Tiers, Paths, and Syntactic Locality: The View from Learning

Kenneth Hanson
Department of Linguistics
Stony Brook University
Stony Brook, NY 11794, USA
kenneth.hanson@stonybrook.edu

Abstract

Many long-distance linguistic dependencies across domains can be modeled as *tier-based strictly local* (TSL) patterns (Graf, 2022a). Such patterns are in principle efficiently learnable, but known algorithms require unrealistic conditions. In contrast, Heuser et al. (2024) present an empirically-grounded algorithm which learns syntactic islands by tracking bigrams along movement paths, but does not involve tiers. I combine the advantages of both approaches by adapting the latter algorithm to produce a TSL grammar. This method is capable of learning other syntactic blockers besides islands, and augments the typological predictions of the TSL model with a version of the Height-Locality Connection (Keine, 2019).

1 Introduction

The *tier-based strictly local* (TSL) languages are a restrictive class of subregular languages over strings or trees which model a wide range of long-distance linguistic dependencies, from consonant and vowel harmony to movement and case licensing (cf. Heinz, 2018; Graf, 2022a). Elements which are irrelevant to a given dependency are treated as invisible, and those remaining are treated as adjacent, forming a structure called a *tier*. From this perspective, a syntactic or phonotactic grammar consists of many intersecting TSL patterns with different tiers. For syntax, these include tiers for *wh*-movement, EPP-movement, ϕ-agreement, etc., plus a tier including all elements to regulate local dependencies.

Generally speaking, linguistic dependencies are subject to various blocking effects, including locality restrictions such as the lack of raising out of finite clauses in English (known as *hyperraising*) as well as the well-known island constraints (see Belletti 2018 for an overview). Exactly which elements block which dependencies varies somewhat across languages, though there are some general tendencies (Keine, 2019). Roughly speaking, it is assumed in the TSL model that dependent elements must be adjacent on a tier; if any other elements intervene on the tier then blocking effects result. Thus, variation in blockers across languages and phenomena equates to differences in the relevant set of tier elements. For example, all C heads appear on the EPP-movement tier in English, but not in a language which allows hyperraising (Graf, 2022b).

While this parameter of the model allows good empirical coverage, it also presents a learning difficulty due to the large number of logically possible tiers, which grows exponentially with the number of elements (segments or syntactic heads). There exist efficient algorithms for learning TSL string patterns, but they either require the tier elements to be fixed in advance (Lambert et al., 2021) or they are not robust against interaction with other constraints (Jardine and McMullin, 2017; Lambert, 2021). The problem is particularly acute for syntax, for even if we can reduce the problem to learning of TSL string languages, the number of tiers and the size of a syntactic lexicon make exhaustive search completely impractical.

A solution may be found by looking to empirically-motivated models of child language acquisition. Heuser et al. (2024) present a model for learning island constraints which constructs a grammar of local bigrams from attested movement paths, supplemented by generalization by the Tolerance Principle (Yang, 2016). They also show that this model makes correct generalizations based on a realistic input distribution. This approach is interesting in that it circumvents the difficulties of tier detection, but only because it lacks tiers altogether: the resulting grammar is *strictly local* (SL) rather than TSL. This brings several limitations, particularly that it can only recognize movement paths which have been delimited in advance.

Ideally, we would like to combine the generality and typological success of the TSL model with an efficient, linguistically-motivated learning algorithm such as that in Heuser et al. (2024). Towards this end, I adapt their algorithm to produce TSL grammars as used in subregular syntax. I also draw attention to several linguistically interesting aspects of the model, which derives a version of the Height-Locality Connection—the observation that higher categories in the clausal spine are subject to fewer locality restrictions—similar to that given in Keine (2019). It is also equally applicable to other pairwise dependencies such as agreement.

The remainder of this paper is laid out as follows. §2 presents a model of syntactic dependencies based on *ancestor strings* (Shafiei and Graf, 2020), whose grammars will be our learning target. §3 adapts the algorithm from Heuser et al. (2024) to the subregular framework, and §4 modifies it to produce a TSL grammar. §5 shows how this model derives a version of the Height-Locality Connection. §6 concludes.

2 Subregular syntax with ancestor strings

This section introduces the class of TSL string languages along with a model of syntactic dependencies based on *ancestor strings* (a-strings, Shafiei and Graf 2020). We begin with the more restrictive class of *strictly local* or SL languages, which model local linguistic dependencies, before moving on to the TSL languages. Examples of string-like constraints from syntax are provided. From there, we discuss the syntactic framework which provides the relevant strings, and the limits of this model.

2.1 Strictly local languages

Many classes of subregular languages, including SL and TSL, are defined in terms of k-factors, which for these classes are substrings, i.e. discrete k-grams. The definitions here follow Mayer (2021).

Let Σ be a fixed alphabet, let s be a string over Σ^*, and let $\rtimes, \ltimes \notin \Sigma$ be the left and right edge markers. The set $f_k(s)$, the k-factors of s, consists of all the length-k substrings of $\rtimes^{k-1} s \ltimes^{k-1}$ where $k \geq 1$. For example, $f_2(ababac) = \{\rtimes a, ab, ba, ac, c\ltimes\}$.

An SL grammar is just a set of forbidden k-factors of fixed width, and its language consists of all strings which do not contain any of these k-factors. Formally:

Definition 1 A *strictly k-local* (SL-k) grammar is a set $G \subseteq (\Sigma \cup \{\rtimes, \ltimes\})^k$. A language $L \subseteq \Sigma^*$ is SL-k iff there exists an SL-k grammar G such that $L = \{s \in \Sigma^* : f_k(s) \cap G = \varnothing\}$.

Alternatively, an SL-k grammar can be defined in terms of permitted k-factors. A set of forbidden factors is a *negative* grammar; its complement, the set of permitted factors, is a *positive* grammar. There are circumstances where either form may be more convenient. When necessary, these will be disambiguated using a superscript: G^+ for a positive grammar and G^- for a negative grammar.

Example 1 Consider the hierarchy of functional categories in a typical English clause. In the sentence *The pizza has been eaten*, it consists of the sequence of categories $\mathsf{T} \cdot \mathsf{Perf} \cdot \mathsf{Prog} \cdot v$. Let us assume that the general form of the hierarchy is

$$\text{T} > (\text{Perf}) > (\text{Prog}) > (\text{Pass}) > v$$

where categories in parentheses are optional.

The set of licit sequences in a functional hierarchy can be encoded using an SL-2 grammar. Though modeled as a string, in the syntactic framework to be developed in §2.3, it represents a path through part of the tree. The positive grammar is as follows (ignoring edge markers for simplicity):

$$G^+ = \left\{ \begin{array}{llll} \mathsf{T\ Perf}, & & & \\ \mathsf{T\ Prog}, & \mathsf{Perf\ Prog}, & & \\ \mathsf{T\ Pass}, & \mathsf{Perf\ Pass}, & \mathsf{Prog\ Pass}, & \\ \mathsf{T}\ v, & \mathsf{Perf}\ v, & \mathsf{Prog}\ v, & \mathsf{Pass}\ v \end{array} \right\}$$

The corresponding negative grammar is:

$$G^- = \left\{ \begin{array}{lllll} \mathsf{T\ T}, & \mathsf{Perf\ T}, & \mathsf{Prog\ T}, & \mathsf{Pass\ T}, & v\ \mathsf{T}, \\ & \mathsf{Perf\ Perf}, & \mathsf{Prog\ Perf}, & \mathsf{Pass\ Perf}, & v\ \mathsf{Perf}, \\ & & \mathsf{Prog\ Prog}, & \mathsf{Pass\ Prog}, & v\ \mathsf{Prog}, \\ & & & \mathsf{Pass\ Pass}, & v\ \mathsf{Pass}, \\ & & & & v\ v \end{array} \right\}$$

Every 2-factor in our example string appears only in the positive grammar.[1] ⌟

2.2 Tier-based strictly local languages

A TSL language is similar to an SL language except that certain symbols are ignored. Let $T \subseteq \Sigma$ be a *tier alphabet*. The string $\pi_T(s)$ is the *tier projection* of s, the result of deleting all σ in s such that $\sigma \notin T$, and concatenating those that remain. For example, if $\Sigma = \{x, a, b, c\}$ and $T = \{a, b, c\}$ then $\pi_T(axxbxxc) = \pi_T(xxxabcxxx) = abc$.

Definition 2 A *tier-based strictly k-local* (TSL-k) grammar is a tuple (T, G), where T is a tier alphabet and G is an SL-k grammar over T. A language $L \subseteq \Sigma^*$ is TSL-k iff there exists a TSL-k grammar such that $L = \{s \in \Sigma^* : f_k(\pi_T(s)) \cap G = \varnothing\}$.

[1]It is not necessary for every functional head to always be present. If syntax includes SL computations then it can implement functional hierarchies just as easily as category selection. See Hanson (2023) for details.

By definition, all symbols not in T may be freely inserted and deleted without affecting the well-formedness of a given string w.r.t. a given TSL grammar, a fact that will be important to the discussion of tier identification in §4.2.

Example 2 DP subjects in English are thought to move to Spec-TP, whether from inside *v*P or an embedded non-finite TP (the raising construction); they cannot move from a finite CP (hyperraising). Examples are given in (1) below. This dependency—call it EPP-movement—can be encoded with a TSL-2 grammar which requires the mover and landing site (marked with an "EPP" subscript) to be adjacent on a tier. In anticipation of the syntactic framework to be developed, we model this dependency with a string which encodes each head along the movement path, projecting a tier that contains only the relevant elements (movers, landing sites, and blockers).[2]

(1) a. We [$_{v\mathrm{P}}$ ___ have a problem].
Path: $\rtimes \cdot \mathbf{D}_{\mathbf{EPP}} \cdot v \cdot \mathbf{T}_{\mathbf{EPP}} \cdot \ltimes$
Tier: $\rtimes \cdot \mathbf{D}_{\mathbf{EPP}} \cdot \mathbf{T}_{\mathbf{EPP}} \cdot \ltimes$

b. We seem [$_{\mathrm{TP}}$ to ___ have a problem].
Path: $\rtimes \cdot \mathbf{D}_{\mathbf{EPP}} \cdot v \cdot \mathsf{T} \cdot \mathsf{V} \cdot v \cdot \mathbf{T}_{\mathbf{EPP}} \cdot \ltimes$
Tier: $\rtimes \cdot \mathbf{D}_{\mathbf{EPP}} \cdot \mathbf{T}_{\mathbf{EPP}} \cdot \ltimes$

c. *We seem [$_{\mathrm{CP}}$ **that** ___ have a problem].
Path: $\rtimes \cdot \mathbf{D}_{\mathbf{EPP}} \cdot v \cdot \mathsf{T} \cdot \mathbf{C} \cdot \mathsf{V} \cdot v \cdot \mathbf{T}_{\mathbf{EPP}} \cdot \ltimes$
Tier: $\rtimes \cdot \mathbf{D}_{\mathbf{EPP}} \cdot \mathbf{C} \cdot \mathbf{T}_{\mathbf{EPP}} \cdot \ltimes$

$\mathrm{D}_{\mathrm{EPP}}$ and $\mathrm{T}_{\mathrm{EPP}}$ are adjacent on the tier in the licit examples (tier $\rtimes \cdot \mathsf{D}_{\mathsf{EPP}} \cdot \mathsf{T}_{\mathsf{EPP}} \cdot \ltimes$) but not in the hyperraising example (tier $\rtimes \cdot \mathsf{D}_{\mathsf{EPP}} \cdot \mathsf{C} \cdot \mathsf{T}_{\mathsf{EPP}} \cdot \ltimes$). As will be discussed shortly, we aim only to ensure that the mover is immediately followed by the landing site. Accordingly, we only need to ban substrings which consist of a mover followed by anything else. Thus, we have the following grammar:

(2) Grammar for EPP-movement
$T = \{\mathsf{D}_{\mathsf{EPP}}, \mathsf{T}_{\mathsf{EPP}}, \mathsf{C}\}$
$G^{-} = \{\mathsf{D}_{\mathsf{EPP}} \cdot \mathsf{D}_{\mathsf{EPP}}, \mathsf{D}_{\mathsf{EPP}} \cdot \mathsf{C}, \mathsf{D}_{\mathsf{EPP}} \cdot \ltimes\}$

The reader may confirm that the tier for the hyperraising example contains the illicit 2-factor $\mathsf{D}_{\mathsf{EPP}} \cdot \mathsf{C}$, while the tiers for the grammatical examples contains no illicit 2-factors. ⌟

Essentially, a TSL grammar allows us to ignore elements like VP, NP, etc., which are irrelevant to the long-distance dependency in question. The next subsection introduces a syntactic framework which provides the strings assumed in the above examples.

[2]Following MG convention, intermediate landing sites are not modeled directly.

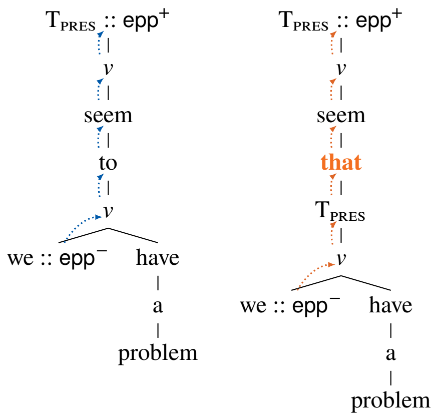


Figure 1: Dependency trees for *We seem to have a problem* (left) and **We seem that have a problem* (right) showing a-strings for moving elements. In the latter structure, *that* intervenes, preventing movement.

2.3 Dependency trees and ancestor strings

Following recent work in subregular syntax (Shafiei and Graf, 2020; Graf, 2022b, a.o.), I use MG dependency trees for the syntactic representation. Examples for sentences (1b) and (1c) are given in Figure 1. In these trees each node is a lexical item; compared to X-bar trees, each head and its projections are collapsed into a single node. The daughters of a node are its arguments, ordered from right to left in order of first merge, such that the rightmost daughter is the complement and all others are specifiers. For example, the right daughter of embedded *v* is the head of the complement VP, and the left daughter is the head of the DP subject (its specifier). In addition, each node is annotated with MG features guiding the Merge and Move operations (cf. Stabler, 1997, 2011). Since we are not concerned with local dependencies here, only Move features are shown. Positive features mark landing sites, and negative features mark moving elements. For example, finite T bears epp^{+} and the subject D head bears epp^{-}. Note that all elements appear in their base positions only, as in standard MG derivation trees.

Let us now implement a string-based model of movement constraints in which we extract the path from each mover to the root of the tree. Essentially, we take the order imposed by the (inverted) dominance relation and ignore the sibling relation. Shafiei and Graf (2020) call such paths *ancestor strings*, or a-strings, which they used to model a subset of the island constraints, including the *wh*-island constraint and the complex NP island constraint. First, we will see how this works for EPP-movement, then briefly discuss *wh*-movement.

Example 3 In order to keep the notation concise, I substitute most lexical items with their categories, and place the movement features as subscripts without the +/− diacritic, as before. Thus, the a-strings for the EPP movers in the structures in Figure 1 are:

Raising (✓):	$\mathsf{D}_{\mathsf{EPP}} \cdot v \cdot \mathsf{T} \cdot \mathsf{V} \cdot v \cdot \mathsf{T}_{\mathsf{EPP}}$
Hyperraising (✗):	$\mathsf{D}_{\mathsf{EPP}} \cdot v \cdot \mathsf{T} \cdot \mathsf{C} \cdot \mathsf{V} \cdot v \cdot \mathsf{T}_{\mathsf{EPP}}$

These are exactly the same strings as before, so we can continue to use the grammar in (2). ⌟

Example 4 The *wh*-island constraint can be described as a ban on A′-movement paths (including but not limited to *wh*-movement) which are interrupted by an interrogative CP, as illustrated by the difference between (3a) and (3b). Movement paths (a-strings) and their *wh*-tiers are included below each example, and the full structures are shown in Figure 2. For simplicity, we abstract away from EPP-movement and model only *wh*-movement.

(3) a. What did you think **that** John ate ___?
Path: $\mathsf{D}_{\mathsf{WH}} \cdot \mathsf{V} \cdot v \cdot \mathsf{T} \cdot \mathsf{that} \cdot \mathsf{V} \cdot v \cdot \mathsf{T} \cdot \mathsf{C}_{\mathsf{WH}}$
Tier: $\mathsf{D}_{\mathsf{WH}} \cdot \mathsf{C}_{\mathsf{WH}}$

b. *What did you wonder **whether** John ate ___?
Path: $\mathsf{D}_{\mathsf{WH}} \cdot \mathsf{V} \cdot v \cdot \mathsf{T} \cdot \mathsf{whether} \cdot \mathsf{V} \cdot v \cdot \mathsf{T} \cdot \mathsf{C}_{\mathsf{WH}}$
Tier: $\mathsf{D}_{\mathsf{WH}} \cdot \mathsf{whether} \cdot \mathsf{C}_{\mathsf{WH}}$

We can construct a very similar grammar to the previous one which captures this blocking effect:

(4) Grammar for *wh*-island constraint
$T \;\; = \{ \mathsf{D}_{\mathsf{WH}}, \mathsf{C}_{\mathsf{WH}}, \mathsf{whether} \}$
$G^- = \{ \mathsf{D}_{\mathsf{WH}} \cdot \mathsf{D}_{\mathsf{WH}}, \mathsf{D}_{\mathsf{WH}} \cdot \mathsf{whether}, \mathsf{D}_{\mathsf{WH}} \cdot \ltimes \}$

As before, the tier projection for the island violation contains the illicit 2-factor $\mathsf{D}_{\mathsf{WH}} \cdot \mathsf{whether}$, while the non-island structure contains no such 2-factors. ⌟

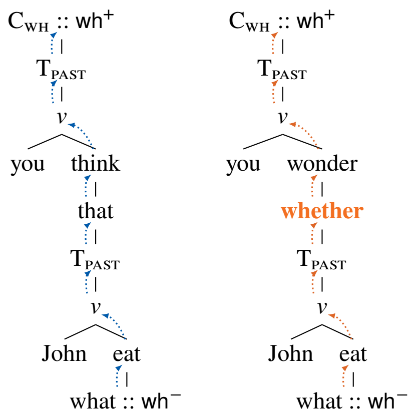


Figure 2: Dependency trees for *What did you think that John ate?* (left) and **What did you wonder whether John ate?* (right). In the latter structure, *whether* intervenes.

Note that because the a-string of a node extends to the root of the tree, it may contain fragments of other movement paths as well as nodes that are not part of any movement path. Our grammar is constructed in such a way that this does not pose an issue. However, the approach does have several limitations, as discussed below. Additionally, applying Heuser et al.'s algorithm to a-strings requires this extra material to be removed, as discussed in §3.

Also note that TSL grammars such as those in (2) and (4) enforce only these constraints and no others. As alluded to in the introduction, we must intersect these and other constraints, including local constraints, to produce a *multi-TSL* (MTSL) grammar. This is just a set of pairs of tier alphabets and associated constraints (grammars with the same tier alphabet can be intersected directly); see De Santo and Graf (2019) for details.

2.4 The strengths and limitations of a-strings

A-strings encode only enough information to enforce constraints base on containment (dominance) from the perspective of the mover. Shafiei and Graf (2020) use them to model island constraints, and as we have seen, certain other blockers can be handled in the same manner. We can also ensure that the mover has a landing site and capture some cases of relativized minimality, namely those where a mover contains another mover of the same type.

So, what can a-strings not do? Notably, they do not allow us to ensure that every landing site has exactly one mover. This requires tree tiers, as in Graf (2022b). They also cannot handle all cases of relativized minimality, as c-commanding specifiers do not appear in an a-string; this requires the *command strings* (c-strings) of Graf and Shafiei (2019). Additionally, to model specifier islands, information encoding left branches must be added to the string. See Shafiei and Graf (2020) for further discussion. The focus of this paper is on learning the tier alphabet; for this the a-string model will suffice, and the results should in principle extended to more complete models.

3 Distributional learning of syntactic blockers

I now describe the algorithm from Heuser et al. (2024), adapted to the syntactic framework presented in the previous section. We then discuss the ways in which the algorithm can do more than it was originally intended to, but being essentially an SL learner rather than a TSL learner, is not a complete solution on its own.

3.1 Preliminaries

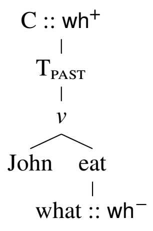
The algorithm assumes that the learner has already parsed the input and identified both moved elements and their initial positions. Now, they must determine the licit paths from the mover to the landing site for each type of movement. This can be cast as the learning an SL-2 grammar over (truncated) a-strings for each movement dependency. It is also assumed that the learner will generalize to unseen paths via the Tolerance Principle (TP, Yang 2016). The equates to a procedure for adding some but not all missing k-factors to the grammar.

While some readers may worry about taking the tree structure as a given, this essentially reduces to the assumption that long-distance syntactic dependencies are parasitic on local constituent structure, which must be learned regardless. Similarly, some other mechanism is responsible for identifying moved elements. It is conceivable that each of these can be learned distributionally with the TP, though such work is still in its infancy. See, e.g., Liang et al. (2022) regarding the learning of syntactic categories, and Li and Schuler (2023) regarding recursive embedding.

3.2 Tracking bigrams

Consider the *wh*-object question in (5), assumed to be in the input. The learner gathers from this that $\mathsf{what} \cdot \mathsf{eat} \cdot v \cdot \mathsf{T}_{\mathsf{PAST}} \cdot \mathsf{C}_{\mathsf{WH}}$ is a licit movement path, but does not know (yet) that every sequence of categories $\mathsf{D}_{\mathsf{WH}} \cdot \mathsf{V} \cdot v \cdot \mathsf{T} \cdot \mathsf{C}_{\mathsf{WH}}$ is a licit path.

(5) a. What did John eat ___?

b. 

c. a-string: $\mathsf{what} \cdot \mathsf{eat} \cdot v \cdot \mathsf{T}_{\mathsf{PAST}} \cdot \mathsf{C}_{\mathsf{WH}}$

The learner begins memorizing the attested 2-factors from each path, which is just the procedure for learning a (positive) SL-2 string grammar (Heinz, 2010). From the current example, they learn that $\{\mathsf{what} \cdot \mathsf{eat},\ \mathsf{eat} \cdot v,\ v \cdot \mathsf{T}_{\mathsf{PAST}},\ \mathsf{T}_{\mathsf{PAST}} \cdot \mathsf{C}_{\mathsf{WH}}\}$ are all licit 2-factors.[3] Heuser et al. show that because functional categories like v and T are few in number and frequent in the input, the learner will discover that all combinations may occur. For example, they will learn that *wh*-movement may occur over transitive and intransitive v, past and present tense, and so on.

Note that we must truncate of the a-string at the landing site when it is not the root, since the portion beyond the landing site may contain bigrams which cannot occur along the movement path. For example, in the sentence *Who wonders what John ate?*, the full a-string for *what* contains the bigram $\mathsf{C}_{\mathsf{WH}} \cdot \mathsf{wonder}$. If not excluded, the learner would conclude that movement over interrogative C is permitted. We will return to this issue in §3.5.

3.3 Generalizing with the Tolerance Principle

With regard to lexical categories such as verbs, the learner needs to invoke the TP. Given a class of N items and a proposed generalization, the TP states that the learner will adopt the generalization iff the number of items M in this class which are known to fit the generalization exceeds a threshold θ_N, where

$$\theta_N = N/ln(N)$$

In this case, N is the total number of verbs they have learned, and M is the number that have been attested with *wh*-movement. Heuser et al. show that for English, *wh*-movement of objects occurs with a large proportion of the most frequent verbs in child-directed speech—the number of exceptions far below the threshold—so the learner will adopt the generalization that *wh*-movement is permitted across all verbs. This is equivalent to adding all missing 2-factors of the form $\mathsf{D}_{\mathsf{WH}} \cdot \mathsf{V}$ and $\mathsf{V} \cdot v$ to the grammar.[4]

This brings us to islands. Once the learner observes cross-clausal movement from an embedded declarative such as (3a), they will add $\mathsf{T}_{\mathsf{PAST}} \cdot \mathsf{that}$ and $\mathsf{that} \cdot \mathsf{think}$ to the grammar. But if movement across a certain structure, such as the *wh*-island violation in (3b), is not attested, and the TP does not permit generalization, then the relevant 2-factors will never be added to the grammar. Heuser et al. (2024) show that this is indeed what we expect for “strong islands” in English. They also show how this derives the fact that not all verbs which take CP complements allow *wh*-movement, forming so-called “selective islands”. Although the learner

[3] I continue to ignore edge markers for simplicity.

[4] The TP does not provide the class of possible generalizations, only whether a given generalization is “good enough”. For present purposes, I assume that syntactic categories such as V/A/N/P/T/C are the only conditioning factors.

observes *wh*-movement across verbs like *think* and *say*, they do not observe movement across verbs such as *complain* and *quip*, and there are too many such verbs for the TP to permit generalization to the full class of verbs which select for a CP.

3.4 Beyond islands

To briefly summarize, the algorithm constructs a positive SL-2 grammar encapsulating the crucial information about licit and illicit movement paths where blockers are effectively encoded as missing 2-factors. Although not discussed by Heuser et al. (2024), the approach is equally applicable to other restrictions on movement such as those discussed by Keine (2019), which are the focus of §5.

It is also applicable to non-movement dependencies, to the extent that pairs of dependent items can be identified. Shafiei and Graf (2020) note that constraints on long-distance linguistic patterns tend to take involve a domain and blockers within that domain. For movement, the domain elements are movers and their landing sites, while for agreement we have, in Minimalist terms, elements with unvalued and valued features of the same type. Indeed, Keine's version of the Height-Locality Connection treats movement and agreement equally. If these are learned in the same way, then we have an explanation for this close correspondence.

Finally, note that the same properties that allow learning of weak islands also allow for cross-linguistic variation such as the availability of hyperraising (Charles Yang, p.c.). Specifically, it predicts that hyperraising should only be allowed if robustly attested in the input. This, of course, raises the question of how such structures ever arise. But we could just as easily ask the same of long-distance *wh*-movement, which is by now known to be more or less restrictive in different languages. For now, we must set these diachronic questions aside.

3.5 Limitations of SL learning

The fact that the Heuser et al. (2024) algorithm is essentially an SL learner means that the resulting grammars cannot be applied to arbitrary a-strings, only those which start with a mover and which are truncated at the first landing site. This is because it is in general not possible for an SL grammar to relate two elements which do not occur in the same k-factor. As a consequence, it is impossible to ensure that there is exactly one landing site per mover, nor to detect whether a blocker actually occurred along a movement path and not somewhere else. In contrast, our TSL grammars from §2 do not suffer from either restriction.

Thus, truncating the a-string only creates the illusion that SL is adequate. While this operation is useful in the learning algorithm, including it in the grammar would increase its power, producing a class that is quite different from TSL.[5] Instead, what we want to do is to take the information that was obtained using this technique and encode it in a TSL grammar, which has the right formal properties. This is the topic of the next section.

4 Constructing the tier

To review the discussion so far, we can frame our learning problem as follows: given a corpus of MG dependency trees, how do we discover the TSL constraints on long-distance dependencies over a-strings? In particular, how do we discover which elements other than the dependent items are visible?

We have already seen how Heuser et al.'s path-based algorithm forms the foundation of an appealing solution, but on its own is not enough. This section begins with a more detailed summary of the issues involved with TSL learning before attempting to bridge the gap by modifying the Heuser et al. algorithm to produce a TSL grammar.

4.1 The problem of learning tiers

TSL languages are efficiently learnable given a fixed tier alphabet and k-factor size(Lambert et al., 2021), but this may not be a realistic assumption for natural language. There is reason to think that the value of k rarely exceeds 2 for long-distance constraints (McMullin, 2016; Graf, 2022b; Hanson, 2024), but it is far less clear that the tier alphabet can be known in advance. Because the number of possible tiers alphabets is exponential in the size of the full alphabet (it is $2^{|\Sigma|}$), we must avoid exhaustive search of this space. While there exist efficient (polynomial time) algorithms that determine the tier alphabet from positive data (Jardine and McMullin, 2017; Lambert, 2021), these are not robust against interaction with other constraints. Since natural language almost always involves the interaction of many constraints, this prevents such algorithms from being used with real world data.

One way of tackling the problem is to find ways to pare down the hypothesis space such that the brute force method becomes practical. For example, we

[5] It would be a subclass of IBSP. Shafiei and Graf (2020) also use IBSP, although in a very different manner.

could appeal to formal universals on the relations between the alphabets of different tiers (Aksënova and Deshmukh, 2018). Alternatively, we could make use of substantive universals such as some version of the Height-Locality Connection; Keine's (2019) version says that a "lower" category can be a blocker for a "higher category", e.g. *v* cannot be a blocker for a landing site at T.

Another possibility, which I pursue here, is to identify a set of heuristics which allows the learner to discover the tier alphabet without ever engaging in exhaustive search. In other words, the supposedly impossible tiers are in fact perfectly valid, but the learner will never posit them under normal conditions due to the way in which they navigate the hypothesis space. The crucial heuristic in this case, taken from Heuser et al. (2024), is that by restricting our attention to the path between two dependent elements, we can identify its blockers, which must appear on the same tier.

In this case, the Height-Locality Connection becomes a side effect of the learning process rather than a cause, and is also unified with the theory of islands. As discussed earlier, the close similarity of movement and agreement constraints is derived as well. Yet another issue with existing TSL learners is that they all involve exact identification in the limit, whereas children must generalize from limited data. Though orthogonal to our main focus, the adoption of the TP largely solves this problem as well. Altogether, the proposed approach not only solves several major learnability problems for the TSL model, but also adds several typological predictions which are not inherent to the model.

4.2 From local to tier-based constraints

Existing TSL learners infer the tier alphabet by utilizing a definitional property of a TSL-k language: any symbol not on the tier can be freely inserted and deleted without changing the well-formedness of a string. As discussed by Lambert (2021), we can do this by keeping track of just the sets of attested local k-factors and $(k+1)$-factors. Since the k-factors can themselves be obtained from the $(k+1)$-factors, only the latter must be memorized. Thus, in principle we can use the local 2-factors discovered by Heuser et al. (2024)'s algorithm to identify tier-based 1-factors, which are the blockers themselves. By recombining these blockers with the dependent items that bookend the path, we can construct the desired TSL-2 grammar.

However, we have still not addressed the problem of interaction with local constraints. Detecting free insertion and deletion as described above requires collecting every possible local $(k+1)$-factor in a TSL language, but the existence of other constraints means that this will never happen. For instance, every permutation of every subset of a functional hierarchy would have to occur in the input for these elements to be removed from the tier.

I propose that we can solve this problem by using the background grammar encoding local constraints as the standard of comparison for free insertion and deletion. Recall the behavior of our path-based learner for *wh*-movement structures such as those in (3a) and (5). After decomposing paths and applying the TP, the resulting grammar will contain a dense network of 2-factors of the form $\{\mathsf{D}_{\mathsf{WH}} \cdot \mathsf{V}$, $\mathsf{V} \cdot v$, $v \cdot \mathsf{T}$, $\mathsf{T} \cdot \mathsf{C}_{\mathsf{DECL}}$, $\mathsf{C}_{\mathsf{DECL}} \cdot \mathsf{V}\}$, but not $\mathsf{T} \cdot$ whether or whether $\cdot\, \mathsf{V}$. All of these 2-factors are licit when they do *not* occur along a *wh*-movement path, and are therefore part of the local constraint grammar. As a result, we can infer that *whether* is a blocker due to the conspicuous absence of 2-factors which contain it. In contrast, 2-factors like $\mathsf{T} \cdot v$ (reverse order) and $\mathsf{V} \cdot \mathsf{C}$ (skipping T) are already missing in the local constraint grammar, so their absence in the movement path grammar can be ignored.

4.3 Algorithm

The proposed algorithm is as follows. Let G^2_L be the positive SL-2 grammar for local constraints and G^2_M be the grammar for movement type M. Construct G^2_L by collecting all 2-factors from all a-strings, and construct G^2_M from truncated a-strings as before. Add missing 2-factors to each where permitted by the TP. Next, construct G^1_L and G^1_M by decomposing the 2-factors in G^2_L and G^2_M into their constituent 1-factors.

Now we test for tier membership. Free deletion is vacuous for TSL-1, since it is trivially true that for every symbol, removing that symbol from an attested 2-factor which contains it in a certain position produces an attested 1-factor (this not necessarily true for larger values of k).

The crucial test, corresponding to the free insertion test, tests for factors missing from G^2_M but present in G^2_L. Let $G^2_D = G^2_L \setminus G^2_M$. For every symbol, we ask if it can be added to either side of 1-factor in G^1_M to produce a 2-factor in G^2_D; if so, then the symbol is a blocker. Finally, we construct the target TSL-2 grammar, which consists of 2-factors containing the mover followed by a blocker, another mover, or the right edge marker.

Example 5 Given typical data, the grammar G_M^2 for *wh*-movement will include all 2-factors of the form $\{\mathsf{D_{WH}} \cdot \mathsf{V},\ \mathsf{V} \cdot v,\ v \cdot \mathsf{T},\ \mathsf{T} \cdot \mathsf{C_{DECL}},\ \mathsf{T} \cdot \mathsf{C_{WH}}\}$. It also contains $\mathsf{that} \cdot \mathsf{think}$ and $\mathsf{that} \cdot \mathsf{say}$ but not $\mathsf{that} \cdot \mathsf{complain}$ or $\mathsf{that} \cdot \mathsf{quip}$. G_L^2 contains all of these, so the difference G_D^2 includes $\mathsf{that} \cdot \mathsf{complain}$ and $\mathsf{that} \cdot \mathsf{quip}$. If we consider the elements *complain* and *quip*, we could add *that* from G_M^1 to 2-factors in G_D^2, so they are blockers. In contrast, even though *that* has containing 2-factors in G_D^2, these cannot be constructed by adding a symbol from G_M^1, so they are not blockers. ⌟

Based on examples like these, it would appear that comparing just G_L^1 and G_M^1 is sufficient, since any element in G_L^1 but not G_M^1 is guaranteed to have a containing factor in G_D^2. If this reasoning is correct, it may be possible to simplify the above procedure. However, it renders the relation to Lambert (2021) opaque, and there may be corner cases which have not been considered. Also, the fact that movement paths are calculated from the base position could affect the predictions of the model when we look beyond EPP-movement and *wh*-movement. I leave the investigation of such details to future work.

4.4 Discussion

The reader may be wondering why we do not simply track local 3-factors in order to directly infer tier-based 2-factors. There are several problems with this method, but first and foremost is that it greatly increases data sparsity. Although Heuser et al. (2024) found empirical success with local 2-factors, it is not clear whether the TP will allow the same generalizations when applied to 3-factors.

Next, I should describe how the model could be extended beyond domain-based constraints on movement. Handing agreement should be straightforward; we just need to add positive and negative agreement features analogous to MG movement features, as in Hanson (2024). Other dependencies such as case assignment would require identification of the relevant domain nodes (i.e. as in dependent case theory), and we could in principle adapt the algorithm to c-strings in order to identify constraints on c-commanding elements.

Finally, I wish to briefly mention some alternative approaches to learning long-distance syntactic dependencies. Many of these are probabilistic models; for example, the model in Pearl and Sprouse (2013) tracks path trigram probabilities in order to learn syntactic islands. This is not entirely dissimilar to the present model, except that we do not attempt to learn gradient constraints. It is, of course, possible to introduce gradience into subregular models; see Mayer (2021) and Torres et al. (2023). The present paper, by incorporating a TP-based model, relegates the use of frequency/probability to a small corner of the learning algorithm. In principle, we could adapt it to produce a probabilistic TSL grammar by comparing k-factor probabilities rather than discrete k-factors.

5 On the Height-Locality Connection

The Height-Locality Connection (HLC) is the observation that restrictions on long-distance syntactic dependencies correlate with the category of the "height" of the upper element (e.g. landing site) such that higher categories can enter into more distant dependencies (Keine, 2019). While several distinct theories can be found in the literature (Williams, 2002; Abels, 2012, a.o.), the present approach is most directly comparable to Keine's theory of Probe Horizons, in which each type of probe (i.e. a head that hosts a landing site or unvalued feature) has a *horizon* beyond which no dependencies can be formed. In TSL terms, a horizon is simply a blocker on a tier, and in this sense no different from an island, a bounding node in the binding theory, or any other such element. I show here that the learning algorithm from the previous section predicts a version of the HLC which is similar though not identical to Keine's.

For Keine, the horizon for each combination of major category and active feature is lexically specified. For example, finite T in English bears some feature (which we have been calling epp^+) which triggers movement of the subject. This probe can see into a non-finite TP, but not a finite CP. Thus, C is a horizon for this dependency in English, but it need not be so in other languages. Keine shows T is a horizon for analogous A-movement in Hindi; in languages with hyperraising neither T nor C is a horizon. As we have discussed, this variation is a core prediction of the TSL model as well.

However, according to Keine, it is not the case that any category is a possible horizon for any probe, only those that are *at least as high* as the category of the probe. This means that a probe on T can never have v or V as a horizon, for example.[6]

[6] I refer the reader to Section 5 of Keine (2019) regarding the derivation of this generalization, which is based on

Restricting our attention to the basic clausal spine, this yields the typology of possible horizons shown in (6). Thus, we might rephrase the HLC as saying that higher categories *must* have a larger locality domain; lower categories may see just as far, but have smaller domains as a tendency.

(6)

Category	Possible Horizons
C	C
T	C, T
v	C, T, *v*

Let us consider how such a generalization could arise from the learning algorithm outlined here. In the case of EPP-movement, the learner observes movement from Spec-*v*P in simple transitive clauses, and out of VP in the case of unaccusatives and passives. When all is said and done, V and *v* do not appear on the tier, and so are not horizons. If the learner also observes raising out of TP (as in English), T will be removed as well, as will C in a language with hyperraising, but for V and *v* this is all but guaranteed, since DPs in general originate within these phrases. By the same logic, the learner will remove C from the tier for *wh*-movement only if cross-clausal movement is observed (as it is in English), but the observation of *wh*-object movement even in simplex clauses necessarily rules out V, *v*, and T since all are below C.

To be fully explicit, the proposed algorithm predicts the HLC to be a tendency rather than a strict rule in *both* directions: lower categories usually have smaller locality domains, and higher categories usually have larger ones, but exceptions are in principle possible in both directions. Again, in our representative examples of EPP-movement and *wh*-movement the relevant class of movers is able to occur in the complement of VP, the lowest possible position in the clausal spine; invisibility of the entire functional sequence below the probe follows as a result. Thus, to determine whether Keine's generalization is truly correct, we would need to find a class of mover which originates only in higher positions, that is, one which does not include any DPs. At present, I do not know of a good candidate class of movers to perform this test.

To close this section, I wish to reemphasize the generality of the proposed learning algorithm, which is equally relevant to islands and other kinds of blockers. In his discussion of acquisition, Keine notes that the implicational hierarchy imposed by his theory provides the learner with a safe way of navigating the space of possible horizons, starting with the assumption that the category of the probe is also the lowest horizons, and removing horizons from the grammar as required by the input. This is correct, and our algorithm works from a similar principle. But Keine's assumption that projections lower than the probe cannot be horizons is not necessary to achieve this.

the assumption that functional projections involve "feature inheritance" of lower categories in the functional sequence.

6 Conclusion

In this paper, I proposed an algorithm which allows for the creation of TSL grammars from the output of Heuser et al.'s path-based algorithm, avoiding the need to search the space of tier alphabets. This approach combines the strengths of their algorithm with those of the TSL model, and derives the Height-Locality Connection as a byproduct of the learning process. While this paper used a-strings and focused on movement, the principle of inferring tier-based constraints via comparison of SL grammars should in principle extend to other TSL models of syntax and other dependencies such as agreement and case. I leave investigation of these to future research.

More broadly, this work represents the start of integration between subregular syntax and acquisition theories based on the TP. I am aware only of one other line of work which involves learning TSL grammars with the TP, which is Belth's (2023) algorithm for learning long-distance harmony. Since subregular linguistics has consistently shown a great deal of formal similarity across domains, it would be prudent to examine whether Belth's algorithm can be applied to the problem of learning syntactic dependencies, and vice versa. Formal learnability has long been central to subregular linguistics, but as I hope to have shown, future progress may rely on looking also to theories grounded in the empirical facts of child language acquisition.

Acknowledgments

This work partly was supported by the National Science Foundation under Grant No. BCS-1845344 and by an award from the Institute for Advanced Computational Science at Stony Brook University. I thank Jordan Kodner and Sarah Payne for reviewing an early draft of the paper. I also thank three anonymous reviewers for their detailed comments, which helped to improve the clarity of several key issues.

References

Klaus Abels. 2012. The Italian left periphery: A view from locality. *Linguistic Inquiry*, 43(1):229–254.

Alëna Aksënova and Sanket Deshmukh. 2018. Formal restrictions on multiple tiers. In *Proceedings of the Society for Computation in Linguistics 2018*, pages 64–73.

Adriana Belletti. 2018. Locality in syntax. In *Oxford Research Encyclopedia of Linguistics*. Oxford University Press.

Caleb Belth. 2023. Towards a learning-based account of underlying forms: A case study in Turkish. In *Proceedings of the Society for Computation in Linguistics 2023*, pages 332–342.

Aniello De Santo and Thomas Graf. 2019. Structure sensitive tier projection: Applications and formal properties. In *Formal Grammar*, pages 35–50, Berlin, Heidelberg. Springer.

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 2022*, pages 184–193.

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

Kenneth Hanson. 2023. Strict locality in syntax. In *Proceedings of CLS 59*.

Kenneth Hanson. 2024. Tier-based strict locality and the typology of agreement. Ms. Stony Brook University.

Jeffrey Heinz. 2010. String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 897–906, Uppsala, Sweden. Association for Computational Linguistics.

Jeffrey Heinz. 2018. The computational nature of phonological generalizations. In Larry M. Hyman and Frans Plank, editors, *Phonological Typology*, number 23 in Phonetics and Phonology, pages 126–195. De Gruyter Mouton.

Annika Heuser, Hector Vazquez Martinez, and Charles Yang. 2024. The learnability of syntactic islands. Presentation at NELS 54.

Adam Jardine and Kevin McMullin. 2017. Efficient learning of tier-based strictly k-local languages. In *Language and Automata Theory and Applications*, pages 64–76, Cham. Springer International Publishing.

Stefan Keine. 2019. Selective opacity. *Linguistic Inquiry*, 50(1):13–62.

Dakotah Lambert. 2021. Grammar interpretations and learning TSL online. In *Proceedings of the Fifteenth International Conference on Grammatical Inference*, volume 153 of *Proceedings of Machine Learning Research*, pages 81–91. PMLR.

Dakotah Lambert, Jonathan Rawski, and Jeffrey Heinz. 2021. Typology emerges from simplicity in representations and learning. *Journal of Language Modelling*, 9(1):151–194.

Daoxin Li and Kathryn D. Schuler. 2023. Acquiring recursive structures through distributional learning. In *BUCLD 47: Proceedings of the 47th Annual Boston University Conference on Language Development*.

Kevin Liang, Diana Marsala, and Charles Yang. 2022. Distributional learning of syntactic categories. In *BUCLD 46: Proceedings of the 46th annual Boston University Conference on Language Development*.

Connor Mayer. 2021. Capturing gradience in long-distance phonology using probabilistic tier-based strictly local grammars. In *Proceedings of the Society for Computation in Linguistics 2021*, pages 39–50.

Kevin McMullin. 2016. *Tier-based locality in long-distance phonotactics: learnability and typology*. Ph.D. thesis, University of British Columbia.

Lisa Pearl and Jon Sprouse. 2013. Syntactic islands and learning biases: Combining experimental syntax and computational modeling to investigate the language acquisition problem. *Language Acquisition*, 20(1):23–68.

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

Edward P. Stabler. 1997. Derivational Minimalism. In Christian Retore, editor, *Logical Aspects of Computational Linguistics*. Springer.

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

Charles Torres, Kenneth Hanson, Thomas Graf, and Connor Mayer. 2023. Modeling island effects with probabilistic tier-based strictly local grammars over trees. In *Proceedings of the Society for Computation in Linguistics 2023*, pages 155–164.

Edwin Williams. 2002. *Representation theory*. MIT Press.

Charles Yang. 2016. *The price of linguistic productivity: How children learn to break the rules of language*. MIT Press.