all arguments in the derivation tree, which is equivalent to the assumption that all adjunction occurs at the XP level. It is also conceivable that adjuncts could occur in other positions. For example, Frey and Gärtner (2002) assume that manner adverbs attach to the verb before the object does in their analysis of German.

We should therefore consider the possibility that the position of adjunction features within the MG feature string may vary. Indeed, one could make the argument that the SL model predicts such variation. In the example just cited, all manner adverbs follow the complement, which is an easy change. We might also ask whether there exist any systems which are not SL. For example, consider a hypothetical language in which PP adjuncts can be inserted freely in any position, similar to (16):

(17) Hypothetical non-SL version of *devour*
$P^* \cdot D \cdot P^* \cdot D \cdot P^*$

This particular example would again be MTSL. If adjunction is SL, then such adjunction paradigms should not exist, even if some other variants do.

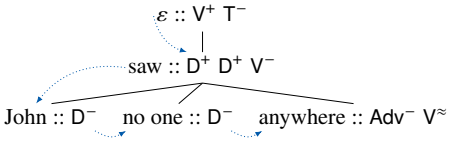
6.3 Left vs. right adjuncts

Right adjuncts in English are unordered, with constituency and scope diagnostics suggesting that the outer adjuncts are higher, but c-command diagnostics such as NPI licensing go the other way.

(18) a. John saw [no one] [anywhere].
b. * John saw [anyone] [nowhere].
(Ernst, 1994)

In previous work (Graf and Shafiei, 2019; Hanson, 2025), a relation called *d[erivational]-command*, which combines the dominance and left-sister relations of the dependency tree, serves as the analog of c-command in the phrase structure tree. The NPI data can therefore be accommodated if we assume that right adjuncts appear *after* all arguments in the dependency tree.

(19) Abbreviated dependency tree for (18a), showing d-command relations



In doing so, we affirm the idea that sister order encodes command at the expense of losing a direct correspondence to constituency and scope. These would instead need to be introduced in the mapping from the dependency tree to the corresponding phrase structure tree. Such an approach would be reminiscent of the dual model of 'cascade syntax' and 'layered syntax' in Pesetsky (1996). I leave the exploration of this possibility to future work.

6.4 Adjunct subcategories

As mentioned in Section 5, the closest alternative to the proposed approach to adjunct ordering—splitting the category rather than the adjunction feature—introduces some lexical redundancy independent of that introduced by the inclusion of adjunction features. But perhaps we could do away with adjunction features entirely and rely on the local context to identify adjunction, as in Fowlie (2013). The structure of the daughter string language would be essentially identical, just with $N_1^{\approx}/N_2^{\approx}/N_3^{\approx}$ substituted by $S^-/C^-/M^-$, and so on. This has been done for example (15) below:

(20) Adjunction grammar for category N (SL-2)
$G^+ = \{\rtimes\ltimes, \rtimes S^-, \rtimes C^-, \rtimes M^-, S^-C^-, C^-M^-, S^-S^-, C^-C^-, M^-M^-, S^-\ltimes, C^-\ltimes, M^-\ltimes\}$

Aside from creating some lexical redundancy in the selectors of S/C/M/etc., a major disadvantage of such a model from a subregular perspective is that arguments and adjuncts of the same category can no longer be easily distinguished for long-distance operations such as movement, as sisterhood in the dependency tree is not preserved by projection to a tree tier. Arguments and adjuncts are usually thought to differ in their behavior with respect to movement (both as movers and as containers for movers), casting doubt on the viability of such an approach, though see 6.6 for a counterargument.

6.5 Selectional approaches

As noted by Fowlie (2013), models that attempt to reduce adjunction to selection suffer from various formal and linguistic shortcomings, particularly in accounting for ordering hierarchies. For example, we could implement a functional sequence, e.g. D < S < C < M < N, by including empty elements to fill the unused slots. Each modifier needs a single lexical entry, but the empty items have no independent morphological or semantic motivation and are therefore "nothing more than a trick to hold the syntax together" (Fowlie, 2013, p. 16).

(21) Functional sequence

wooden :: N^+ M^-	ε :: N^+ M^-
blue :: M^+ C^-	ε :: M^+ C^-
big :: C^+ S^-	ε :: C^+ S^-
the :: S^+ D^-	

Conversely, we can avoid empty heads by means of lexical homophony, but the lexical redundancy factor is far worse than other alternatives, on the order of n^2 (ibid.). Even if we dismiss the increased memory burden, the pattern feels particularly accidental when analyzed in this way, as there is nothing which prevents items of the same category from selecting a different set of 'next' elements.

(22) Massive homophony

wooden $\rightarrow$ N^+ M^-
blue $\rightarrow$ N^+ C^- / M^+ C^-
big $\rightarrow$ N^+ S^- / M^+ S^- / C^+ S^-
the $\rightarrow$ N^+ D^- / M^+ D^- / C^+ D^- / S^+ D^-

A third alternative, not considered by Fowlie, utilizes 'adjunctizer' heads which introduce the adjunct itself as a specifier. This contains the scope of redundancy to a small subset of the lexicon, but then we are back to the problem of unmotivated empty elements.

(23) Adjunctizer heads

M-ADJ $\rightarrow$ N^+ M^+ M^-
C-ADJ $\rightarrow$ N^+ C^+ C^- / M^+ C^+ C^-
S-ADJ $\rightarrow$ N^+ S^+ S^- / M^+ S^+ S^- / C^+ S^+ S^-
the $\rightarrow$ N^+ D^- / M^+ D^- / C^+ D^- / S^+ D^-

In each case, the difficulty of distinguishing arguments and adjuncts which we noted earlier still applies. Overall, it seems to be preferable to keep adjunction as a distinct operation, and factor out ordering restrictions into the SL grammar.

6.6 Additional puzzles

Throughout this paper, I have assumed that the given generalizations about adjunction are actually correct, but various exceptions have long been known. For example, as a reviewer notes, violations of the adjective order in cases of recursive adjunction seem less bad compared to simple adjunction.

(24) a. a big blue house
b. ?? a blue big house

(25) a. a very big very blue house
b. ? a very blue very big house

As discussed by Hanson (2023a), there are various ways in which adjunct orders are more fluid than is often supposed; in languages such as German, they seem not to exist (Thomas Graf, p.c.). It is therefore not clear that they should even be modeled in the syntactic grammar. For present purposes, the crucial point is that if we decide to do so, they remain within the power of SL.

Similarly, I have taken for granted that the argument-adjunct distinction exists and must be accounted for. This might also not be so clear cut: a reviewer cites McInnerney (2022), who argues that the distinction is not well-supported on syntactic or semantic grounds. This seems compatible with the central claim of this paper, since selection and adjunction are SL both in isolation and in combination. It would be only a small step to eliminate the distinction entirely, with the caveats discussed in 6.5. That said, given that the study by McInnerney focuses almost exclusively on PPs, further investigation is needed to determine whether the same arguments apply to adjectives and adverbs.

7 Conclusion

I have shown that a linguistically appealing model of adjunction based on a pairing of category and adjunction features is SL over MG dependency trees, inclusive of formal challenges such as recursive adjunction and adjunct ordering restrictions. Selection and adjunction can be combined into a single SL daughter string language, and beyond this, certain variants such as low manner adverb attachment and the distinction between left and right adjuncts may be accommodated.

Overall, these results support the classification of adjunction as a local phenomenon. If it is determined that the interspersing of ordered and unordered adjuncts in the dependency tree cannot be avoided, then the combined complexity of selection and adjunction increases to SL-2[MTSL-k]. Now that most major syntactic operations (selection, adjunction, movement, case, agreement, binding) have been studied in isolation, the next step is to determine to what extent the interactions between them can be handled within the bounds of the (M)TSL tree languages.

Acknowledgments

This work was partly supported by the Institute for Advanced Computational Science at Stony Brook University. My thanks to Thomas Graf and three anonymous reviewers for their detailed feedback on earlier versions of this paper.

References

Rolf Backofen, James Rogers, and K. Vijay-Shanker. 1995. A first-order axiomatization of the theory of finite trees. *Journal of Logic, Language and Information*, 4(1):5–39.

Noam Chomsky. 2004. Beyond explanatory adequacy. In Adriana Belletti, editor, *Structures and Beyond*, pages 104–131. Oxford University Press, New York, NY.

Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. 2008. Tree automata techniques and applications. Online.

Aniello De Santo and Thomas Graf. 2019. Structure sensitive tier projection: Applications and formal properties. In *Formal Grammar: 24th International Conference*, pages 35–50, Riga, Latvia.

Thomas Ernst. 1994. M-command and precedence. *Linguistic Inquiry*, 25(2):327–335.

Meaghan Fowlie. 2013. Order and optionality: Minimalist Grammars with adjunction. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pages 12–20, Sofia, Bulgaria.

Werner Frey and Hans-Martin Gärtner. 2002. On the treatment of scrambling and adjunction in minimalist grammars. In *Proceedings of Formal Grammar 2002*, pages 41–52, Trento, Italy.

Thomas Graf. 2014. Models of adjunction in minimalist grammars. In *Formal Grammar: 19th International Conference*, pages 52–68, Tübingen, Germany.

Thomas Graf. 2018. Why movement comes for free once you have adjunction. In *Proceedings of CLS 53*, pages 117–136, Chicago, IL.

Thomas Graf. 2022a. Subregular linguistics: bridging theoretical linguistics and formal grammar. *Theoretical Linguistics*, 48(3–4):145–184.

Thomas Graf. 2022b. Typological implications of tier-based strictly local movement. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2022*, pages 184–193, Online.

Thomas Graf and Kalina Kostyszyn. 2021. Multiple wh-movement is not special: The subregular complexity of persistent features in Minimalist Grammars. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2021*, pages 275–285, Online.

Thomas Graf and Connor Mayer. 2018. Sanskrit n-retroflexion is input-output tier-based strictly local. In *Proceedings of SIGMORPHON 2018*, pages 151–160, Brussels.

Thomas Graf and Nazila Shafiei. 2019. C-command dependencies as TSL string constraints. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pages 205–215, New York, NY.

Kenneth Hanson. 2023a. Strict locality in syntax. In *Proceedings of CLS 59*, pages 131–145, Chicago, IL.

Kenneth Hanson. 2023b. A TSL analysis of Japanese case. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2023*, pages 15–24, Amherst, MA.

Kenneth Hanson. 2025. Tier-based strict locality and the typology of agreement. To appear in *Journal of Language Modelling*.

Jeffrey Heinz. 2018. The computational nature of phonological generalizations. In Larry M. Hyman and Frans Plank, editors, *Phonological Typology*, pages 126–195. De Gruyter Mouton.

Jeffrey Heinz, Chetan Rawal, and Herbert G. Tanner. 2011. Tier-based strictly local constraints for phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 58–64, Portland, OR.

Gregory M. Kobele. 2012. Eliding the derivation: A minimalist formalization of ellipsis. In *Proceedings of the 19th International Conference on Head-Driven Phrase Structure Grammar*, Chungnam National University Daejeon.

Dakotah Lambert and James Rogers. 2020. Tier-based strictly local stringsets: Perspectives from model and automata theory. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2020*, pages 159–166, New Orleans, LA.

Connor Mayer and Travis Major. 2018. A challenge for tier-based strict locality from uyghur backness harmony. In *Formal Grammar 2018: 23rd International Conference*, pages 62–83, Sofia, Bulgaria.

Andrew McInnerney. 2022. *The Argument/Adjunct Distinction and the Structure of Prepositional Phrases*. Ph.D. thesis, University of Michigan, Ann Arbor, MI.

David Pesetsky. 1996. *Zero syntax: Experiencers and cascades*. MIT Press, Cambridge, MA.

James Rogers. 1997. Strict LT_2 : Regular :: Local : Recognizable. In *Logical Aspects of Computational Linguistics: First International Conference, LACL '96 (Selected Papers)*, pages 366–385, Nancy, France.

James Rogers, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. 2013. Cognitive and sub-regular complexity. In *Formal Grammar: 17th and 18th International Conferences*, pages 90–108.

Nazila Shafiei and Thomas Graf. 2020. The subregular complexity of syntactic islands. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2020*, pages 421–430, New Orleans, LA.

Edward P. Stabler. 1997. Derivational minimalism. In *Logical Aspects of Computational Linguistics: First International Conference, LACL '96 (Selected Papers)*, pages 68–95, Nancy, France.

Edward P. Stabler. 2011. Computational perspectives on Minimalism. In Cedric Boeckx, editor, *Oxford handbook of linguistic Minimalism*, pages 617–643. Oxford University Press, Oxford, UK.

Mai Ha Vu, Nazila Shafiei, and Thomas Graf. 2019. Case assignment in TSL syntax: A case study. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pages 267–276, New York, NY.

A Additional adjunction grammars

Section 6 provided examples of daughter string languages for several hypothetical adjunction patterns, not all of which are SL. Very briefly, a TSL language is one in which certain elements are ignored, forming a *tier projection*. Which elements appear on the tier is determined completely by their labels, and those that do are treated as if adjacent, subject to a SL grammar. See Heinz et al. (2011); Lambert and Rogers (2020) for details. An MTSL grammar is just the intersection of several TSL grammars (De Santo and Graf, 2019).

A.1 Ordered and unordered adjuncts

First, the hypothetical language from 6.1, which freely intersperses ordered AP adjuncts and unordered PP adjuncts, is repeated below. The constraints of the grammars are unchanged from our earlier SL grammars. The only difference is that tier projection is used to ignore adjuncts of the opposite type. For simplicity, I use mnemonic labels rather than MG feature specifications.

(26) Ordered APs and unordered PPs (MTSL-2)

a. Language:
$P^* \cdot A_1^* \cdot P^* \cdot A_2^* \cdot P^* \cdot A_3^* \cdot P^*$

b. AP adjunction grammar (TSL-2)
$T = \{A_1, A_2, A_3\}$
$G^+ = \left\{ \begin{array}{c} \rtimes\ltimes,\ \rtimes A_1,\ \rtimes A_2,\ \rtimes A_3, \\ A_1A_2,\ A_2A_3, \\ A_1A_1,\ A_2A_2,\ A_3A_3, \\ A_1\ltimes,\ A_2\ltimes,\ A_3\ltimes \end{array} \right\}$

c. PP adjunction grammar (TSL-2)
$T = \{P\}$
$G^+ = \{\rtimes\ltimes, \rtimes P, PP, P\ltimes\}$

A.2 Unordered adjunction everywhere

If unordered adjuncts can be freely interspersed with arguments, the result is MTSL, similar to free mixing of ordered and unordered adjuncts. In Section 6.2, I predicted that this should not occur.

(27) Unordered adjunction + selection (MTSL-3)

a. Language:
$P^* \cdot D \cdot P^* \cdot D \cdot P^*$

b. Selection grammar (TSL-3)
$T = \{D\}$
$G^+ = \{\rtimes\rtimes D, \rtimes DD, DD\ltimes, D\ltimes\ltimes\}$

c. Adjunction grammar (TSL-2)
$T = \{P\}$
$G^+ = \{\rtimes\ltimes, \rtimes P, PP, P\ltimes\}$

A.3 Low adjunction

The proposed daughter string language and grammar proposed for low manner adverbs as described in 6.2 is given below. Unlike the previous grammars, this one remains SL.

(28) Low adjunction equivalent of *devour* (SL-3)

a. Language:
$D \cdot D \cdot Adv^*$

b. Grammar:
$G^+ = \left\{ \begin{array}{c} \rtimes\rtimes D,\ \rtimes DD,\ DD\ltimes, \\ D\ltimes\ltimes,\ D\,D\,Adv,\ D\,Adv\,Adv, \\ Adv\,Adv\,Adv,\ D\,Adv\ltimes, \\ Adv\,Adv\ltimes,\ Adv\ltimes\ltimes \end{array} \right\}$

This could be further generalized to allow different types of adjuncts in different positions as long as they can be distinguished from one another, as shown below.

A.4 Left and right adjuncts

I noted in 6.3 that a grammar with left adjuncts at the beginning and right adjuncts at the end would be SL, as long as distinct indices are used. In this case, we can safely allow ordered adverbs on the left and unordered adverbs and PPs on the right.

For simplicity, I assume a single index R for right adjuncts, and I do not pad the 2-factors to 3-factors as is technically required (such a factor should be interpreted as standing in for any 3-factor that contains it as a substring). Effectively, we combine the grammars from (9) and (28).

(29) Left and right adjunction (SL-3)

a. Language:
$Adv_1^* \cdot Adv_2^* \cdot Adv_2^* \cdot D \cdot D \cdot Adv_R^*$

b. Grammar:
$G^+ = \left\{ \begin{array}{c} \rtimes\ltimes,\ \rtimes A_1,\ \rtimes A_2,\ \rtimes A_3, \\ A_1A_2,\ A_2A_3, \\ A_1A_1,\ A_2A_2,\ A_3A_3, \\ A_1D,\ A_2D,\ A_3D, \\ \rtimes\rtimes D,\ \rtimes DD,\ DD\ltimes, D\ltimes\ltimes, \\ DA_R, A_RA_R, A_R\ltimes \end{array} \right\}$