Syntactic Learning over Tree Tiers

Logan Swanson

Stony Brook University

Abstract. This paper presents a polynomial time learning algorithm for a subclass of the *multi tier-based strictly 2-local* (MTSL-2) tree languages. MTSL-2 tree languages are useful in modeling syntactic phenomena in natural language. I define the novel subclass of *2-factor MTSL-2* tree languages, which still captures many syntactic dependencies but is simpler and more parallel to the definition of MTSL-2 over strings, allowing existing learning algorithms for MTSL-2 string languages to be easily generalized to this class of tree languages. Although this algorithm is not intended to be a literal acquisition model, it delivers a key learnability result for subregular syntax, and demonstrates that string-based learning algorithms can be usefully generalized to tree languages.

Keywords: Formal language theory · Formal learning · Computational syntax · Tree languages · Subregular linguistics · Tier projection

1 Introduction

Linguistics is driven by two central questions: what is the character of linguistic knowledge, and how is it acquired from limited data? The program of *subregular linguistics* (see [12], [14], and references therein) has identified classes of formal languages sufficiently expressive to account for phenomena in phonology and morphology but also sufficiently limited to be efficiently learned from positive data. There is also evidence that syntactic dependencies over trees are subregular by virtue of belonging to the class of *multi tier-based strictly 2-local tree languages* (MTSL-2) [6, 7]. This class is finite for a fixed alphabet, and is therefore learnable in the limit by enumeration, as described by Gold [5]. This paper, however, provides a more efficient algorithm for learning a linguistically relevant subclass of MTSL-2 which runs in polynomial time and requires a polynomial-sized characteristic sample of well-formed trees.

Related work on learning syntactic structures is largely in the realm of probabilistic approaches [1, 3] or the formal learning of context-free grammars and some relevant superclasses directly from strings [2]. In contrast to these approaches, the algorithm in this paper is deterministic and receives trees as input rather than strings. It may seem odd to assume that the input for the learner includes the actual tree structures, and there is certainly a relevant future direction in addressing how to learn the tree structures themselves from linear input. However, it is worth noting that what this paper delivers is a *formal learning result*, rather than an explicit theory of language acquisition. The algorithm presented here proves that a relevant sublass of tree MTSL-2 can be learned from positive data, and offers mathematical guarantees about time complexity and requisite input sample. Moreover, the tree structures in subregular syntax are similar to feature-annotated dependency trees and thus look very similar to the structure of basic semantics, i.e. predicate-argument relations. Thus, the input a human learner receives may indeed provide tree structures roughly along these lines.

This paper uses subregular language classes to connect learning results across linguistic domains: the class of MTSL is linguistically important over both strings and trees. As such, the algorithm presented here represents an important step towards a unified theory of language learning. The paper is laid out as follows: Sec. 2 introduces the mathematical preliminaries, including a formalization of trees and MTSL-2 tree languages. Section 3 defines the tree substructure of 2-paths and the class of 2-factor MTSL (2FMTSL), which are leveraged in Sec. 4 to adapt the string MTSL-2 learning algorithm of McMullin et al [15] to 2FMTSL tree languages. In addition to defining the learning algorithm, this section demonstrates its behavior on a detailed example based on natural language. The algorithm is built around identifying missing substructures and calculating the smallest tier on which they are absent. Section 5 shows how these principles work together to ensure the algorithm's correctness and efficiency. Section 6 concludes with some remarks on the larger significance of this result and pointers for future work.

2 Preliminaries

Trees. We define trees in terms of *Gorn domains.* A *Gorn address* $a \in \mathbb{N}^*$ is a string of natural numbers, and a Gorn domain is a set D of Gorn addresses such that for every string uj with $u \in \mathbb{N}^*$ and $j \in \mathbb{N}$, it holds that $uj \in D$ implies both $u \in D$ (mother-of closure) and $ui \in D$ for every $0 \leq i < j$ (left sibling closure). The Gorn address of the root is always the empty string ε. A *Σ-tree* is a pair $\langle D, \ell \rangle$ with Gorn domain D and a labeling function $\ell : D \to \Sigma$. The set of all Σ-trees is denoted $\mathbb{T}_\Sigma$.

Tree relations. We define a few additional relations given $u, v, w \in \mathbb{N}^*$, $q \in \mathbb{N}^+$, and $i < j \in \mathbb{N}$: $u \lhd ui$ (*immediate dominance*), $u \lhd^+ uq$ (*proper dominance*), $ui \prec u(i+1)$ (*immediate left sibling*), $uiv \prec ujw$ (*precedence*).

Tree tiers. Let $T \subseteq \Sigma$ be a *tier alphabet.* Then $u \lhd_T w$ (*immediate tier dominance*) iff $\ell(u), \ell(w) \in T$ and $u \lhd^+ w$ and there is no v such that $\ell(v) \in T$ and $u \lhd^+ v$ and $v \lhd^+ w$. For example, if $T := \{a, b, c\}$, then in Fig. 1 the root ε immediately tier dominates 00 but not 000 — and it both immediately dominates and tier dominates 1. Tier dominance thus is proper dominance relativized to symbols in the tier alphabet T (cf.[14]). Furthermore, v is a *left tier sibling* of w ($v \prec_T^+ w$) iff $v \prec w$ and there is some u such that $u \lhd_T v$ and $u \lhd_T w$. The string of the labels of all tier daughters of u ordered by $\prec_T^+$ is also called the *tier daughter string* $u \lhd_T$

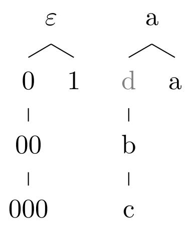


Fig. 1. Example tree showing Gorn addresses (left) and labels (right)

of u (if u has no tier daughters, then $u\lhd_T = \varepsilon$). In the example in Fig. 1, the tier daughter string of the root node ε is ba. For mathematical convenience, we stipulate that there is a distinguished root marker $\rtimes \notin \Sigma$ such that for all $v \in D$ with $\ell(v) \in T$, it holds that $\rtimes \lhd_T v$ iff there is no $u \in D$ with $u \lhd_T v$.

(M)TSL-2. Given $T \subseteq \Sigma$, we call $c_T : T \cup \{\rtimes\} \to \wp(T^*)$ a *tier daughter string constraint function.* A Σ-tree $\langle D, \ell\rangle \in \mathbb{T}_\Sigma$ is well-formed with respect to c_T iff for every $u \in D$ with $\ell(u) \in T$, it holds that $u\lhd_T \in c_T(\ell(u))$. A tree language $L \in \mathbb{T}_\Sigma$ is *tier-based strictly 2-local* (TSL-2) iff there is some c_T such that L is the set of all Σ-trees that are well-formed with respect to c_T [9]. It is *multi tier-based strictly 2-local* (MTSL-2) iff it is the intersection of one or more TSL-2 tree languages.

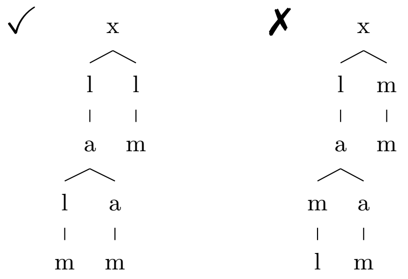


Fig. 2. Example of well-fored tree (left) and ill-formed tree (right)

Example 1. Let $\Sigma := \{l, m, a, x\}$, $T := \{l, m\}$, and c_T as follows: $c_T(\rtimes) := l^*$, $c_T(l) := l^*ml^*$, and $c_T(m) := l^*$. Then the Σ-tree on the left in Fig. 2 is well-formed with respect to c_T, but the one on the right is not because: I) the tier daughter string of $\rtimes$ is lm, which is not in $c_T(\rtimes)$, II) the set of tier daughter strings of nodes labeled m includes the string m, which is not in $c_T(m)$, and III) the set of tier daughter strings of nodes labeled l includes the strings mm and ε, which are not in $c_T(l)$. This example is inspired by the subregular discussion of Minimalist syntax in [7]. Intuitively, c_T enforces that every landing site l must have exactly one mover m among its tier daughters, and every m must be the tier daughter of some l.

Now let $T' := \{l, m, a\}$ with $c_{T'}(\rtimes) := c_{T'}(m) := \{l, m\}^*$, $c_{T'}(l) := \{l, m, a\}^*$, and $c_{T'}(a) := \{a, m\}^+$. Here $c_{T'}$ enforces that a can only appear as a tier daughter of l or another a, that it cannot be a leaf node on the tier projection, and that it cannot have l among its tier children. In isolation, these constraints may seem strange, but in tandem with the constraints on movers and landing sites outlined above, they enforce a pattern where a can only occur along a path between a mover and its corresponding landing site. This is inspired by extraction morphology phenomena like Irish wh-agreement, where some heads, e.g. complementizers, may appear with a special form a, but only if a mover moves across them [6]. The set of all Σ-trees well-formed with respect to both c_T and $c_{T'}$ is MTSL-2. It contains all trees such that there is a one-to-one match between movers and (closest) landing sites, and a only occurs along such movement paths.

The space of TSL-2 tree languages captures the typology for many syntactic phenomena, including verb agreement [11], case assignment [10], and movement patterns [8]. As demonstrated by the above example, MTSL-2 is able to capture the interactions of multiple TSL-2 phenomena, making it a good candidate to model the complexity of human languages.

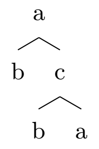
3 2-Paths for MTSL-2

To learn the MTSL2 languages, the algorithm in Sec. 4 constructs tiers for each potential constraint by identifying illicit substructures which can be made licit by allowing certain elements to "intervene" in the middle of the substructure. The "interveners" which impact licitness form the tier for that constraint. To identify these elements, it is helpful to introduce a tree-based generalization of the notion of *2-paths* given by Jardine and Heinz [13].

Given $t := \langle D, \ell \rangle \in \mathbb{T}_\Sigma$, $x, y \in \Sigma$, $T \subseteq \Sigma$ and some defined relation $R_T \in \{\lhd_T, \prec_T^+\}$ over D, $x\ R_T\ y$ is an R_T *2-factor* of t on tier T iff there are nodes $u, w \in D$ such that $\ell(u) = x$, $\ell(w) = y$, and $u\ R_T\ w$ (note that if T is unspecified, it is taken to be equal to Σ, and $\lhd_T$ and $\prec_T^+$ reduce to $\lhd$ and $\prec$ in this case). For any tree t, we define $2fac_T(t)$ as the set of all 2-factors present in t on tier T. A *2-path* of t is a pair $\langle \ell(u)\ R\ \ell(v), I_R(u,v)\rangle$ ($R \in \{\lhd, \prec\}$) that consists of a "possible" 2-factor and a set $I_R(u,v)$ of intervening symbols. The intervener set contains all the symbols which would need to be removed from the tree in order for the 2-factor in question to be present. For $u \lhd^+ w$ ($u, w \in \mathbb{N}^*$), $I_\lhd(u,w) := \{\ell(x) \mid u \lhd^+ x \text{ and } x \lhd^+ w\}$. That is to say, the set of dominance interveners for u and w is the set of all labels that occur along the path properly between u and w. And for $uiv \prec ujw$ ($u, v, w \in \mathbb{N}^*$, $i \neq j \in \mathbb{N}$), $I_\prec(uiv, ujw) := \{\ell(x) \mid u \lhd^+ x \text{ and either } (x \lhd^+ uiv \text{ or } x \lhd^+ ujw) \text{ or } (uiv \prec x \text{ and } x \prec ujw)\}$. In other words, the set of sibling interveners for u and w is the set of all labels which are dominated by the least common ancestor of u and w and which either occur on the direct path between u and w or are to the right of u and to the left of w.

An example tree and its corresponding 2-paths are shown in Tab. 1. Note that the same 2-factor may appear in multiple 2-paths: in this example, $b \prec a$ appears once with no interveners (corresponding to the nodes labeled b and a which are immediate siblings), and once with c and b as interveners (corresponding to the higher b node and the same a node).

For any language L, we define define $2paths(L)$ as the set of all 2-paths which are present in that language.



2-paths:		$\langle b \prec a, \varnothing\rangle$
$\langle a \lhd b, \varnothing\rangle$	$\langle c \lhd b, \varnothing\rangle$	$\langle b \prec a, \{b, c\}\rangle$
$\langle a \lhd b, \{c\}\rangle$	$\langle c \lhd a, \varnothing\rangle$	$\langle b \prec b, \{a\}\rangle$
$\langle a \lhd c, \varnothing\rangle$	$\langle a \lhd a, \{c\}\rangle$	$\langle b \prec c, \varnothing\rangle$

Table 1. Example tree and 2-paths

Using these substructures, we can define a subclass of tree MTSL-2 which more closely mirrors string MTSL-2:

Definition 1 (2-Factor MTSL). *A tree language $L \subseteq \mathbb{T}_\Sigma$ is 2-factor MTSL (2FMTSL) iff it can be defined by a conjunction of pairs $G := \langle T_0, f_0 \rangle \wedge \langle T_1, f_1 \rangle \wedge ... \langle T_n, f_n \rangle$ such that $L = \{w \in \mathbb{T}_\Sigma : \forall 0 \leq i \leq n, f_i \notin 2fac_{T_i}(w)\}$*

In addition, L must meet the following criteria:

1. ***Constraint uniqueness:** For any 2-factor $f_i \notin 2fac(L)$, there is exactly one tier T_i such that $\langle T_i, f_i \rangle \in G$.*
2. ***Tier-element independence:** For each 2-factor f_i banned on tier T_i, it must be the case that* I*) each tier symbol which is not part of f_i is attested in an intervener set for f_i in $2paths(L)$, and* II*) each smallest intervener set in which those tier symbols are present (for f_i in $2paths(L)$) contains only other tier elements which do not themselves show up in smaller intervener sets.*

Every 2FMTSL language is an MTSL-2 language where each tier-daughter string function is strictly 2-local [16]: each banned factor $x \lhd y$ indicates that y is a banned symbol in $c_T(x)$, and each banned factor $x \prec y$ indicates that xy is a banned substring $c_T(u)$ for all $u \in T$. However, a 2FMTSL grammar can *only* enforce constraints on 2-factors which are illicit on a particular tier, without regard to daughter string languages for particular symbols. This is further restricted to languages where no 2-factor needs to be banned on incomparable tiers (constraint 1), all tier elements are independent from all non-tier elements (i.e., no tier element is limited to only showing up next to a non-tier element), and any dependency between tier elements is symmetrical (constraint 2). Both of these constraints are necessary for this learning approach because a core assumption of the algorithm in Sec. 4 is that a given 2-factor is banned on a single tier (this is critical for establishing polynomial time complexity). If 2-factors need to be banned on multiple incomparable tiers, or if there are non-tier elements which always occur in tandem with tier elements, it makes it impossible to accurately construct a single tier with the correct members. This qualification is important to investigate further in future work, especially in the domain of syntax where this kind of dependence between elements which must always appear together is much more common than in phonology.

The grammar for a 2FMTSL language is a basis for a set of MTSL-2 languages: it provides a foundational set of tier projections and constraints, over which any type of symbol-specific daughter string constraints can be layered.

4 Learning Algorithm

With these preliminaries in place, I turn to the central result of this paper: a learning algorithm for the class of 2FMTSL tree languages. The Multi Tier-based 2-Strictly Local Inference Algorithm (MT2SLIA), defined in Algorithm 1, builds directly on the string MTSL-2 algorithm in [15]. The MT2SLIA leverages the idea that any 2-factor which is absent in the input sample must be banned on *some* tier, and uses the notion of 2-paths to systematically construct this tier.

The algorithm proceeds by iterating through all 2-factors which are absent in the input data. For each of these, the tier is initialized to the symbols present in the 2-factor. Then, all the attested intervener sets for this 2-factor are considered in batches by ascending cardinality. In each batch, all elements of each set which does not contain any tier elements are added (simultaneously) to the tier. The final tier is added to the grammar, paired with the corresponding 2-factor.

Algorithm 1 MT2SLIA

```
G := {}
B := $2fac(\mathbb{T}_\Sigma) - 2fac(Input)$
**foreach** $f := \rho_1 R \rho_2 \in B$:
    $T_i := \{\rho_1, \rho_2\}$
    $V := \{I$ **for** $\langle x, I\rangle \in 2paths(Input)$ **where** $x = f\}$
    **for** cardinality $c$ in $\{1, \ldots, |\Sigma|\}$:
        $V_c := \{s \in V$ **where** $|s| = c\}$
        $N := \{\}$
        **foreach** $v \in V_c$:
            **if**: $\neg\exists\sigma[\sigma \in v \wedge \sigma \in T]$
            **then**: $N := N \cup v$
        $T := T \cup N$
    $G := G \cup \{\langle T_i, \rho_1 R \rho_2\rangle\}$
**return** $\bigwedge_{p \in G} p$
```

I will illustrate how this algorithm learns a specific 2FMTSL language. Recall the MTSL-2 language L from example 1: $t \in L$ iff t is well-formed with respect to both c_T (one-to-one match between movers m and closest landing sites l) and $c_{T'}$ (a can only occur along movement paths). We will learn an 2FMTSL superset grammar for this language, which generates a close approximation of the true pattern, but cannot enforce that l nodes have *exactly one* m tier daughter, only that two m nodes cannot be *immediate* tier siblings. This constraint, which is string TSL-2, could be learned using the tiers output by MT2SLIA by applying any TSL-2 string learning algorithm to the tier daughter strings of l.

Suppose we are given the sample in Fig. 3, as well as (for compactness, since these phenomena are symmetric) the mirror images of those trees.

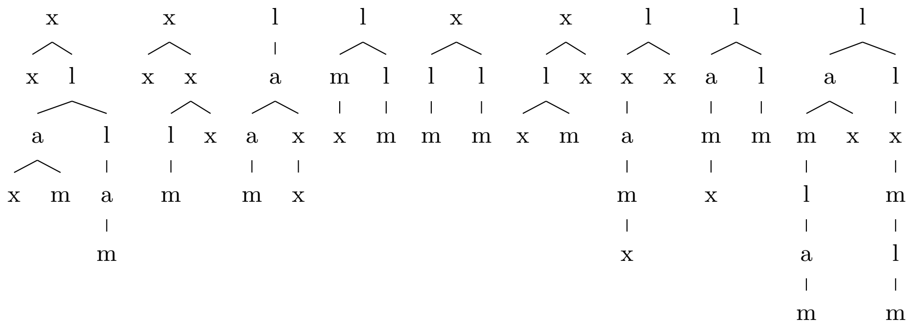


Fig. 3. Input sample for learning the MTSL-2 language in example 1

We first determine which 2-factors are absent in the input, shown in Tab. 2.

$\rtimes \lhd l$	$(\rtimes \lhd m)$	$(\rtimes \lhd a)$	$\rtimes \lhd x$				
$l \lhd l$	$l \lhd m$	$l \lhd a$	$l \lhd x$	$l \prec l$	$l \prec m$	$l \prec a$	$l \prec x$
$m \lhd l$	$(m \lhd m)$	$(m \lhd a)$	$m \lhd x$	$m \prec l$	$(m \prec m)$	$(m \prec a)$	$m \prec x$
$(a \lhd l)$	$a \lhd m$	$a \lhd a$	$a \lhd x$	$a \prec l$	$(a \prec m)$	$(a \prec a)$	$a \prec x$
$x \lhd l$	$x \lhd m$	$x \lhd a$	$x \lhd x$	$x \prec l$	$x \prec m$	$x \prec a$	$x \prec x$

Table 2. Set of all possible immediate dominance and immediate left sibling 2-factors, with unattested ones in parenthesis

Then, we calculate the tier for each unattested 2-factor $\sigma_1 R \sigma_2$ starting from $T := \{\sigma_1, \sigma_2\}$. Tab. 3 shows the evaluation of all intervener sets for each 2-factor, ordered by size. All elements of intervener sets which do not contain any existing tier elements are inserted (in batches by intervener set size, which will be essential to estabilishing polynomial sample size in Sec. 5) into T. At each step, the constraint and the tier will be added to G. The final grammar consists of all 2-factors from Tab. 3 paired with their corresponding final tier.

2-factor	Interveners				Tier	
$\rtimes \lhd m$	$\{l\}$	~~$\{l,x\}$~~ ~~$\{l,a\}$~~	~~$\{l,x,a\}$~~ ~~$\{l,m,x\}$~~ ~~$\{l,m,a\}$~~	~~$\{l,m,x,a\}$~~	$\{m\}$	$\{m,l\}$
$\rtimes \lhd a$	$\{l\}$	~~$\{l,x\}$~~ ~~$\{l,a\}$~~	~~$\{l,a,m\}$~~		$\{a\}$	$\{a,l\}$
$m \lhd m$	$\{l\}$	~~$\{l,a\}$~~			$\{m\}$	$\{m,l\}$
$m \lhd a$	$\{l\}$				$\{m,a\}$	$\{m,a,l\}$
$a \lhd l$	$\{m\}$				$\{a,l\}$	$\{m,a,l\}$
$m \prec m$	$\{l\}$	~~$\{l,a\}$~~	~~$\{l,a,x\}$~~	~~$\{l,m,a,x\}$~~	$\{m\}$	$\{m,l\}$
$m \prec a$	$\{l\}$	~~$\{l,x\}$~~	~~$\{l,m,x\}$~~	~~$\{l,m,a,x\}$~~	$\{m,a\}$	$\{m,a,l\}$
$a \prec m$	$\{l\}$	~~$\{l,x\}$~~	~~$\{l,m,x\}$~~	~~$\{l,m,a,x\}$~~	$\{m,a\}$	$\{m,a,l\}$
$a \prec a$	$\{l\}$				$\{a\}$	$\{a,l\}$

Table 3. Tier calculation for each missing 2-factor. Grayed-out elements are those which are already on the tier by the time that intervener set is considered. Sets containing such elements (crossed out) are not added to the tier.

In the final grammar, the $\{m,l\}$ constraints ($\rtimes \lhd m$, $m \lhd m$, and $m \prec m$) ensure that each mover must be dominated by a landing site, and two movers cannot be consecutive tier-children of the same landing site (two movers for a single target). In addition, an agreeing head can only appear below a landing site, but never above one unless a mover intervenes. The $\{m,a,l\}$ sibling constraints broaden the restrictions against multiple movers for a single target by preventing *any* adjacent tier-sibling movement path elements (m or a).

By leveraging the properties of 2FMTSL, the MT2SLIA is able to learn the bulk of the constraints needed to enforce multiple interacting TSL patterns of the type found in natural language.

5 Identification in Polynomial Time and Data

In this section, I establish that the MT2SLIA constitutes a conclusive learnability result by proving that it identifies the class of tree 2FMTSL in the limit from positive data in the sense of Gold [5], with polynomial bounds on time and data as per de la Higuera [4]. The proof relies on establishing a representative sample and demonstrating that it characterizes the language with respect to the MT2SLIA.

Definition 2 (Representative Sample). *For a 2FMTSL language L over alphabet Σ whose grammar is $G = \langle T_0, f_0 \rangle \wedge \langle T_1, f_1 \rangle \wedge \ldots$ a set D of trees is a representative sample iff all of the following hold:*

1. $\forall x \in 2fac(\mathbb{T}_\Sigma)[x \notin \{f : \exists \langle T, f \rangle \in G\} \implies x \in 2fac(D)]$
2. $\forall \langle T, f \rangle \in G[\forall \sigma \in T - symbols(f)[\exists \langle f, V \rangle \in 2paths(D)[\sigma \in V \wedge \neg \exists \langle f, V' \rangle \in 2paths(L)[\sigma \in V' \wedge |V'| < |V|\,]]]]$

This essentially states that a representative sample must contain all 2-factors which are not banned on any tier, and that for each banned 2-factor, each symbol of the tier on which it is banned (except those already present in the 2-factor) must be attested as an intervener for that 2-factor as part of a smallest intervener set that it could be part of.

Lemma 1. *For a language L, the size of the representative sample D for L is polynomial in the size of G for any grammar G of L.*

Proof. The first condition requires that any 2-factor which is not banned on some tier is present in the sample. There are at most $2 \cdot |\Sigma|^2$ such 2-factors, each of which can be constructed with at most 3 nodes (inserting a shared parent node for sibling 2-factors), giving a total size of $6 \cdot |\Sigma|^2$

The second condition requires that for each banned 2-factor, each tier symbol from the tier on which it is banned must be attested as an intervener in the smallest possible intervener set it can be contained in. Since there are at most $2 \cdot |\Sigma|^2$ banned 2-factor and $|\Sigma|$ symbols on any tier, this condition imposes an additional $6 \cdot |\Sigma|^3$ space requirement.

Taken together, the space complexity of the representative sample is $O(|\Sigma|^3)$, and therefore polynomial in both the size of the alphabet and the size of the grammar (assuming all alphabet symbols are used in the grammar).

Lemma 2. *Given an input sample I of size n, tree-MT2SLIA runs in polynomial time in the size of n.*

Proof. The algorithm will first need to compute 2-factors and 2-paths for I. As discussed in Sec. 2, 2-paths can be computed using Gorn addresses. Gorn addresses can be mapped to nodes in a single tree traversal (i.e., linear time in the number of nodes). Each pair of nodes must be considered both as possible dominance 2-factors and possible sibling 2-factors, and all other nodes must be considered as possible interveners. Membership of each node in the intervener set can be decided in linear time at worst, by comparing each symbol in the Gorn addresses of the relevant nodes (the maximum length of a Gorn address is the size of the tree). Thus, the time complexity of computing 2-paths is $O(n^4)$.

Once the 2-paths have been computed, finding the attested 2-factors is trivial, since these are simply the 2-factors from any 2-path where the intervener set is the empty set.

Then, the algorithm will execute the outer for-loop. This loop will run at most $2 \cdot |\Sigma|^2$ times. Σ itself is bounded by the size of the input (in the worst case, each node will have a different label), and so we will just use n going forward. The two inner loops together iterate through all intervener sets (just once, even though these are two loops. The middle loop is just handling bucketing by size.), of which there are at most n^2 (one for each pair of nodes). Thus, the for-loop runs in $O(n^4)$ time.

In total, the algorithm runs in $O(n^4 + n^4) = O(n^4)$ time.

Lemma 3. *Given any superset I of a representative sample D for a 2FMTSL tree language L with grammar $G = \langle T_0, f_0 \rangle \wedge \langle T_1, f_1 \rangle \wedge \ldots \wedge \langle T_i, f_i \rangle$, the MT2SLIA 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 trees 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. Thus, the set which will be iterated over in the outer loop, $B' = B$. Each 2-factor $f_i' \in B'$ will be associated to some tier T_i'. Since $B' = B$, f_i' must also be part of some pair $\langle T_j, f_i' \rangle \in G$. T_i' will be determined by finding all intervener sets for f_i', and considering them in groups from the smallest cardinality to the largest. T_i' will be the union of all the intervener sets for which no element of that set shows up in a smaller intervener set. By definition of the representative sample, each element of T_j must show up in the smallest possible intervener set. Thus, each element of T_j will be added to T_i'. Furthermore, no matter what data is in I, no symbol which is not part of T_j can be attested in an intervener set *without* a member of T_j (otherwise this would be an illicit construction). Since all elements in T_j are guaranteed to show up in intervener sets without any non-tier elements (which are consequently smaller), they will have been added to T_i' already by the time any intervener set containing a non-tier element is considered. Therefore, T_i' will contain *all* and *only* the elements of T_j, i.e. $T_i' = T_j$. Since $B' = B$, $G' = \langle T_0, f_0 \rangle \wedge \langle T_1, f_1 \rangle \wedge \ldots \wedge \langle T_i, f_i \rangle = G$.

Theorem 1. *For any 2FMTSL tree language L, the MT2SLIA identifies a grammar for L in polynomial time and data.*

Proof. From lemmas 1, 2, and 3.

6 Conclusion

This paper introduces tree-MT2SLIA which is guaranteed to induce a 2FMTSL grammar over trees in polynomial time and data, and defines the required representative sample. The class of 2FMTSL captures many of the patterns described in the subregular syntax literature, and can be combined with existing string-TSL2 learning algorithms to capture the remainder. In tandem with this fact, the learnability results given in this paper make the (2F)MTSL tree languages a viable candidate for representing the space of possible human languages. Next steps for this work include formalizing how these algorithms can fit together, and also connecting this approach with existing strategies for learning tree structure from surface strings to offer a complete picture of syntactic learning. This algorithm demonstrates the utility of subregular linguistics in finding unified learning strategies that generalize across different linguistic domains, and marks a step towards understanding how humans learn language.

7 Acknowledgements

The work carried out for this project was supported by the National Science Foundation under Grant No. BCS-1845344.

References

1. Bod, R.: From exemplar to grammar: Integrating analogy and probability in language learning. Cognitive Science **33**(4), 752–793 (2008)
2. Clark, A.: Beyond Chomsky normal form: Extending strong learning algorithms for PCFGs. In: International Conference on Grammatical Inference. pp. 4–17. PMLR (2021)
3. Clark, A.: Strong learning of probabilistic tree adjoining grammars. Proceedings of the Society for Computation in Linguistics **4**(1), 406–414 (2021)
4. De La Higuera, C.: Characteristic sets for polynomial grammatical inference. Machine Learning **27**, 125–138 (1997)
5. Gold, E.M.: Language identification in the limit. Information and control **10**(5), 447–474 (1967)
6. Graf, T.: Diving deeper into subregular syntax. Theoretical Linguistics **48**(3-4), 245–278 (2022)
7. Graf, T.: Subregular linguistics: bridging theoretical linguistics and formal grammar. Theoretical Linguistics **48**(3-4), 145–184 (2022)
8. Graf, T.: Typological implications of tier-based strictly local movement. In: Proceedings of the Society for Computation in Linguistics 2022. pp. 184–193 (2022)
9. Graf, T., Kostyszyn, K.: Multiple wh-movement is not special: The subregular complexity of persistent features in minimalist grammars. Proceedings of the Society for Computation in Linguistics **4**, 275–285 (2021)
10. Hanson, K.: A TSL Analysis of Japanese Case. Proceedings of the Society for Computation in Linguistics **6**(1), 15–24 (2023)
11. Hanson, K.: A computational perspective on the typology of agreement (2023), http://www.kennethhanson.net/files/hanson-nyubb2023-agreement-slides.pdf
12. Heinz, J.: The computational nature of phonological generalizations. Phonological typology, Phonetics and Phonology pp. 126–195 (2018)
13. Jardine, A., Heinz, J.: Learning tier-based strictly 2-local languages. Transactions of the Association for Computational Linguistics **4**, 87–98 (2016)
14. Lambert, D.: Relativized adjacency. Journal of Logic Language and Information **32**, 707–731 (2023)
15. McMullin, K., Aksënova, A., De Santo, A.: Learning phonotactic restrictions on multiple tiers. Proceedings of the Society for Computation in Linguistics **2**(1), 377–378 (2019)
16. McNaughton, R., Papert, S.A.: Counter-Free Automata (MIT research monograph no. 65). The MIT Press (1971)