Learning Multi Tier-Based Strictly 2-Local Languages

Logan Swanson
Stony Brook University
logan.swanson@stonybrook.edu

Abstract

The class of *tier-based strictly 2-local* (TSL-2) languages has been shown to be useful in modeling patterns in both phonology (Heinz et al., 2011) and syntax (Graf, 2022a). This paper presents an algorithm for learning the intersection closure of the TSL-2 languages, the *multi*-TSL-2 (MTSL2) languages over arbitrary structures. The algorithm builds on prior work on learning a subclass of MTSL2 over trees (Swanson, 2024b), as well as insights about searching partially ordered spaces (Chandlee et al., 2019). I show that the algorithm correctly learns the MTSL2 class from a limited data sample and discuss tradeoffs with the existing approach offered by Swanson (2024b).

1 Introduction

Understanding the structure of linguistic knowledge and how humans acquire this knowledge from limited input is one of the key questions in linguistics. The field of *subregular linguistics* (Heinz, 2018, Graf, 2022a a.o.) approaches this question by seeking out formal language classes which are complex enough to represent the range of patterns found in human language, but structured enough to be efficiently learned. Creating learning algorithms for these classes not only demonstrates that they *can* be learned, but also sheds light on *how* different learning strategies behave, what kinds of time and data requirements they impose, and how they compare with human learners.

The subregular class of *tier-based strictly local* (TSL) languages has recently emerged as particularly relevant for linguists. Intuitively, TSL languages can capture dependencies which are immediately local, once a certain set of irrelevant elements is ignored (Lambert, 2023). This class is useful for analyzing many of the patterns found in human language, where long-distance dependencies are often restricted or relativized to a particular set of elements (Heinz, 2018). One such example is harmony patterns, like the sibilant harmony pattern found in Samala, a Chumash language from southern California. In Samala, sibilants in the same word must agree in anteriority, so words containing 's...s' are permitted, but those with '*s...ʃ' are forbidden (Hansson, 2010). The TSL grammar for this pattern consists of a tier for the sibilant sounds, $tier$ = {s, ʃ}, and constraints over that tier banning adjacent sibilants which disagree in anteriority: $constraints$ = *sʃ, *ʃs. So, a word like [sapitsolus] ('he has a stroke of good luck') is acceptable because its projection onto the sibilant tier, "sss", does not contain any of these banned factors. Adding the past-tense suffix /-waʃ/, however, results in the form [ʃapitʃoluʃwaʃ]. This form is grammatical, with tier projection "ʃʃʃʃ", but the fully faithful form [sapitsolus-waʃ] is ungrammatical, since its tier projection "sssʃ" contains the banned factor "sʃ". Specifically, this pattern is TSL-**2**, since the constraints needed to enforce it contain at most two elements each.

TSL-2 can largely capture the typology of phonotactic patterns seen in natural language, including local dependencies, long-distance harmony, and blocking (McMullin and Hansson, 2014). In addition, a variety of syntactic patterns have been shown to belong to the parallel class of TSL-2 over tree structures, including verb agreement (Hanson, 2023b), case assignment (Hanson, 2023a), and movement patterns (Graf, 2022b). Moreover, the TSL-k languages are known to be efficiently learnable for any fixed value of k (Jardine and McMullin, 2017, Lambert, 2021). Human languages, however, typically involve many such TSL patterns operating at once, and these may interact with each other. To capture multiple TSL patterns which are active at once, the more complex class of *multi*-TSL (MTSL) is needed. An MTSL-k language is simply the intersection of one or more TSL-k languages, in other words "several TSL patterns applying at once".

Existing work on learning MTSL is limited to an algorithm sketched by McMullin et al. (2019) and adapted by Swanson (2024b) to be provably efficient and generalizable across trees and strings. This algorithm, however, does not learn the exact class of MTSL, but rather a subclass with a few additional restrictions–restrictions which prove problematic in the realm of syntax, where they do not seem to hold on natural language data (see Section 3).

This paper introduces MTSL-BUFIA, an algorithm which learns the class of MTSL proper without these additional restrictions. I show that MTSL-BUFIA exactly identifies the class of MTSL in the limit, in the sense of Gold (1967) and has polynomial data complexity, in the sense of de la Higuera (1997). However it has exponential time complexity in the worst case.

The remainder of this paper is organized as follows: Section 2 offers definitions for relevant terms used in this paper. Section 3 briefly recaps the existing MTSL learning work and summarizes the outstanding problems. Section 4 describes the novel MTSL learning algorithm MTSL-BUFIA, and Section 5 works through an example of it in action. Section 6 defines the representative sample for MTSL and proves that MTSL-BUFIA exactly identifies the class in the limit. Finally, Section 8 offers future directions and concludes.

2 Preliminaries

2.1 Representations and Relational Structure

An important insight from Swanson (2024b) is that MTSL-2 can be defined over any type of structure built from symbols and the relations between them. Rogers and Lambert (2019) describe this type of relational model and show some of their properties. I adopt adapted versions of three of their definitions here:

Definition 2.1 (Relational Structure)**.** A relational signature, $\mathbb{R}$, is a finite ranked alphabet of relation symbols. An $\mathbb{R}$-structure is a tuple $\mathcal{S} = \langle D, R_1^{\mathcal{S}}, R_2^{\mathcal{S}} ... \rangle$ where D is the domain and each $R_i^{\mathcal{S}}$ is an interpretation of some symbol from $\mathbb{R}$.

Definition 2.2 (Homomorphism)**.** Given $\mathbb{R}$-structures $\mathcal{S}$ and $\mathcal{S}'$ with domains D and D' respectively, a *homomorphism* from $\mathcal{S}$ to $\mathcal{S}'$ is a (total) function $h : D \rightarrow D'$ such that $\vec{a} \in R^{\mathcal{S}} \Longrightarrow h(\vec{a}) \in R^{\mathcal{S}'}$

Definition 2.3 (2-Factor)**.** Let $\mathcal{F}$ and $\mathcal{S}$ be relational structures with domains F and D respectively. $\mathcal{F}$ is a *2-factor* of $\mathcal{S}$ iff:

1. $|F| = 2$
2. $\forall x, y \in F[x \neq y \Longrightarrow \exists R \in \mathbb{R}, \vec{a} \in R^F[x, y$ both occur in $\vec{a}]]]$
3. $\exists h : F \rightarrow D$, a homomorphism

The set of all 2-factors of any structure $\mathcal{S}$ is denoted $2fac(\mathcal{S})$. Additionally, given some set of structures I, $2fac(I) := \{f \mid \exists \mathcal{S} \in I[f \in 2fac(\mathcal{S})]\}$.

2.2 Strings and Trees

These definitions are highly general, and can cover many types of structures. For the purposes of modeling language, however, the primary areas of interest are string models and tree models. I offer examples here to illustrate how these two kinds of models operate.

Example 2.1 (String Models)**.** *This example is adapted from Rogers and Lambert (2019). Let s be a string over the alphabet Σ. Let $|s|$ be the length of s. A string model for s is a structure:*

$$\mathcal{M}^{\triangleleft}(s) := \langle D^s, \triangleleft^s, P^s_{\sigma \in \Sigma} \rangle$$

Where:

D^s–is a set of natural numbers such that:

1. $0 \in D$
2. $\forall i, j \in D[i + 1 = j \Longleftrightarrow i \triangleleft j]$

$\triangleleft^s$–is the successor relation on s

P^s_σ–is the set of all positions in s at which the symbol σ occurs.

So the string "*pat*" would be modeled:

$$\mathcal{M}^{\triangleleft}(pat) := \langle D = \{0, 1, 2\}, \triangleleft = \{(0, 1), (1, 2)\}, P_p = \{0\}, P_a = \{1\}, P_t = \{2\} \rangle$$

And the 2-factor corresponding to the "*pa*" substring would be modeled:

$$\mathcal{M}^{\triangleleft}(pa) := \langle D = \{0, 1\}, \triangleleft = \{(0, 1)\}, P_p = \{0\}, P_a = \{1\} \rangle$$

These can trivially be related by a homomorphism $h(x) \rightarrow x$.

Example 2.2 (Tree Models)**.** *Let t be a tree over the alphabet Σ. Let $|t|$ be the number of nodes in t. A tree model for t is a structure:*

$$\mathcal{M}^{\lhd,\prec}(t) := \langle D^t, \lhd^t, \prec^t P^t_{\sigma\in\Sigma}\rangle$$

Where:

D^t–is a set of strings of natural numbers, including the empty string ε, such that:

1. $\varepsilon \in D$
2. $\forall u \in \mathbb{N}^*, j \in \mathbb{N}[uj \in D \Longrightarrow u \in D \wedge u \lhd uj]$
3. $\forall u, v \in \mathbb{N}^*[u \lhd v \Longrightarrow \exists i \in \mathbb{N}[v = ui]]$
4. $\forall u \in \mathbb{N}^*, j \in \mathbb{N}[uj \in D \Longrightarrow \forall i \in \mathbb{N}[i < j \Longrightarrow ui \in D]]$
5. $\forall u \in \mathbb{N}^*, j > 0 \in \mathbb{N}[uj \in D \Longrightarrow u(j-1) \prec uj]$
6. $\forall u, v \in \mathbb{N}^*[u \prec v \Longrightarrow \exists i, j \in \mathbb{N}, w \in \mathbb{N}^*[u = wi \wedge v = wj \wedge i + 1 = j]]$

$\lhd^t$–is the proper dominance relation on t

$\prec^t$–is the immediate left sibling relation on t

P^t_σ–is the set of all positions in t at which the symbol σ occurs.

These domain definitions essentially encode ordered addresses to each element, which is relevant for defining how tiers and 2-paths are constructed. For strings, this address encoding is a simple positional index. For trees, it is Gorn addresses, which encode the path through the tree required to reach that node from the root. Notably, these addresses are computationally easy to assign, requiring only a single traversal ($O(n)$ time) to do so.

In both of these structures, the unary relations of the form P_σ induce a *partition* on the domain elements, with each element being a member of exactly one of these sets. We can thus refer to the label of any given element $l(\alpha) = \sigma \Longleftrightarrow P_\sigma(\alpha)$.

Definition 2.4 (Tier Projection)**.** A *tier projection function* is a function $\tau()$ which takes as input a relational structure $\mathcal{S}$ with domain D and relations $\mathbb{R}$ and a set of unary relations $\mathbb{P} \subseteq \mathbb{R}$ and returns an output structure $\mathcal{S}'$ for which the following hold:

1. $\forall P \in \mathbb{P}[P = \varnothing]$
2. $\forall e \in D[\forall P \in \mathbb{P}[P(e) \Longrightarrow e \notin D']]$
3. $\forall R \in \mathbb{R}[R \notin \mathbb{P} \Longrightarrow ((R^{\mathcal{S}}(e_1, e_2, ...e_n) \wedge \nexists P \in \mathbb{P}[P(e_{i<=n})]) \Longrightarrow R^{\mathcal{S}'}(e_1, e_2, ...e_n))]$

For a structure $\mathcal{S}$ over some alphabet Σ, with each element in the structure bearing exactly one symbol $\sigma \in \Sigma$ (like trees or strings) we can also write $\tau(\mathcal{S}, T) = \tau(\mathcal{S}, \mathbb{P})$, where $T = \Sigma - \{\sigma | P^{\mathcal{S}}_\sigma \notin \mathbb{P}\}$.

For strings, we use the tier projection function

$$\tau_s(\langle D, \lhd, P_\sigma\rangle_{\sigma\in\Sigma}, T) := \langle D_T, \lhd_T, P_\sigma\rangle_{\sigma\in T}$$

where:

$$D_T := \{e \in D | l(e) \in T\}$$

$$\lhd_T := \{\langle x, y\rangle | x \in D' \wedge y \in D' \wedge x < y \wedge \nexists z \in D'[x < z < y]\}$$

Intuitively, this is just removing the non-tier elements and re-stringing the elements back together in the successor relation in their original order. In other words, precedence is preserved.

For trees, we use the tier projection function

$$\tau_t(\langle D, \lhd, \prec, P_\sigma\rangle_{\sigma\in\Sigma}, T) := \langle D_T, \lhd_T, \prec_T, P_\sigma\rangle_{\sigma\in T}$$

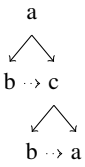
where:

$$D_T := \{e \in D | l(e) \in T\} \cup \{\rtimes, \ltimes\}$$

$$\begin{aligned}\lhd_T := \{\langle x, y\rangle | x \in D' \wedge y \in D' \wedge \exists u \in \mathbb{N}^*[y = xu] \wedge \\ \nexists z \in D'[\exists u, v \in \mathbb{N}^*[z = xu \wedge y = zv]]\} \quad \cup \\ \{\langle \rtimes, x\rangle | \nexists u, v \in \mathbb{N}^*[u \in D' \wedge x = uv]\} \quad \cup \\ \{\langle x, \ltimes\rangle | \nexists u, v \in \mathbb{N}^*[u \in D' \wedge u = xv]\}\end{aligned}$$

$$\begin{aligned}\prec_T := \{\langle x, y\rangle | \\ \exists u, v, w \in \mathbb{N}^*, i < j \in \mathbb{N}[x = uiv \wedge y = ujw] \wedge \\ \exists u \in \mathbb{N}^*[u \lhd_T x \wedge u \lhd_T y \wedge \\ \nexists z \in D'[\exists v, w, q \in \mathbb{N}^*, i < j \in \mathbb{N}[x = viw \wedge z = vjq] \wedge \\ \exists v, w, q \in \mathbb{N}^*, i < j \in \mathbb{N}[z = viw \wedge y = vjq] \wedge \\ u \lhd_T z]]\}\end{aligned}$$

Intuitively, this can be thought of as ripping out all non-tier elements from the tree and attaching their children as additional daughters in the same slot that node was pulled out of. For notational convenience, we also stipulate designated head and foot markers, $\rtimes$ and $\ltimes$ which mark the upper and lower edges of the tree. The $\rtimes$ marker also ensures that the output is a well-formed tree by maintaining parent closure.



2-paths:

		$\langle b \prec a, \varnothing \rangle$
$\langle a \lhd b, \varnothing \rangle$	$\langle c \lhd b, \varnothing \rangle$	$\langle b \prec a, \{b, c\} \rangle$
$\langle a \lhd b, \{c\} \rangle$	$\langle c \lhd a, \varnothing \rangle$	$\langle b \prec b, \{c\} \rangle$
$\langle a \lhd c, \varnothing \rangle$	$\langle a \lhd a, \{c\} \rangle$	$\langle b \prec c, \varnothing \rangle$

Table 1: Example tree and 2-paths. Solid arrows indicate dominance relation, and dashed arrows indicate sibling relation.

Definition 2.5 (2-Path)**.** The notion of *2-paths* was introduced by Jardine and Heinz (2016), and it is used to represent the idea of **intervention** in potential 2-factors. A 2-path (for a structure) is a tuple $\langle f, V \rangle$ where f is a 2-factor and V is a set of symbols which prevent that 2-factor from being present. This set of symbols which, if removed, would allow the 2-factor to be present is known as an **intervener set** for that 2-factor. The 2-path for factor f with intervener set V is denoted $\langle f, V \rangle$. For some set of structures D, we say that $paths(D)$ is all 2-paths present in all structures in D. For strings and trees, 2-paths can be computed in much the same way as tier projections, using the addresses of each element (Jardine and Heinz, 2016; Swanson, 2024b).

Table 1 demonstrates finding the 2-paths for an example tree. In this example, the tree is visualized with the labels of each domain element occupying each node, and with solid arrows indicating dominance and dashed arrows indicating siblinghood. Note that every relation which is present in this structure corresponds to a 2-path with an empty intervener set.

The remainder of this paper will not treat specific structures differently, but will instead deal exclusively with tiers, 2-factors, and 2-paths.

Definition 2.6 (MTSL)**.** An *MTSL language* is a set of structures over some alphabet Σ for which membership is defined by a grammar $G = \neg\langle f_1, T_1 \rangle \wedge \neg\langle f_2, T_2 \rangle \wedge ... \neg\langle f_i, T_i \rangle$ of 2-factor, tier pairs. A structure $\mathcal{S}$ is in the language (notated $L(G)$) iff $\nexists\, \langle f, T \in G \rangle [f \in 2fac(\tau(\mathcal{S}, T))]$.

3 Existing Approach: MT2SLIA

Swanson (2024b) introduces the *multi tier-based 2-strictly local inference algorithm* (MT2SLIA), a learning algorithm for a subclass of MTSL (over both trees and strings) which runs in polynomial time and data. This algorithm uses the idea that any 2-factor which is not present in the input sample must be banned on some tier. For each of these missing 2-factors, it finds all the intervener sets present in the data and uses these to construct the tier which forbids that 2-factor. To do this, elements from the smallest intervener sets are added to the hypothesized tier until each intervener set contains at least one tier element (this ensures that the 2-factor in question is in fact absent on that tier in the input data).

The MT2SLIA is provably efficient, operating in polynomial time and data with respect to the size of the target grammar. However, the concept class it learns imposes two additional requirements on the MTSL languages that it can induce. Firstly, a given structure can be banned on at most one tier. Secondly, all tier elements (for each tier) must be *independent* from all non-tier elements for that tier (meaning they can freely occur with or without each other).

This second restriction introduces issues in the realm of syntax, where many theories of syntactic structure predict that syntactic elements are *not* independent of each other in this way. In particular, theories of syntax rely on the *universal spine* (aka hierarchy of projections), which is the idea that certain functional elements always show up in a certain order in syntactic trees. This inherently introduces exactly the kind of element dependence which cannot be represented by the subclass of MTSL which the MT2SLIA learns. Indeed, Swanson (2024a) demonstrates how the universal spine disrupts the ability of the MT2SLIA to correctly learn the English that-trace effect pattern, which can be represented with a fairly simple TSL-2 analysis (Graf, 2022c). This poses a challenge to the idea that MTSL with these restrictions is a good model of possible human languages, and underscores the need for algorithms which can learn the class of MTSL proper.

4 MTSL-BUFIA

The algorithm introduced in this section combines key insights from the MT2SLIA and from the *Bottom-Up Factor Inference Algorithm* (BUFIA)

introduced by (Chandlee et al., 2019). Similar to the MT2SLIA, this algorithm leverages the idea that any 2-factor which is not present in the input data must be absent because it is banned on at least one tier. Additionally, 2-paths are used to inform how these forbidding tiers should be constructed: each set of interveners for a given 2-factor *must* contain at least one element from each tier on which that 2-factor is banned. Unlike the MT2SLIA, however, this algorithm also leverages the fact that the space of possible forbidding tiers is partially ordered, and can therefore be traversed using a bottom-up breadth-first search, where the search space is pruned as constraints are located. This search strategy is inspired by (Chandlee et al., 2019)'s Bottom-Up Factor Inference Algorithm (BUFIA). Using this approach, the most general tiers on which 2-factors do not appear can be exhaustively located.

4.1 Canonical Form

It is possible for multiple distinct MTSL grammars to be extensionally equivalent, i.e., generate the same language. For this reason, I provide a canonical form for MTSL grammars.

Definition 4.1 (MTSL Canonical Grammar)**.** The Canonical Grammar for an MTSL language is a conjunction of 2-factor, tier pairs $G = \neg\langle f_0, T_0\rangle \wedge \neg\langle f_1, T_1\rangle \wedge ...$ for which the following hold:

1. $\forall \neg\langle f_i, T_i\rangle, \neg\langle f_j, T_j\rangle \in G[f_i = f_j \Longrightarrow T_i \notin T_j \wedge T_j \notin T_i]$
2. $\forall \langle f_i, T_i\rangle[L(G) \subseteq L(\neg\langle f_i, T_i\rangle) \Longrightarrow \exists T_i'[T_i' \subseteq T_i \wedge \neg\langle f_i, T_i'\rangle \in G]]$

This says that in order for an MTSL grammar to be canonical, 1) all forbidding tiers for the same factor must be incomparable, and 2) if the language generated by G obeys some constraint against a 2-factor f_i on some tier T_i, then there must be a constraint in G which bans f_i on T_i or some subset tier. The essential idea is that *all* surface-true constraints must be represented in the grammar in their most general form (i.e., on the smallest possible tier).

Lemma 1. *Any MTSL grammar is extensionally equivalent to a unique canonical MTSL grammar.*

Proof. Consider any MTSL grammar G. Suppose $L(G)$ obeys some constraint $\neg\langle f_i, T_i\rangle \notin G$. We can then construct a grammar $G' = G \wedge \neg\langle f_i, T_i\rangle$. Since $\neg\langle f_i, T_i\rangle$ is true over $L(G)$ and the two grammars differ only in the presence of this constraint, $L(G') = L(G)$. This process can be repeated to yield a G' for which $\nexists\ \neg\langle f_i, T_i\rangle[\neg\langle f_i, T_i\rangle \notin G' \wedge L(G') \subseteq L(\neg\langle f_i, T_i\rangle)]$ and $L(G') = L(G)$. G' then meets requirement 2) of a canonical grammar, since every surface-true constraint of $L(G)$ is present in G'.

Then, if $\exists \neg\langle f_i, T_i\rangle, \neg\langle f_i, T_j\rangle \in G'[T_i \subset T_j]$, we can construct $G'' = G' - \neg\langle f_i, T_j\rangle$. Since any form which contains the 2-factor f_i on the T_j tier will necessarily also contain f_i on the T_i tier, any langauge which obeys $\neg\langle f_i, T_i\rangle$ must also obey $\neg\langle f_i, T_j\rangle$. Therefore $L(G'') = L(G')$. Additionally, G'' still meets requirement 2) since the subset tier is always preserved. Once again, we can repeat this process until we reach a G'' for which $\nexists \neg\langle f_i, T_i\rangle, \neg\langle f_i, T_j\rangle \in G'[T_i \subset T_j]$ (i.e., requirement 1) holds) and $L(G'') = L(G') = L(G)$. G'' thus meets both requirements for a canonical MTSL grammar and is equivalent to G, meaning any MTSL grammar is equivalent to some canonical MTSL grammar.

Next, consider any canonical MTSL grammar G, and consider some other canonical MTSL grammar G' such that $L(G') = L(G)$. If $G' \neq G$, it must be the case that either G' contains some constraint which is not in G, or that G contains some constraint which is not in G'. Suppose G' contains a constraint $\neg\langle f_i, T_i\rangle$ which is not in G. Since $L(G) = L(G')$, $L(G)$ must obey this constraint. Since G is canonical, it must then (by requirement 2) contain some constraint $\neg\langle f_i, T_j\rangle$ such that $T_j \subset T_i$. But then, since G' is also canonical and must obey $\neg\langle f_i, T_j\rangle$, G' must contain some constraint $\neg\langle f_i, T_k\rangle$ such that $T_k \subset T_j$. But then by transitivity, $T_k \subset T_i$, meaning that G' violates requirement 1 and cannot be canonical. Thus, G' cannot contain any constraints which are not in G, and $G' \subseteq G$. Suppose G contains some constraint $\neg\langle f_i, T_i\rangle$ not in G'. By the same logic, G' must then contain some $\neg\langle f_i, T_j\rangle$, and G must contain some $\neg\langle f_i, T_k\rangle$ such that $T_k \subset T_j \subset T_i$. This means G violates requirement 1 and cannot be canonical. Thus, $G \subseteq G'$, and $G' = G$, meaning any canonical MTSL grammar is unique. □

4.2 BUFIA

To find the forbidding tier(s) for each 2-factor, MTSL-BUFIA uses a bottom-up, breadth-first search of the partially ordered set of possible tiers. This approach is inspired by BUFIA (Chandlee

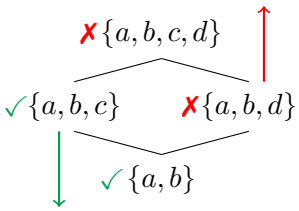


Figure 1: Hierarchy of possible forbidding tiers for ab over $\Sigma = \{a, b, c, d\}$. If ab is absent from a particular tier (denoted by ✗), it must be absent from all superset tiers. If it is present on a tier (denoted ✓) it is necesarily present on all subset tiers.

et al., 2019; Rawski, 2021), an algorithm designed to find the most general constraints over this type of partially ordered search space.

It is easy enough to see that the set of possible tiers (i.e., the powerset of Σ) is partially ordered under the subset relation. Furthermore, the set of constraints introduced by these possible tiers (for the same 2-factor) obeys that same partial ordering. For example, consider a toy language with alphabet $\Sigma = \{a, b, c, d\}$ with just one constraint: $\langle ab, \{a, b, d\}\rangle$. So strings like $adbcca$ and $bcaccdb$ (with tier projections $adba$ and $badb$) are in the language, but $daccbda$ is out, since its projection ($dabda$) includes the substring ab. It is immediately clear that no string in this language can contain ab on the superset tier of $\{a, b, c, d\}(= \Sigma)$, since this would necessarily mean ab is present on the $\{a, b, d\}$ tier as well. More generally, a factor f_i is absent from a tier T_i if and only if every possible occurrence of f_i is precluded by the intervention of some element from T_i. If $T_i \subset T_j$, then any factor absent from T_i will also be absent from T_j, since the intervention of an element from T_i entails the intervention of an element from T_j (all elements of T_i are in T_j). Therefore, any language which obeys the constraint $\langle f_i, T_i\rangle$ obeys all other constraints $\langle f_i, T_j\rangle$ where $T_i \subset T_j$. Figure 1 visualizes this property on the set of possible forbidding tiers for this toy language rooted at the $\{a, b\}$ tier.

These entailment relationships between possible constraints are exactly what BUFIA uses for learning. Starting from the "bottom", BUFIA proceeds upwards layer by layer, looking for constraints which are surface-true. When it finds them, it adds them to the grammar and prunes away the section of the search space above that new constraint. Chandlee et al. (2019) also outline several useful learning guarantees for this algorithm. Namely, BUFIA is guaranteed to find only and all incomparable constraints which are consistent with the input data, and it is guaranteed to find the *most general* such constraints. These are exactly the properties required to construct canonical MTSL grammars.

4.3 Algorithm

MTSL-BUFIA, defined in Algorithm 1, operates as follows: First, it finds all possible 2-factors which are *absent* from the input sample. This absence is an indicator that each of these 2-factors is forbidden on some tier(s). The algorithm then iterates through these missing 2-factors and computes the intervener sets for each. Once intervener sets are collected, the algorithm begins its bottom-up search for the smallest tier(s) on which that 2-factor is missing. It begins by looking at the tier consisting only of the symbols present in the 2-factor itself, and proceeds breadth-first to increasingly larger possible tiers. To determine whether a 2-factor is absent from a tier projection t, it must be the case that there is no intervener set i for that 2-factor such that $i \cap t$ is empty. In other words, each intervener set must contain *some* tier element, otherwise the 2-factor in question is present on that tier.

Any time the search encounters a tier where the 2-factor is not present in the data over that tier, the 2-factor, tier pair are added to the grammar, and *all* supersets of that tier are removed from the search space. In this way, the search space of possible tiers is pruned as the search proceeds.

Once this bottom-up search has been conducted for every 2-factor, the final grammar is returned.

5 Example

To demonstrate this algorithm in action, I will use a toy example language in which local and long-distance dependencies interact (this example is string-based, but recall that this can be smoothly generalized to tree structures as described in Section 2). The string language we will consider is defined by the following regular expression: $((ab^+c)|(eb^+d))^*$. This is all strings consisting of any number of sequences of one a followed by one or more bs followed by one c interspersed with any number of sequences of one e followed by one or more bs followed by one d. So $abbbcebd, ebbbbbd$, and $abcebdebdabbc$ are all valid strings in the language, but abd and $edac$ are not. This involves local restrictions, for example a and e can only be followed by b, but also non-local dependencies

Algorithm 1 MTSL-BUFIA

Data: Positive sample D
Result: Grammar G, a conjunction of ⟨2-factor, forbidding tier⟩ pairs.

G := {}
B := $2fac(\Sigma^*) - 2fac(D)$
foreach $f := \rho_1 R \rho_2 \in B$:
  $S := \{I$ **for** $\langle x, I\rangle \in paths(D)$ **where** $x = f\}$
  Q := [{ ρ_1, ρ_2 }]
  while Q ≠ []:
    T' = Q.$pop()$
    if $\exists \langle f, T\rangle \in G[T \subseteq T']$:
      continue
    if $\exists V \in S[T \cap V = \varnothing]$:
      Q.$append(NextSupersets(T'))$
    else:
      G = G∪¬$\langle f, T'\rangle$
return $\bigwedge_{c \in G} c$

which interact: b can be followed by c only in the case where the first b in its sequence was preceded by a, and similarly with d and e. This violates the "tier-element independence" requirement of the MT2SLIA, which stipulates that each element on each tier must be independent from (i.e., not bound to always occur next to) each non-tier element. However, in this language it is critical to enforce constraints like "a cannot be followed by d unless there is a e in between them". In this case, a, d, and e must all be tier elements. However, b must be off the tier, since its presence as an intervener between a and d does **not** license the structure. But none of these tier elements are independent from b: each is required to either precede or follow a b. Therefore, this language would not be learnable by the MT2SLIA. As we will see, however, the BUFIA-MTSL learner has no problem.

Suppose we are given the data presented in Example 1.

$$D = abbbc, ebd, abcebdebdebd, \\ ebdebdabcabcabbcebd, \\ abcebdabcebdabcebd \quad (1)$$

In the first step, the algorithm will compute the 2-factors which are missing from this sample. These are given in the first column of Table 2.

Then, for each missing 2-factor, the intervener sets will be computed. This is column 2 of Table 2.

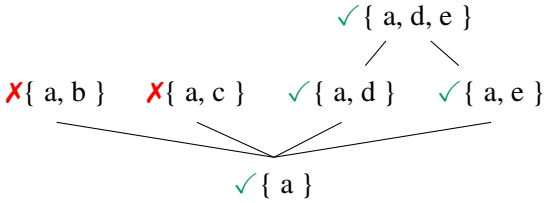


Figure 2: Caclulation of tiers for aa 2-factor.

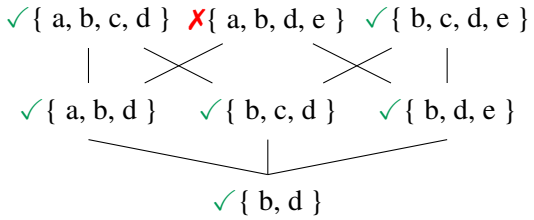


Figure 3: Calculation of tiers for be 2-factor.

Finally, the algorithm will conduct a bottom-up search of the possible tiers. The search procedure is diagrammed in Figures 2 and 3, for the 2-factors aa and db respectively. If each intervener set contains at least one tier element, the factor is missing on that tier, and the tier is added as a constraint (notated by the ✗ symbol). For example, in Figure 2, the $\{a, b\}$ tier is added as a forbidding tier for the aa 2-factor because each intervener set contains a b. This is exactly as expected–aa cannot be present on the $\{a, b\}$ tier because every a *must* be followed by at least one b. Similarly, each a must be followed by a c before another a can be present, and so this 2-factor is also banned on the $\{a, c\}$ tier.

For the db 2-factor, meanwhile, there must be *either* a or e occurring between them, but projecting just one to the tier would predict that that symbols is *always* required. For example, if db were banned on the $\{a, b, d\}$ tier, the (licit) sequence $ebdebd$ would be banned, since its tier projection would be $bdbd$, which contains the db 2-factor.

If a tier is not a forbidding tier (indicated with ✓), then the search continues upwards. Notice, however, that supersets of previously added forbidding tiers are *not* searched, so the final set $\{a, b, c, d, e\}$ (i.e., Σ) is never reached in Figure 3, and the search in Figure 2 can get no higher than $\{a, d, e\}$, since any other tiers that could be searched would be supersets of one of the existing forbidding tiers, $\{a, b\}$ or $\{a, c\}$.

The final forbidding tier(s) for all 2-factors are given in column three of Table 2.

In some ways, the grammar found by this al-

2-factor	interveners	tier(s)
aa	{ b, c }, { b, c, a }, { b, c, e, d }, { b, c, e, d, a }	{ a, b }, { a, c }
ac	{ b }, { a, b, c }, { b, c, e, d, a }	{ a, c, b }
ad	{ b, c, e }, { b, c, e, a }, { b, c, e, d }, { a, b, c, d, e }	{ a, d, b }, { a, d, c }, { a, d, e }
ae	{ b, c }, { b, c, a }, { b, c, a, e, d }	{ a, e, c }, { a, e, b }
ba	{ c }, { d }, { b, c }, { b, d }, { c, a, b }, { d, e, b }, { c, e, b, d }, { d, a, b, c }, { a, b, c, d, e }	{ a, b, c, d }
be	{ c }, { d }, { b, c }, { b, d }, { b, d, e }, { a, b, c }, { a, b, c, d }, { b, c, d, e }, { a, b, c, d, e }	{ b, c, d, e }
cb	{ a }, { e }, { a, b }, { b, e }, { a, b, c }, { e, b, d }, { a, b, c, d }, { e, b, d, a }, { a, b, c, d, e }	{ a, b, c, e }
cc	{ a, b }, { a, b, c }, { a, b, e, d }, { a, b, c, d, e }	{ c, a }, { c, b }
cd	{ e, b }, { e, b, d }, { a, b, c, e }, { a, b, c, d, e }	{ c, d, b }, { c, d, e }
db	{ a }, { e }, { a, b }, { b, e }, { a, b, c }, { e, b, d }, { a, b, c, e }, { a, b, d, e }, { a, b, c, d, e }	{ a, b, d, e }
dc	{ a, b }, { a, b, c }, { a, b, d, e }, { a, b, c, d, e }	{ d, c, a }, { d, c, b }
dd	{ e, b }, { e, b, d }, { e, b, a, c }, { a, b, c, d, e }	{ d, b }, { d, e }
ea	{ b, d }, { b, d, e }, { a, b, c, d, e }	{ a, b, e }, { a, d, e }
ec	{ b, d, a }, { b, d, a, e }, { b, d, a, c }, { a, b, c, d, e }	{ e, c, a }, { e, c, b }, { e, c, d }
ed	{ b }, { e, b, d }, { a, b, c, d, e }	{ e, d, b }
ee	{ b, d }, { b, d, e }, { b, d, a, c }, { a, b, c, d, e }	{ e, b }, { e, d }

Table 2: Missing 2-factors, intervener sets, and calculated tiers.

gorithm could be considered “inefficient", since it contains some constraints that are not strictly needed to enforce the target pattern. For example, there is no need for the constraint $\langle ad, \{a, b, d\}\rangle$, which enforces that a b must occur between a and d, since the other constraints already enforce that b is the only symbol which can follow a in the first place. This, however, is part of the definition of canonical grammar: if a constraint is true it *must* be represented in the grammar. This allows the grammar to be more universal by obviating the question of *which* constraint should be used for a given purpose.

6 Exact Identification in the Limit with Polynomial Data

The approach adopted here follows the tradition of grammatical inference, using the exact identification in the limit learning paradigm (Gold, 1967) with a polynomial bound on data (de la Higuera, 1997). Under this paradigm, the learner is presented with a *positive text* t of examples drawn from the target language L. The first n in t are denoted t_n. The examples can appear in any order, and may repeat, but for any given structure s in L, it is guaranteed that there is some finite $n \in \mathbb{N}$ such that $s \in t_n$.

In this sense, it is assumed that the learner will eventually receive a *representative sample* which characterizes L with respect to MTSL-BUFIA. Once this sample has been seen, the learner must 1) converge on the correct grammar and 2) not deviate from this grammar when more positive data is presented.

Definition 6.1 (Grammar Size)**.** For any MTSL-k gramamr G, its size is defined by:

$$|G| := \sum_{\neg\langle f,T\rangle \in G} k + |T|$$

Definition 6.2 (Representative Sample)**.** For a MTSL language L over alphabet Σ whose grammar is $G = \langle f_0, T_0\rangle \wedge \langle f_1, T_1\rangle \wedge ...\langle f_i, T_i\rangle$, a set D of structures is a *representative sample* iff all of the following hold:

1. $\forall x \in 2fac(\Sigma)[x \notin \{f : \exists\langle f, T\rangle \in G\} \Longrightarrow x \in 2fac(D)]$
2. $\forall\langle f, T\rangle \in G[\forall\sigma \in T\text{-}symbols(f)[\exists\langle f, V\rangle \in 2paths(D)[\sigma \in V \wedge \neg\exists\sigma' \in V[\sigma' \in T \wedge \sigma' \neq \sigma]]]]$
3. $\forall x \in \{f : \exists\langle f, T\rangle \in G\}[\neg\exists T'[(\langle x, V\rangle \in 2paths(D) \Longrightarrow V \cap T' \neq \varnothing) \wedge (\neg\exists\langle x, T\rangle \in G[T \notin T'])]]$

This essentially states that 1) a representative sample must contain all 2-factors which are not banned on any tier, 2) for each banned 2-factor and each tier on which it is banned, every tier element that is not part of the 2-factor itself must be present in an intervener set (for that same 2-factor) which contains no other tier elements, and 3) for each banned 2-factor there can be no set of elements which is "represented" (i.e., at least one element of this set is present) in every intervener set (for that 2-factor) but which is not itself a superset of a tier on which that 2-factor is banned.

Lemma 2. *For a MTSL language L, the size of the representative sample D for L is polynomial in the size of G for any MTSL grammar G of L.*

Proof. The first condition requires that all 2-factors which are not banned on any tier are present in the sample. The number of possible 2-factors will vary with the type of structure being used, but it will be limited to $c \cdot |\Sigma|^2$ where c is the number of possible relations (for trees this is two, while for strings it is just one). Embedding these 2-factors in well-formed structures (to form a data sample) may require the addition of extra "connecting" symbols (such as in the case for the *sibling* relation over trees, where an additional parent node is needed). Assuming the number of "connecting" symbols required is bounded by another constant k (which it is for both trees and strings), the total size will be $ck \cdot |\Sigma|^2$.

The second condition stipulates that each tier element of each restricting tier for each banned 2-factor must appear in some intervener set for that 2-factor without any other tier elements. Ensuring these intervener sets requires structures containing the two symbols present in the 2-factor, plus the tier element, plus any additional "connecting" symbols necessitated by the type of structure. These will contain some constant number of symbols (just 3 in the case of strings and 3 or 4 in the case of trees). Since one such structure is needed for each tier symbol, the amount data required to satisfy condition two is linear in the size of the grammar (since this is just the number of total tier symbols plus twice the number of 2-factors).

The third condition is about making sure that there are "small enough" intervener sets represented in the sample (i.e., those uncluttered by many non-tier elements). Constructing the smallest possible sample which fufills conditions 1 and 2 ensures this condition without the need for additional data.

Therefore, the space complexity of the representative sample is $O(|\Sigma|^2 + |G|)$ (where $|G|$ is the number of total symbols present in G). Assuming all alphabet symbols are used in the grammar, this is polynomial in the size of the grammar. □

Lemma 3. *Given any superset I of a representative sample D for a MTSL language L with canonical grammar $G = \neg\langle f_0, T_0\rangle \wedge \neg\langle f_1, T_1\rangle \wedge ... \wedge \neg\langle f_i, T_i\rangle$, MTSL-BUFIA will return a grammar $G' = G$.*

Proof. Consider the set B of all the 2-factors which are banned on any tier in G. Since I contains only valid structures in L, these 2-factors must all be absent from I. By definition of a representative sample, all other possible 2-factors over Σ must be present in D, and therefore also in I. B is therefore equivalent to the set which will be iterated over in the outer loop. Each $f_i' \in B$ must therefore be associated to one or more pairs $\langle f_i', T_1\rangle, \langle f_i', T_2\rangle ... \langle f_i', T_k\rangle \in G$. First, we show that every constraint in G' is in G. Suppose G' contains a forbidding tier for some f_i', $\langle f_i', T_i'\rangle$ which is *not* one of those in G. It must then be the case that each intervener set for f_i', contains at least one element from T_i', otherwise this pair would not get added to the grammar. Because $D \subset I$, the set of intervener sets for f_i', S must contain all intervener sets for f_i' which are present in D. By requirement 3 for representative samples, it must then be the case that there is some constraint $\neg\langle f_j, T_j\rangle \in G$ such that $T_j \subset T_i'$. Since the queue (Q) grows breadth-first, the algorithm will consider T_j as a forbidding tier for f_i' before it considers T_i'. Since $\neg\langle f_i', T_j\rangle$ holds on all the data, each intervener set in I for f_i must contain at least one element of T_j. Therefore, the algorithm will add $\neg\langle f_i', T_j\rangle$ to the grammar, and when it considers T_i' this tier will be discounted since $T_j \subset T_i'$ is already present as a forbidding tier for f_i'. Therefore, G' cannot contain any forbidding tiers for any f_i' which are not contained in G.

Next, we show that every constraint in G is in G'. Suppose G contains a forbidding tier for f_i, $\langle f_i, T_i\rangle$ which is not contained in G'. In order for T_i to *not* be added to G' in the inner loop, it must be the case that *either* a forbidding tier $\langle f_i, T_j\rangle$ is already in G', where $T_j \subset T_i$, *or* there is some intervener set V which contains no elements of T_i. As established above, the only constraints which will be added to G' are those which are also present in G. Therefore, $\langle f_i, T_j\rangle \in G' \Longrightarrow$

$f_i, T_j \in G$. However, if G were to contain $\langle f_i, T_j \rangle$ and $\langle f_i, T_i \rangle$, where $T_j \subset T_i$ it would not be a canonical grammar by the definition in Section 4. Since $\langle f_i, T_i \rangle \in G$, and I consists of only positive data generated by G, there can be no such intervener set V such that $V \cap T_i = \varnothing$, since this would mean f_i was in fact present on the tier T_i, thereby directly violating the constraint. Therefore, T_i will be added to G' during the inner loop, and G cannot contain any constraints not contained in G'.

Since $G \subset G'$ and $G' \subset G$, the two are equal: $G' = G$. □

Theorem 1. *For any MTSL language L, MTSL-BUFIA identifies the canonical grammar for L in the limit using polynomial data.*

Proof. From Lemmas 2 and 3. □

7 Time complexity

In the worst case, MTSL-BUFIA runs in exponential time. To see why, consider the time required for each step: The time complexity of computing 2-paths is $O(n^2)$ for strings (Jardine and Heinz, 2016) and $O(n^4)$ for trees (Swanson, 2024b). Then, for each missing 2-factor, the relevant tier(s) must be found. There are at most $|\Sigma|^2$ (or $2 \cdot |\Sigma|^2$ for trees) 2-factors to consider. Any missing 2-factor could be banned only on the segmental tier (ie $T = \Sigma$), and in this case the algorithm would have to traverse all possible tiers to discover this, with a time complexity of $2^{|\Sigma|}$ for each factor. This yields a time complexity of $O(|\Sigma|^2 \cdot 2^{|\Sigma|})$ The final step of traversing the powerset of the alphabet introduces exponential complexity to this algorithm, a disadvantage against the MT2SLIA which is guaranteed to run in polynomial time. The utility of BUFIA, however, does not come from its worst-case performance but rather from its ability to turn sparsity in the input data to its advantage. As in the example given in Section 5, when the input data obeys highly general constraints, BUFIA is able to prune away large chunks of the search space along the way, enabling it to tractably search very large spaces under the right conditions. Indeed, BUFIA has been successfully implemented and used to analyze natural-language scale data (Swanson et al., 2025). So although MTSL-BUFIA to some degree trades a tighter concept class for a worse time complexity, its exponential worst-case bound may not indicate intractability on natural language data, which is highly marked by sparsity. In the case of subregular syntax, many of the phenomena which have been analyzed as TSL require tiers with three or fewer symbols (ie highly restrictive), which would play into BUFIA's ability to prune the search space.

8 Conclusion

This paper introduces MTSL-BUFIA, an algorithm which exactly identifies the class of MTSL in the limit from a polynomially-sized data sample. This algorithm avoids the shortcoming of previous MTSL learning algorithms which placed additional problematic restrictions on the concept class which could be learned. The greater expressiveness of patterns that MTSL-BUFIA can induce comes at the cost of a polynomial runtime guarantee, but this does not mean the algorithm is universally intractable. The BUFIA approach does best (in terms of time complexity) with sparse data which can be captured by more general constraints.

Future work in this area involves testing MTSL-BUFIA on natural-language data samples, which would not only allow exploration of its average-case time performance (on the types of data found in human language), but also give insight into whether natural language typically furnishes the requisite representative sample to learn the target patterns.

Additionally, MTSL-BUFIA is suggestive of several possible extensions which could be fruitful to explore. Lambert (2021) provides a method for online learning of TSL-k languages, and it might be possible to adapt MTSL-BUFIA along those same lines. While MTSL-BUFIA is limited to the MTSL-2 languages, the concept of intervener sets can be extended to larger k-factors, allowing for the possibility of learning MTSL-k. With larger k values, the set of possible 2-factor, tier pairs is itself partially ordered and can be traversed in the same fashion to yield a grammar with mixed sizes of k-factors. Another useful aspect of BUFIA is that it is well-suited to feature-based representations, so this approach could also be easily extended to operate over features rather than segments.

Finally, another area of future exploration is into additional concept classes between MTSL and the more restricted class learned by the MT2SLIA. For example, what would it mean to enforce *only* constraint-uniqueness or *only* tier-element independence? Are there different restrictions on MTSL that are more appropriate for natural language? The

work presented in this paper offers a foundation on which to continue probing these open questions.

References

Jane Chandlee, Remi Eyraud, Jeffrey Heinz, Adam Jardine, and Jonathan Rawski. 2019. Learning with partially ordered representations. In *Proceedings of the 16th Meeting on the Mathematics of Language*, pages 91–101, Toronto, Canada. Association for Computational Linguistics.

Colin de la Higuera. 1997. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27:125–138.

E Mark Gold. 1967. Language identification in the limit. *Information and control*, 10(5):447–474.

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. 2022c. Typological implications of tier-based strictly local movement. In *Proceedings of the Society for Computation in Linguistics 2022*, pages 184–193.

Kenneth Hanson. 2023a. A TSL Analysis of Japanese Case. *Proceedings of the Society for Computation in Linguistics*, 6(1):15–24.

Kenneth Hanson. 2023b. A computational perspective on the typology of agreement.

Gunnar Ólafur Hansson. 2010. *Consonant harmony: Long-distance interactions in phonology*, volume 145. Univ of California Press.

Jeffrey Heinz. 2018. The computational nature of phonological generalizations. *Phonological typology, Phonetics and Phonology*, pages 126–195.

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*, pages 58–64, Portland, Oregon, USA. Association for Computational Linguistics.

Adam Jardine and Jeffrey Heinz. 2016. Learning tier-based strictly 2-local languages. *Transactions of the Association for Computational Linguistics*, 4:87–98.

Adam Jardine and Kevin McMullin. 2017. Efficient learning of tier-based strictly k-local languages. In *International conference on language and automata theory and applications*, pages 64–76. Springer.

Dakotah Lambert. 2021. Grammar interpretations and learning tsl online. In *International Conference on Grammatical Inference*, pages 81–91. PMLR.

Dakotah Lambert. 2023. Relativized adjacency. *Journal of Logic Language and Information*, 32:707–731.

Kevin McMullin, Alëna Aksënova, and Aniello De Santo. 2019. Learning phonotactic restrictions on multiple tiers. *Proceedings of the Society for Computation in Linguistics*, 2(1):377–378.

Kevin McMullin and Gunnar Ólafur Hansson. 2014. Long-distance phonotactics as tier-based strictly 2-local languages. In *Proceedings of the annual meetings on phonology*.

Jonathan Rawski. 2021. *Structure and Learning in Natural Language*. Ph.D. thesis, Stony Brook University.

James Rogers and Dakotah Lambert. 2019. Some classes of sets of structures definable without quantifiers. In *Proceedings of the 16th Meeting on the Mathematics of Language*, pages 63–77.

Logan Swanson. 2024a. Representing syntax for learning.

Logan Swanson. 2024b. Syntactic learning over tree tiers. In *Proceedings of the ESSLLI 2024 Student Session*.

Logan Swanson, Jeffrey Heinz, and Jonathan Rawski. 2025. Phonotactic learning with structure, not statistics. Submitted to Linguistic Inquiry remarks replies.