Pushing the Boundaries of TSL Languages

Anonymous ACL submission

Abstract

Recent research in computational linguistics suggests that unbounded dependencies in phonotactics, morphology, and even syntax can all be captured by the class of Tier-based Strictly Local languages (TSL). Here, grounded in the fact that there are phonotactic processes that cannot be captured by a simple tier-based account, I explore new classes of sub-regular languages obtained by relaxing a particular constraint on the tier-projection mechanism of TSL grammars.

1 Introduction

The complexity of linguistics dependencies has been a topic of great interest in the last decade. While many dependencies in phonology can be captured by *local constraints*, a recent fruitful hypothesis has been that unbounded dependencies can be captured by the class of Tier-based Strictly Local languages (TSL).

The idea of a phonological tier was already present in representational theories of phonology, for example to describe tone systems (Goldsmith and Club, 1976), but Heinz et al. (2011) were the first to argue that a formal notion of tier could be used to capture long-distance dependencies. Thus, the class of TSL languages was introduced, in which a tier is defined as the projection of a subset of the segments of the input string, and the strictly-local grammar constraints act only over that subset. This paper addresses some of the formal properties of TSL languages which in the past have been used to evaluate typological claims, and explores possible extensions of the TSL class that are grounded in the need to describe problematic phonotactic patters. By modifying TSL grammars' projection mechanism, I show how these classes are capable of capturing these outlier processes, and study the increase in generative power we get through different changes to the definition of tier-projection.

The paper is structured as follows. Section 2 presents phonotactic processes problematic for the TSL account. Section 3 introduces essential mathematical notation. Section 4 is an overview of a subregion of the Chomsky Hierarchy known as the sub-regular hierarchy. Moreover, it discusses several equivalent characterizations of TSL languages, and of their intersection closure. Section 5, and 6 present new extensions of TSL, and relate each new class to the rest of the sub-regular hierarchy. Section 7 discusses the implications of these results for linguistics and learnability.

2 Linguistic Motivation

The primary motivation for studying the properties of the TSL neighborhood in the Sub-regular Hierarchy comes from its relevance to the study of phonotactic patterns in natural languages.

As already mentioned, many dependencies in phonology can be captured by *local constraints* that only make distinctions on the basis of contiguous subsequences of segments up to some length k. For example, a (k=2) local dependency requiring /s/ to surface as [z] when followed by [l] can be captured by a grammar that only contains the sequence *zl* (or, alternatively, the negative constraint *sl). However, (unbounded) long-distance dependencies do not fit this patter, and have been characterized instead as *tier-based strictly local*.

A full introduction to the formal properties of TSL is given in Section 4.1 but, intuitively, a tier is defined as the projection of a subset of the segments of the input string, and the grammar constraints are characterized as the set of sequences of length k not allowed on the tier. For instance,

the example in Figure 1 (from AARI, an Omotic language of south Ethiopia) shows how to enforce long-distance sibilant harmony (in anteriority) by projecting a tier T only containing sibilants from the input, and ban contiguous ʒs and sʒ on T (c.f. (Hayward, 1990)).

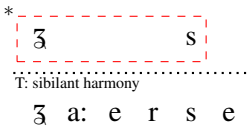

Figure 1: Example of sibilant harmony over tier from AARI.

However, several processes have been reported that struggle with the way TSL currently combines a projection mechanism used to filter relevant segments, and strictly local dependencies. In particular, there are cases in which it is necessary to check some local properties of a segment – in addition to membership to the tier-alphabet – *before* projecting it on a tier.

For example, Baek (2016) discusses the case of unbounded stress in EASTERN CHEREMIS and DONGOLESE NUMBIAN, in which the primary stress falls on the right-most non-final heavy syllable or – if there is none – on the initial syllable. Baek shows how no TSL grammar is able to generate this patter, which should allow well-formed strings like **ĹLLLH**, while also ruling out ill-formed strings like ***ĹLLLHH**. The problem lies in the fact that, in order to generate only the well-formed strings, a grammar would need to check the position of a syllable before projecting it on a tier. Similarly, McMullin (2016) reports a case of sibilant harmony in the SAMALA language of Southern California, in which a regressive sibilant harmony with unbounded locality ([s] and [ʃ] may not co-occur anywhere within the same word) overrides a restriction against string-adjacent $^*st$,$^*sn$,$^*sl$ that results in a pattern of dissimilation (see also (Applegate, 1972) for the original data set). For example /sn/ surfaces as [ʃn], unless there is an [s] following in the string, in which case it surfaces as [sn].

The example in Figure 2 shows why it is not possible to capture the overall pattern for sibilant harmony in SAMALA with a single TSL grammar. Since [sn] is sometimes observed in a string-adjacent context, it must be permitted as a 2-factor on a tier that includes all segments (even though

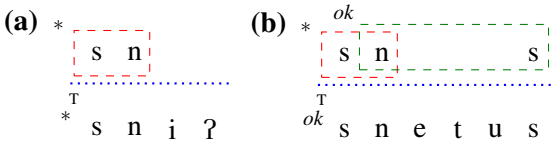

Figure 2: Example from SAMALA: **(a)** is ill-formed because of adjacent $^*ac$; **(b)** is well-formed since ac is followed by another s later in the string, but it is still ruled out by the grammar.

it is only permitted when a segment such as [s] follows them later in the string). However, since [sn] is allowed to occur in such contexts – and the number of possible other /n/ in the string is potentially unbounded – then a TSL grammar would have no means of banning *[sn] when there is no subsequent [s] in the string.

Other examples of this kind of processes can be found in YAKA, where [+*nasal*] segments start an harmony domain only if they are not immediately followed by voiced oral stops (Walker, 2000), or among culminativity patterns (cf. (Heinz, 2014)). For all these processes, it seems that a TSL grammar would need to be able to access additional structural information that is available in the input string, in order to decide whether to project a segment on a tier or not. In Section 5, I show that this kind of expressivity can be accomplished by increasing the locality window of the *projection mechanism* associated to the notion of tier.

Finally, McMullin (2016) and De Santo (2017) discuss a set of patterns that can be accounted for only by a conjunction of TSL grammars. Interestingly, for most of these processes, there is always one tier-alphabet that is the superset of all the others involved in the conjunction. Section 6 explores this idea by studying a class of languages allowed to project subsets of tiers in an embedded fashion.

3 Preliminaries

Familiarity with set notation is assumed. Given a finite alphabet Σ , the set of all possible finite strings of symbols from Σ and the set of strings of length $\leq n$ are Σ^* and $\Sigma^{\leq n}$, respectively. A language L is a subset of Σ^*, which we also refer to as a Σ-language. For all strings w, $|w|$ denotes the length of the string, while λ is the unique empty string. Left and right word boundaries are marked by the symbol $\rtimes$ and $\ltimes$ respectively. A string u is a *k-factor* of a string w iff $\exists x,y \in \Sigma^*$ such that

$w = xuy$ and $|u| = k$. As defined in Heinz et al. (2011), the function F_k maps words to the set of k-factors within them.

$$F_k(w) := \{u : u \text{ is a } k\text{-factor of } w\}$$

For example, $F_2(aab) = \{aa, ab\}$. The set of prefixes of w is $Pref(w) = \{p \in \Sigma^* | (\exists s \in \Sigma^*)[w = ps]\}$ and the set of suffixes is $Suff(w) = \{s \in \Sigma^* | (\exists p \in \Sigma^*)[w = ps]\}$. For all $w \in \Sigma^*$ and $n \in \mathbb{N}$, if $|w| \geq m$ $Suff^n(w)$ is the single suffix of w of length n, otherwise $Suff^n(w) = w$. For any given string w, $P_{\leq k}(w)$ is a function that maps w to the set of subsequences up to length k in w. Following Oncina and García (1991), for any function $f : \Sigma^* \to T^*$ and $x \in \Sigma^*$, the *tails* of x with respect to f are defined as:

$$tails_f(x) := \{(y, v) | f(xy) = uv \wedge lcp(f(x\Sigma^*))\}$$

where $lcp(S)$ is the longest common prefix of a set of strings S.

Finally, familiarity with general notions of formal logic is also expected, and I restrict the discussion to the specific signature of first-order logic that is used in this paper. We allow standard Boolean connectives ($\wedge$, $\vee$, $\neg$, $\rightarrow$), and first-order quantification ($\exists$, $\forall$) over individuals. We let $x \prec y$ denote *precedence*, $x \approx y$ denote *identity*, and x, y denote variables ranging over positions in a finite string $w \in \Sigma^*$. The remaining logical connectives are obtained from the given ones in the standard fashion, and brackets may be dropped where convenient. For example, *immediate precedence* is defined as $x \lhd y \leftrightarrow x \prec y \wedge \nexists z[x \prec z \wedge z \prec y]$. We can always assume that for each $\sigma \in \Sigma$ and each unary predicate P the truth-value $P(\sigma)$ must be well-defined. For each label $\sigma \in \Sigma$ we include a predicate P_σ such that if x is a position in w with label $\sigma \in \Sigma$, then $P(x)$ is defined by $P(\sigma)$. In other words, we add a dedicated predicate for each label we wish to use. We let $P_\sigma(x) = \sigma(x)$. Classical results on definability of strings represented as finite first-order structures are then used(McNaughton and Papert, 1971). If $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$, then a string $w \in \Sigma^*$ can be represented as a structure M_s in the signature $(P_{\sigma_1}, \ldots, P_{\sigma_n}, \prec)$. If φ is a logical formulae (a *sentence* of some logic), we use $L(\varphi) = \{w \in \Sigma^* | M_s \vDash \varphi\}$ to stand for the stringset extension of φ .

4 The Sub-regular Hierarchy

Due to space constraints, it is here impossible to give a full characterization of each of the classes in the sub-regular hierarchy. Thus, this section just gives the definition of the Strictly Local (SL), Locally Threshold Testable (LTT), and Strictly Piecewise (SP) languages. For a more complete discussion of the properties of these classes, the reader is referred to McNaughton and Papert (1971), Rogers and Pullum (2011), Rogers et al. (2013), Rogers et al. (2010), among others. I will instead discuss in more detail the Tier-based Strictly Local class (Heinz et al., 2011), as well as its intersection closure (MTSL, (De Santo, 2017)) since they are central to our results.

Definition 1. (Strictly k-Local) A language L is Strictly k-Local iff there exist a finite set $S \subseteq F_k(\rtimes \Sigma^* \ltimes)$ such that

$$L = \{w \in \Sigma^* : F_k(\rtimes w \ltimes) \subseteq S\}$$

A language L is Strictly Local iff it is stricly k-local for some $k \in \mathbb{N}$. For example, the set of strings $(ab)^n$ is a strictly 2-local language licensed by the grammar $G = \{\rtimes a, ab, ba, b \ltimes\}$, and $Sigma = \{a, b\}$. Basically, SL languages describe patterns which depend solely on the relation between consecutive symbols in a string.

Rogers and Pullum (2011) define *Suffix Substitution* (also k-Local Substring Substitution) closure as a property of strictly local languages:

Definition 2. (Suffix Substitution Closure) A language L satisfies k-local suffix substitution closure iff for all strings u_1, v_1, u_2, v_2, there exists $k \geq 1 \in \mathbb{N}$ such that for any string x of length $k-1$, if $u_1 \cdot x \cdot v_1$, $u_2 \cdot x \cdot v_2 \in L$, then $u_1 \cdot x \cdot v_2 \in L$.

In fact, Strictly Local languages can be uniquely characterized in terms of it:

Theorem 1. *A language is strictly k-local SL_k iff it satisfies k-local suffix substitution closure.*

Suffix substitution is then an excellent way of showing that a language is not strictly local.

Definition 3. (Locally t-Threshold k-Testable) A language L is Locally t-Threshold k-Testable iff $\exists t, k \in \mathbb{N}$ such that $\forall w, v \in Sigma^*$, if $F_{k,t}(w) = F_{k,t}(v)$ then $w \in L \Leftrightarrow v \in L$.

Intuitively LTT grammars can, for some fixed k, restrict the number of occurrences of any k-factor in a string. Moreover, they can distinguish which symbols occur at which positions. Practically, LTT languages can *count*, but only up to some fixed threshold t, since there is a fixed finite bound on the number of positions a given grammar can distinguish.

Definition 4. (Piecewise k-Testable) A language L is Piecewise k-Testable iff $\exists k \in \mathbb{N}$ such that $\forall w, v \in \Sigma^*$, if $P_{\leq k}(w) = P_{\leq k}(v)$ then $w \in L \Leftrightarrow v \in L$.

A language is Piecewise Testable if it is Piecewise k-Testable for some k. Essentially, Piecewise languages are sensible to relationships between segments based on *precedence* (over arbitrary distances) instead than *adjacency* (immediate precedence).

4.1 Tier-based Strictly Local

Tier-based Strictly Local languages (Heinz et al., 2011) are an extension of SL languages, where k-local constraints only apply to elements of a tier $T \subseteq \Sigma$. An erasing (string projection) function is introduced to delete all the symbols that are not in T. Consider a string $w \in \Sigma^*$, for every $\sigma_1 ... \sigma_n \in w$ the erasing function $E_T(\sigma_i)$ is:

$$E_T(\sigma_i) := \begin{cases} \sigma_i & \text{if } \sigma_i \in T \\ \lambda & \text{otherwise} \end{cases}$$

Then, a TSL language is defined as follows.

Definition 5. (TSL Language) A language L is Strictly k-Local on Tier T iff there exists a tier $T \subseteq \Sigma$ and a finite set $S \subseteq F_k(\rtimes T^* \ltimes)$ such that

$$L = \{w \in \Sigma^* : F_k(\rtimes E_T(w) \ltimes) \subseteq S\}$$

where S is the set of all the permissible k-factors on tier T.

In other words, Definition 5 states that a language L is Tier-based Strictly Local iff it is Strictly k-Local on tier T, for some tier $T \subseteq \Sigma$ and $k \in \mathbb{N}$. Note that the erasing function defines an *Input Strictly Local* mapping as described by Chandlee (2014).

Definition 6. (Input Strictly Local Function) A function f is Input Strictly Local (ISL) if there is a k such that for all $u_1, u_2 \in \Sigma^*$, if $Suff^{k-1}(u_1) = Suff^{k-1}(u_2)$ then $tails_f(u_1) = tails_f(u_2)$.

The definition above makes clear that given an ISL function, the output for input symbol σ only depends on the last $(k-1)$ input symbols.

Proposition 1. The erasing function of a TSL language is Input Strictly Local with $k = 1$.

Proof. Consider the erasing function $E_T(\sigma_1 ... \sigma_n)$ as defined above. The mapping of each σ to the output string is 1-local, since it only depends on whether $\sigma_i \in T$ or not. Then, $\forall w \in \Sigma^*$: $Suff^0(w) = \lambda$ and $tails_{E_T}(w) = \Sigma^* \times (T \cup \lambda)^*$, and it is trivially true that $Suff^{k-1}(w_1) = Suff^{k-1}(w_2) \Rightarrow tails_{E_T}(w_1) = tails_{E_T}(w_2)$. Hence, E_T is ISL. □

In the most general case, a TSL grammar can be decomposed into the cascade of an ISL transducer over Σ^* – computing the erasing function as described above – combined with a Strictly Local filter applied to its output, and a final transducer $\mathscr{D}$ mapping the SL tier-language to an output language over Σ. The automaton $\mathscr{D}$ performing the transduction back from the tier-language is shown in Figure 3.

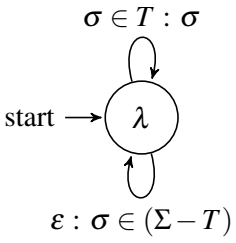

Figure 3: Automaton for transducer $\mathscr{D}$

This characterization gives us interesting insights into the relation between tier-projection and strict locality, and will help simplify some of the proofs introduced later in the paper.

4.2 Multiple-Tier Strictly Local Languages

Starting from the consideration that TSL languages are not closed under intersection, De Santo (2017) introduces *Multiple Tier Strictly Local (MTSL)* languages as a generalization of TSL based on a more explicit formal characterization of the conjunction operation over tiers. Informally, MTSL can be defined as TSL languages projecting over multiple tiers and enforcing a – potentially different – set of constraints for each tier. Then, a string is well-formed for the language iff each substring is well-formed on every projected tier. Interestingly, the MTSL class is exactly the intersection closure of TSL.

Definition 7. (Multiple-Tier Strictly Local) A language L is Multiple-Tier Strictly Local (MTSL) iff there exists a finite set of tiers $\mathscr{T} = \{T_i : T_i \subseteq \Sigma, 0 \leq i \leq n \in \mathbb{N}\}$ such that L is Strictly k-local on every tier $T_i \in \mathscr{T}$.

The following two Theorems sum up the most of the properties of MTSL languages that are relevant to the current work. The reader is referred to

the original paper for proofs of these results.

Theorem 2. *The class of MTSL languages is not closed under union, relative complement, and relabelings.*

Theorem 3. *TSL $\subset$ MTSL, MTSL $\subset$ SF, MTSL is incomparable to LTT and PT.*

Notation-wise, we write that L is k-MTSL$_n$, where $n = |\mathscr{T}| \in \mathbb{N}$ is the number of tiers needed to generate L, and k is the locality of the TSL grammar. Since this notation make the relationship between MTSL and TSL explicit, it is also possible to refer to any MTSL language simply by k-TSL$_n$.

4.3 Logical Definability and Sub-regularity

In an attempt to provide alternative characterizations of classes in the Sub-regular Hierarchy, Rogers and Pullum (2011) use different types of logic (conjunction of negative literals, propositional logic, FO, and MSO) and the kind of relation used to handle the order of elements in a string (*precedence* and *immediate precedence*) as parameters to define the separate sub-regular regions.

This approach establishes a link between the expressive power of language classes, and the power of different logics – which is already well understood outside of formal language theory. For example, it is well-known that the languages described by monadic second-order (MSO) logic formulas are precisely the regular languages. When the logic is first-order (FO), then the languages that arise are precisely the star-free languages (that is, those that can be obtained from a set of unary predicates by using the operations of union, complement, and concatenation (McNaughton and Papert, 1971)).

Rogers and Pullum (2011) show that strictly local grammars can be described by *conjunction of negative literals*: formulae in propositional logic using only $\wedge$ and $\neg$ as connectives (see also (Heinz, 2015), i.a.). For example, consider the strictly local language $L := (ab)^+$, which can be described with a negative grammar S^* listing all the k-grams that may not occur in a string: $\{^*\rtimes b, ^*bb, ^*aa, ^*\ltimes a\}$.

A string satisfies the property p_S iff it contains the k-gram p of grammar S. So the strings that are well-formed with respect to S^* are those for which the following holds:

$$\phi = \neg \rtimes b \wedge \neg bb \wedge \neg aa \wedge \neg \ltimes a$$

This statement is a formula of propositional logic, where each k-gram corresponds to a proposition p, also called a literal.

More succinctly then, a Strictly k-Local grammar S can be written as a formula

$$\phi = \bigwedge_{p \in S} \neg p$$

where

$$p := \exists x_1 \ldots x_k \left[\left(\bigwedge x_i \neq x_{i+1} \right) \wedge \left(\bigwedge x_i \lhd x_{i+1} \right) \right]$$

It is easy to extend this characterization to TSL languages. Let $T \subseteq \Sigma$ be a tier-alphabet, and let $t(x)$ denote the predicate associated to a label $t \in T$ marking the string element x. We can define *tier-precedence* as the binary predicate:

$$\begin{aligned} x \lhd_T y \leftrightarrow & T(x) \wedge T(y) \wedge x \prec y \\ & \wedge \neg \exists z \, [T(z) \wedge x \prec z \wedge z \prec y] \end{aligned}$$

and

$$T(x) \leftrightarrow \bigvee_{t \in T} t(x)$$

At this point, a TSL language can be characterized as a FO formula substituting *tier-precedence* to standard *immediate precedence* in the definition of p.

By extending the FO definition of TSL, it is also possible to characterize MTSL grammars as:

$$\phi = \bigwedge_{T \in \mathscr{T}} \left(\bigwedge_{p \in S_T} \neg p \right)$$

Interestingly, the FO characterization directly relies on the fact that each MTSL is the intersection of distinct TSL grammars.

5 Structure-Sensitive TSL Languages

Essentially, the projection mechanism of TSL is based on each segment's 1-local properties. This Section presents a new class of grammars – *Structure-Sensitive TSL (SS-TLS)* – that rely on structural relationships between segments in the input string to choose what to project. This is accomplished simply by allowing the projection function E_T of a TSL grammar to be ISL$_{k \geq 1}$ instead than just ISL_1, and makes it possible to account for the controversial patterns mentioned in Section 2.

5.1 Definition

Definition 8. (Structure-Sensitive TSL) A language L is Structure-Sensitive Tier-based Strictly Local (SS-TSL) with *tier sensitivity* m, iff there exist an erasing function E_T which is an Input Strictly m-Local mapping of any string w to a tier $T \subseteq \Sigma$, and exists $S \subseteq F_k(\rtimes T^* \ltimes)$ such that

$$L = \{w \in \Sigma^* : F_k(\rtimes E_T(w) \ltimes) \subseteq S\}$$

Notation wise, we write L is k-SS-TSL^m (or simply, k-TSL^m) if there exists an ISL$_m$ erasing function E_T such that L is strictly k local on the tier T output of E_T. A TSL erasing function with $m > 1$ can also be referred to as *structural projection*. As an example, assuming $\Sigma = \{a, b\}$, Figure 4 presents an ISL automaton for the erasing function E_T with structure sensitivity $m = 2$, projecting on the tier only as in initial/final position.

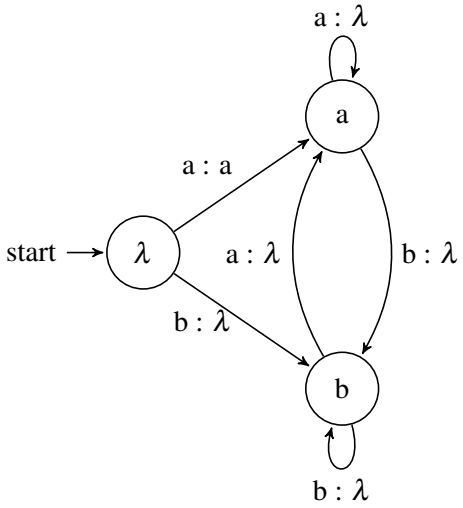

Figure 4: ISL$_2$ automaton for tier of first/last a

SS-TSL languages can be characterized in terms of FO logic by adapting the notion of *tier-membership* used in the characterization of tier-precedence introduced before. For every segment, we need to re-define what it means to belong on a tier based on its m-local properties.

$$T(x) \leftrightarrow \left(\bigvee_{t \in T} t(x) \right) \wedge \exists y_1 \ldots y_{m-1} \left[\bigwedge_{1 \le i \le m-1} \left(\bigvee_{\gamma \in \Gamma} \gamma(y_i) \right) \right]$$

where Γ is the set of predicates representing the structural properties to consider in oder to project a segment on a tier. To give a practical example, suppose we want a SS-TSL^m language such that it projects a tier of initial/final as. Then:

$$T(x) \leftrightarrow a(x) \wedge \exists y[(\rtimes(y) \wedge y \lhd x) \vee (\ltimes(y) \wedge x \lhd y)]$$

For ease of exposition, I will refer to tier-precedence employing this notion of tier-membership with $\blacktriangleleft_T$. Then, a SS-TSL language is described by a FO formula as the one given for TSL, using $\blacktriangleleft_T$ instead of $\lhd_T$ to define the tier.

5.2 Relations to other Sub-regular Classes

Having provided a complete characterization of SS-TSL languages, this section establishes their position inside the sub-regular hierarchy. In particular, I will show that SS-TSL generalizes TSL in a different way than MTSL, while still remaining a proper sub-set of Star-Free.

Theorem 4. *TSL languages $\subset$ Structure-Sensitive TSL.*

Proof. This follows directly from Proposition 1 which states that the erasing function of TSL languages is in fact an ISL mapping with $m = 1$. □

Lemma 5. *SS-TSL languages $\nsubseteq$ MTSL.*

Proof. It suffices to give an example of a language that is SS-TSL, but it is not MTSL$_n$ for any $n \in \mathbb{N}$.

Assume $\Sigma = \{a, b\}$, and consider a language L_{FL} such that, for instance, $abab, baba \in L_{FL}$ but $aba \notin L_{FL}$. It is easy to see that L_{FL} can be generated by an SS-TSL^2 grammar with $T = \{a : \rtimes a \vee a \ltimes\}$ and $S = \{^*aa\}$ via the erasing function in Figure 4, but that $L_{FL} \notin$ MTSL$_n$ for any number of possible tiers. We can reason as follows. To ban aba, an MTSL grammar would have to project at least one tier T_i containing a and enforcing $^*aa$. Since the erasing function for MTSL is ISL$_1$, every a in the input will be on T_i. Thus, no matter how many tiers we project, there will always be at least one on which $abab$ is ill-formed. □

Lemma 6. *MTSL languages $\nsubseteq$ SS-TSL.*

Proof. Again, it is possible to give an example of a language that is MTSL but not SS-TSL^m for any m. Assume $\Sigma = \{a, b, c, d\}$, and consider a language $L_{block} \in$ MTSL $_2$ such as:

$$T_1 = \{a, b\}, S_{T_1} = \{ab, bb, ba\}$$

$$T_2 = \{a, c\}, S_{T_2} = \{aa, cc, ca\}$$

For example, $caba \in L_{block}$ but $caca, cabca \notin L_{block}$. It should be easy to see that L_{block} is not SS-TSL. In order to enforce $^*aa$ and $^*ac$ while allowing ab, it is necessary to have a tier $T_{block} = \{a,b,c\}$. However, once b and c are on the same tier, there is nothing in the local properties of the input string that can be used to include ab^ma and bca while banning ab^mca. Hence, $L_{block} \notin$ SS-TSL^m for any $m \in \mathbb{N}$. □

Theorem 7. *The class of MTSL languages and the class of SS-TSL languages are incomparable.*

Proof. The fact follows immediately from Lemma 6 and Lemma 7. Moreover, the two classes are obviously not disjoint, since TSL languages are both MTSL and SS-TSL. □

Theorem 8. *SS-TSL is incomparable to LTT, PT.*

Proof. That SS-TSL $\nsubseteq$ LTT, PT follows from the fact that it includes TSL, which is neither.

To see why LTT is not a subclass of SS-TSL, consider $\Sigma = \{a,b,c\}$ and a sentential logic formula $\varphi : aa \rightarrow bb$ such that $L(\varphi) = \{w \in \Sigma^* | w \vDash \varphi\}$. Thus, $aabb, acbcacbcc, babbb \in L$ but $aaaaa, cacacaccaa \notin L(\varphi)$. This language is 2-LT, and clearly describes patterns that require more than local constraints. Thus, following the strategy used for the counter-examples in Lemma 5 and Lemma 6, it should be easy to see that it is not SS-TSL^m independently of the locality of the structural projection.

For PT $\nsubseteq$ MTSL, consider the 3-SP language L_P that allows for strings like ab^k and b^ka but not ab^ka. This language is not SS-TSL^m, for any m. □

Finally, the next result follows naturally from the possibility to define SS-TSL tiers as first-order predicates just from precedence.

Lemma 9. *SS-TSL* $\subset$ *Star-Free* [1].

5.3 Combining Multiple Tiers and Structural Sensitivity

The fundamental insight in De Santo (2017) is that the intersection closure of TSL is obtained by simply allowing a TSL grammar to exploit multiple projection functions. Intuitively, we can get the intersection closure of SS-TSL (namely, SS-MTSL) in the same way, by minimally extending these classes in order to allow for multiple tier-projection.

In fact, we simply need to define for every $T_i \in \mathscr{T}$ an ISL$_m$ erasing function E_{T_i} such that L is strictly local on T_i. We can write that L is k-SS-MTSL^m_n (or simply k-TSL^m_n).

Definition 9. (Structure-Sensitive MTSL) A language L is Structure-Sensitive Multiple-Tier Strictly Local (SS-MTSL) with *tier sensitivity m*, iff there exists a finite set of tiers $\mathscr{T} = \{T_i : T_i \subseteq \Sigma, 0 \leq i \leq n \in \mathbb{N}\}$, and for every $T_i \in \mathscr{T}$ there exist an erasing function E_{T_i} which is an Input Strictly m-Local mapping of any string w to a tier $T_i \subseteq \Sigma$, and $S_{T_i} \subseteq F_k(\rtimes T_i^* \ltimes)$ such that

$$L = \{w \in \Sigma^* : F_k(\rtimes E_{T_i}(w) \ltimes) \subseteq S_{T_i}, \forall T_i \in \mathscr{T}\}$$

Clearly, it is also possible to characterize SS-TSL languages in terms of FO logic, just by substituting, in the formula used for MTSL, *tier-precedence* $\blacktriangleleft_T$ as defined for SS-TSL.

The following result clarifies that SS-TSL and SS-MTSL are in a proper subsumption relationship.

Theorem 10. *SS-TSL* $\subset$ *SS-MTSL*

Proof. That $SS-TSL \subseteq SS-MTSL$, follows directly from the definition by allowing $\mathscr{T} = \{T\}$. For an example of a language that is SS-MTSL but not SS-TSL. Consider a grammar over $\Sigma = \{a,b,c,d\}$ where $T_1 = \{a,c\}$ and filters $^*aa$, whereas $a \in T_2$ and $b \in T_2$ iff $c \lhd b$ and requires the tier to be $(ab)^+$. □

As for the relation with other sub-regular classes, that MTSL is a subclass of SS-MTSL is trivial, MTSL being the set of languages in SS-MTSL with *tier-sensitivity* $m = 1$.

Lemma 11. *MTSL* $\subset$ *SS-MTSL*

Moreover, from the fact that SS-MTSL grammars are FO definable follows that they are also Star Free.

Lemma 12. *SS-MTSL* $\subset$ *SF.*

Finally, the following result derives the relation of SS-MTSL to the rest of the hierarchy.

Theorem 13. *The class of SS-MTSL languages is incomparable to LTT, SP.*

[1] By definition, SS-TSL are TSL languages with an erasing function that is ISL$_{m\geq 1}$ instead than just ISL$_1$. Intuitively, then, this result also clarifies how none of the TSL's proofs presented by (Heinz et al., 2011) hinges on the locality of E_T

Proof. That SS-MTSL $\nsubseteq$ LTT, PT follows from the fact that SS-MTSL properly extends MTSL and SS-TSL, and from Theorem 8. For the other direction, we can simply refer to the counter-examples used in Theorem 8, which are not SS-MTSL independently of the number of tiers the grammar is allowed to project. □

6 Tier-Embedding Strictly Local Languages

Having introduced structural projection, we may wonder what happens if we allow for an erasing function that is sensible to the new local relationships introduced by projecting a tier. This is the intuition behind the class of languages discussed in this Section, which has interesting relations to the other classes in the TSL neighborhood.

6.1 Definition

Definition 10. (Tier-Embedding SL) A language L is Tier-Embedding Strictly k-Local (TESL) iff there exists a set of tiers $\mathscr{T} := \{T_i : T_i \subseteq \Sigma, 0 < i \leq n \in \mathbb{N}\}$ and for each T_i there exists a finite set $S_{T_i} \subseteq F_k(\rtimes T_i^* \ltimes)$ such that

$$L = \{w \in \Sigma^* : F_k(\rtimes E_{T_i}(w) \ltimes) \subseteq S_{T_i}, \forall T_i \in \mathscr{T}\}$$

where S_{T_i} represents all the permissible k-factors on tier T_i, and the erasing function for each tier is an Input Strictly m-Local function defined as:

$$E_{T_i} : T_{i-1} \to T_i$$
$$E_{T_1} : \Sigma \to T_1$$

In other words, Definition 10 characterizes the erasing function for TESL in terms of a cascade composition of ISL mappings from tier to tier. The set of tiers is restricted in such a way that there is an ordering of application upon the ISL functions, and only the first one has as its domain the original input string. The domain of every other function E_{T_i} is the output of $E_{T_{i-1}}$. We write L is k-TESL^m_n, where k is the locality of the constraints imposed on each tier, n is the number of embedded tiers, and m is the locality of the projection functions.

TESL languages can also be characterized in terms of FO formulae, by refining the definition of tier-precedence one last time.

$$x <_{T_n} y \leftrightarrow x \blacktriangleleft_{T_n} y \wedge x <_{T_{n-1}} y$$

where $T(\cdot)$ is defined in $x \blacktriangleleft_{T_i}$ as for SS-TSL languages.

Clearly, $x <_{T_1} y \leftrightarrow x \blacktriangleleft_{T_1} y$. Once this change is the definition of $\lhd_T$ is in place, any TESL language can be described by an FO formula similar to the one used for MTSL:

$$\phi = \bigwedge_{1 \leq i \leq n} \left(\bigwedge_{p \in S_{T_i}} \neg p \right)$$

6.2 Relations to other Sub-regular Classes

Finally, I am once again interested in placing this class of languages in inclusion relationships with other existing classes.

Theorem 14. *TSL languages $\subset$ TESL.*

Proof. This result follows immediately from the definition of TSL and TESL, since any TSL language L can be described by a TESL grammar such that $\mathscr{T} = \{T\}$ and whose only erasing function is E_T. □

It should be clearer at this point how TESL is extending SS-TSL, since both classes rely on projection functions that are ISL$_k$, with the addition that TESL have the option to project tiers from tiers. The following results just formalizes this intuition: any SS-TSL language can be captured by a TESL grammar with $\mathscr{T} = \{T \subseteq \Sigma\}$ and $E_T : \Sigma \to T$.

Theorem 15. *The class of SS-TSL $\subset$ TESL.*

Interestingly, TESL languages for which the locality of the erasing function is $m = 1$ are a subclass of MTSL languages.

Theorem 16. *If $m = 1$, TESL $\subset$ MTSL.*

Proof. Consider any language L, such that Σ is its string alphabet and $\mathscr{T}$ is an arbitrary set of $n \in \mathbb{N}$ distinct tiers. If $L \in$ TESL, for every tier $T_i \in \mathscr{T}$ it is true that $T_i \subseteq \Sigma$. Then, by Definition 7, if $\forall T_i \in \mathscr{T}, E_{T_i} \in$ ISL$_1$ it follows that $L \in n$-MTSL. □

However, if we allow $m > 1$, the previous result does not hold.

Lemma 17. *If $m > 1$, the class of TESL $\nsubseteq$ MTSL.*

Proof. This follows directly from the fact that MTSL and SS-TSL are incomparable, as show in Lemma 7, and from SS-TSL $\subset$ TESL. □

Lemma 18. *The class of MTSL $\nsubseteq$ TESL.*

Proof. Take any MTSL language L such that $\mathscr{T} = \{T_1, T_2\}$ and $T_2 \subseteq \Sigma$, $T_1 \subseteq \Sigma$ but $T_1 \nsubseteq T_2$ and $T_2 \nsubseteq T_1$. Then L is not $TESL$ by definition, independently of any structural information the erasing function is allowed to consider. □

Theorem 19. *TESL is incomparable to MTSL.*

Proof. This results follows immediately from Lemma 17 and Lemma 18. □

We could now expect that, if structural projection is added to MTSL (as proposed in Section 5.3), than TESL (with $m > 1$) would be again in a proper subsumption relationship it. Almost surprisingly though, the following Theorem shows that this is not the case.

Theorem 20. *The class of TESL is incomparable to SS-MTSL.*

Proof. That SS-MTSL are not TESL follows once again from Lemma 18. For the other direction, consider a language $L \in$ TESL, with $\Sigma = \{a,b,c,d\}$ and the tier-grammars such that T_1 projects all as, bs, and cs and nothing is filtered out, whereas T_2 projects every a or b that follows a c, and requires the tier to be $(ab)^+$:

$$T_1 = \{a,b,c\}, S_1 = \{\}$$
$$T_2 = \{a,b\}, S_1 = \{^*\rtimes b, ^*bb, ^*aa, ^*a\ltimes\}$$

The erasing functions for L_{ab} are ISL_2 mappings defined as:

$$E_{T_1}(\sigma_i) = \begin{cases} \sigma_i & \text{if } \sigma_i \in T_1 \\ \lambda & \text{otherwise} \end{cases}$$

$$E_{T_2}(\sigma_i) = \begin{cases} \sigma_i & \text{if } \sigma_i \in T_2 \wedge c \prec^+ \sigma_i \\ \lambda & \text{otherwise} \end{cases}$$

For example $cacdb, cadcb \in L$ but $cadb, cadcbca, cacdcb \notin L$.

To an alert reader should be evident that the purpose of projecting T_1 is to create new structural relationships (a,b adjacency to c) that the projection function E_{T_2} can then exploit when creating T_2. This allows the grammar to rule in strings like cac^+d^+b, while ruling out cad^+b. Since the configurations T_2 needs to be sensible to in order to distinguish these two strings are not available in the original input, there in no number n of tiers such that $L \in \text{SS-MTSL}_n$. □

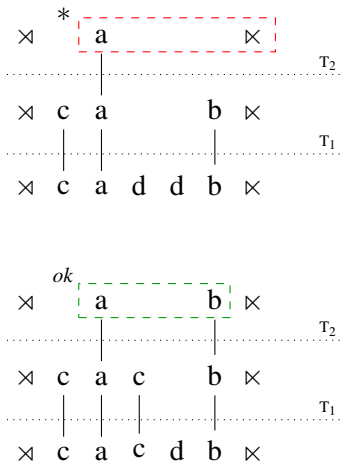

Figure 5: Example strings from L in the proof for Lemma 17. Via tier T_1, the grammar establishes direct relationships between a,bs and cs that are not available in the input string. It then makes sure that the language on T_2 is $(ab)^+$. The language is TESL but not MTSL.

The intuition behind most of the proofs above is that TESL introduces structural relationships that are unavailable in the input string. Since every erasing function of any SS-MTSL grammar can only use as input the original string, no combination of n tiers and m-local projections can discriminate between the strings in Figure 5.a and Figure 5.b. Vice-versa, MTSL allows for the projection of tiers with disjunct alphabets, while TESL requires all its tiers to be in a subset relationship with each other.

Now that the properties of TESL are clearer, the following results show where the class stands with respect to the rest of the hierarchy.

Lemma 21. *TESL is incomparable to LTT and PT.*

Proof. That TESL $\nsubseteq$ LTT, PT follows from TSL $\subset$ TESL. For LTT, PT $\nsubseteq$ TESL we can once again use the examples in Theorem 8. □

Finally, that TESL are Star-Free simply follows from their first-order definability.

Lemma 22. *The class of TESL $\subset$ SF.*

6.3 Combining Multiple Tiers with Tier Embedding

We can now get the intersection closure of TESL (MTESL) in the same way we did for SS-TSL, by minimally extending this class in order to allow for multiple parallel tier-projections.

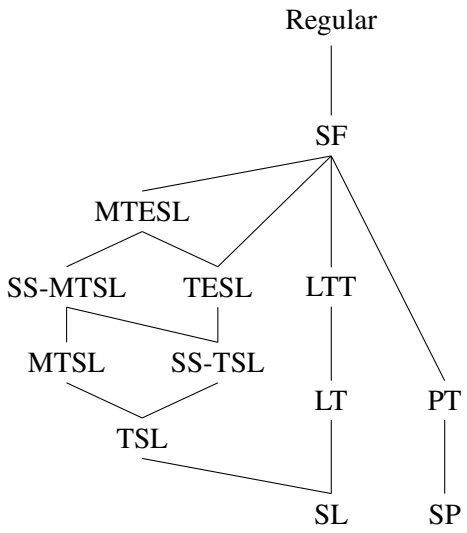

Figure 6: Proper inclusion relationships among sub-regular language classes. This paper establishes the SS-TSL, SS-MTSL, TESL, and MTESL classes.

Definition 11. (Multiple-Tier Embedding Strictly Local) A language L is Multiple-Tier Embedding Strictly Local (MTESL) iff there exists a finite set of sets $\mathscr{T}_{order} = \{\mathscr{T}_i : \{T_i : T_i \subseteq \Sigma, 0 \leq i \leq n \in \mathbb{N}\}\}$ and L is Tier-Embedding Strictly k-Local for every set $\mathscr{T}_i \in \mathscr{T}_{order}$.

Essentially, by combining tier-embedding and multiple projections, we loosen the order on the tier-projections from TESL's total order/string to a tree, giving rise to a class of languages subsuming both SS-MTSL and TESL.

Lemma 23. *SS-MTSL, MTSL* $\subset$ *MTESL.*

This class is FO definable from TESL, in the same way MTSL and SS-MTSL were characterized starting from their respective single-tier classes, just by allowing for the intersection of FO formulae used to define distinct TESL grammars:

Lemma 24. *MTESL* $\subset$ *SF.*

Moreover, the proof that MTESL $\not\subseteq$ LTT, PT is the same as in Theorem 13.

Lemma 25. *TESL is incomparable to LTT, SP.*

The final hierarchy of sub-regular classes as discussed in this paper is shown in Figure 6.

7 Discussion

The TSL class comes with particularly tight constraints on the erasing function. The inspiration behind this paper was to try and explore the effects of relaxing such constraints, thus allowing for more general definitions of the projection mechanism. As shown in Figure 6, each of the classes introduced above generalizes TSL in a different way, while covering the problematic patters discussed in Section 2.

From a linguistic point of view, one problem with the additional structure-sensibility of SS-TSL grammars is that it can lead to the generation of patterns that are unattested among natural phonotactics. For instance, the sibilant harmony in SAMALA discussed in Section 2 can easily be converted to a case of first/last harmony – thus leading to problematic over-generation – by allowing the grammar to project only initial/final sibilants. In Section 6.1 I also discussed how, if we don't allow structural projection, tier-embedding languages are just a special case of multiple tier projection. In a similar way, it would be interesting to see whether it is possible to carve a subclass of structure-sensitive languages that covers the desired SS-TSL patterns while avoiding the unnatural ones. Another useful direction could be to compare the classes introduced in this paper against the constraint-based approach to TSL proposed by McMullin (2016) to account for phonotactic patterns in which local and non-local dependencies interact. In fact, the patterns in (McMullin, 2016) can all be captured by SS-TSL grammars – and the generative complexity of these classes in better defined in formal terms than the one of *ranked-TSL* grammars.

With respect to the other classes discussed in this paper, I already mentioned that all the patterns discussed by De Santo (2017) to motivate a formal study of a conjunction of TSL grammars, always project one tier whose alphabet is the superset of all the others. Clearly, TESL with structure-sensitivity reduced to 1 can easily account for these processes, by enforcing a specific set/subset relationships over the multiple-tier alphabet. Interestingly, MTSL and TESL make different predictions with respect to the kind of grammars we can expect to find among natural languages. TESL also predicts languages that establish a long-distance agreement over a set of segments (e.g. sibilant voicing harmony blocked by nasals) and then another kind of agreement among a subset of them (e.g. only voiced sibilants agree in anteriority).

Evidently, to really assess the linguistic useful-

ness of what we can now call the *TSL neighborhood*, there is need for more careful typological exploration. In particular, moving from phonotactic to linguistics domains in which attempts to a sub-regular analysis are just at the beginning (i.e. morphotactics (Aksënova et al., 2016), syntax, or even semantics (Graf, 2017)) should improve our empirical understanding of TESL, and of the intersection-closure classes (SS-MTSL, MTESL). Generally though, the fact that a few minor modification to TSL allow us to effectively combine the full set of phonotactic processes for a language while keeping the generative power in check, supports future studies of this region.

Finally, we could gain a more comprehensive understanding of the *refined* sub-regular hierarchy as developed in recent years by comparing this paper's TSL extensions to the class of Interval-based Strictly Piecewise languages introduced by Graf (2016) as an extension of TSL, SL and SP.

7.1 Closure Properties

The current characterization of the classes presented in this paper still lacks a discussion of their closure properties. Clearly, the fact that SS-MTSL and MTESL are closed under intersection follows from their definition. Moreover, these classes are not closed under relabeling. Simply consider the SL (thus SS-TSL, TESL, SS-MTSL, and MTESL) language $L_{ab} = (ab)^+$ and the relabeling $r(L) := \{r(w) | w \in L\}$ s.t. $r : \Sigma \to \{a\}$: the resulting language is $r(L_{ab}) = (aa)^+$, which is not even SF.

In general, it can be shown that SS-TSL and TESL are not closed under intersection, union, and relative complement by exploiting the counter-examples used in De Santo (2017)'s non-closure proofs for TSL and MTSL languages. Furthermore, the MTSL examples can be used to show that SS-MTSL and MTESL are also not closed under union and complement. However, explicitly adapting these results is notationally convoluted, since we would need to evaluate all the additional cases of tier-combinations allowed by multiple projections interacting with structural-sensitivity. Thus, the actual proof is not particularly informative and it is omitted here as, also due to space constraints, I prefer to outline the essential underlying intuitions instead.

Too see how this works, let us introduce some additional notation, and briefly consider the case of non-closure of SS-TSL and TESL under intersection. Consider a language L over alphabet Σ. Given $\{a,b\} \subseteq \Sigma$, $\{a,b\}(L)$ denotes $\{E_{\{a,b\}}(s) \mid s \in L\}$. Then, $T(L) := \{E_T(s) \mid s \in L\}$, and we let the *down-projection* of T be $\downarrow T(L) := \{s \in \Sigma^* \mid E_T(s) \in T(L)\}$, with $T \subseteq \Sigma$. Note that $\downarrow T(L)$ is the language resulting from the cascade $ISL \to id(SL) \to \mathscr{D}$ described in Section 4.1, and may be a proper superset of L. Now, take the largest Σ-languages L_1 and L_2 with $\{a,b\}(L_1) := a^+b$ and $\{c\}(L_2) := c$. That is, $L_1 := \{a,c\}^*a^+\{a,c\}^*bc^*$ and $L_2 := \{a,b\}^*c\{a,b\}^*$, wherefore $L_1 \cap L_2 = ca^+b \cup a^+cb \cup a^+bc$. It holds for all $T \neq \{a,b,c\}$ that $L_1 \cap L_2 \neq \downarrow T(L_1 \cap L_2)$. But if $T = \{a,b,c\}$ then $T(L_1 \cap L_2) = L_1 \cap L_2$. Since $L_1 \cap L_2$ is not SL, $T(L_1 \cap L_2)$ is not either, and consequently $L_1 \cap L_2$ is not TSL by definition. The problem here is the union of two TSL languages results is a tier-language that is not strictly local. It should be possible to see that this cannot be fixed by moving to an SS-TSL grammar, since increasinf the locality of the ISL projection function will not simplify the tier-language necessary to get $L_1 \cap L_2$. Moreover, the resulting language cannot be obtained by a decomposition of the tier-language into simple SL languages over embedded tiers, thus $L_1 \cap L_2$ is not TESL either. Although this example is not an exhaustive formal proof, it illustrates the general reasoning behind the results in (De Santo, 2017), which can be extended to all the new classes.

7.2 Implications for Syntax

Since the current consensus is that the requirements of syntax greatly exceed the computational power of regular languages[2], it is probably not particularly surprising that most of the work connecting notions in the sub-regular hierarchy to natural language processes has been done in phonology.

Recently, Graf (2014) proposed that syntactic dependencies can be seen as are finite-state over MG derivation trees (Stabler, 1997), in the same way phonological dependencies are finite-state over strings (see also (Graf and Heinz, 2015)). Following up on this proposal, Graf and Heinz (2016) explore links between syntax and the sub-regular hierarchy, showing that *Merge* and *Move* are in fact TSL operations – given some assumption on the properties of the MG used to encode the language. In this account, a grammar projects

[2]There is still some disagreement about the maximum complexity of syntactic patterns. See (Kobele, 2006), (Shieber, 1985) for discussions on this issue.

a node on a tree-tier based on that node label (e.g. project a tier of nodes labeled *TP* or *DP*). However, in standard MGs *merge* nodes are not explicitly labeled, since the grammar is always able to reconstruct the *type* of a node via the features that lead to that constituent's formation. Thus, to really be able to project tiers on a MG tree, a grammar needs to reconstruct the *type* of a node from the properties encoded in the features of the head of the constituent rooted in that node. It should be easy to see that, while this would be tricky to accomplish with a purely TSL grammar, the projection of a node based on the properties of its neighbors is exactly what SS-TSL allows.

7.3 Learnability Considerations

From a linguistic perspective, the phonotactic learning problem in concerned with the way a speaker learns to distinguish between ill-formed and well-formed strings given a finite set of words that are in the language. As observed by Heinz and Riggle (2011) (cf. (Albright and Hayes, 2011)), by focusing on formal classes of languages, a theory of learning will be able to determine characteristics of the inputs data that fundamentally underlie learnability/acquisition.

It is known that TSL languages are learnable in the limit from positive data. Given a fixed alphabet and a fixed k, the number of possible tiers and permissible tier k-factors is finite, and thus learnable, since any finite class of languages is identifiable in the limit via an enumeration method (Gold, 1967). Not surprisingly, this result extends to our new classes, which are all finite given upper bounds on their fundamental parameters (number of tiers, locality of constraints, locality of projection). In fact, the true constraints to learnability come just from the locality of the ISL projection function, and from the locality of tier-grams, since the number of useful (i.e. distinct and with separate grammars) tiers is always finite, and bounded by the size of the alphabet.

Theorem 26. *The classes of SS-TSL, TESL languages, and their intersection closures are learnable in the limit from positive data.*

Besides general learnability, proving that subregular classes are learnable with computationally efficient methods has important consequences for the cognitive relevance of the TSL neighborhood. Although learners based on the Gold paradigm are reportedly inefficient, a series of learning algorithms grounded in grammatical inference and formal language theory have been proposed in the past. For example, Jardine and Heinz (2016) present an algorithm for learning 2-TSL languages, proving that constraints over phonological tiers can be learned even when the tier alphabet is not known *a priori*. This learner seems to be easily adaptable to SS-TSL, by inducing a tier of segments based on their k-local properties. Chandlee et al. (2014) also presents an algorithm crucially based on the properties of Input Strictly Local functions, which offers promising perspectives in efficiently learning the ISL_k erasing function.

As for multiple-tier grammars, at the present stage neither Jardine & Heinz or Chandlee's algorithms seems to work with multiple tiers. However, in the future it would be interesting to see whether the intuitions behind their learning strategies can be extended. Moreover, McMullin and Allen (2015) propose a learner for conjoined TSL languages that exploits search over lattices. While their implementation still lacks generality, it should be possible to adapt it to work for SS-MTSL and MTESL too. Among alternative approaches worth exploring, (Goldsmith and Riggle, 2012) propose a tier-based learner for harmony patterns relying on mutual information.

8 Conclusions

A growing body of literature is exploring TSL languages as a good computational hypothesis for the complexity of phonotactic patterns. However, the TSL class comes with particularly tight constraints on the projection function. Here I relax some of these constraints, allowing for more general definitions of tier-projection. The resulting new classes naturally extend TSL, and easily captures patterns in which local and non-local dependencies interact that have been a problem for TSL accounts. This also suggests that understanding the way constraints on the projection mechanism restrain TSL generative power could help identify fundamental underlying properties of phonotactic dependencies, and opens the way to significant future work both in formal language theory, and in sub-regular approaches to linguistic phenomena.

References

Alëna Aksënova, Thomas Graf, and Sedigheh Moradi. 2016. Morphotactics as tier-based strictly local dependencies. In *Proceedings of SIGMorPhon 2016*. To appear.

Adam Albright and Bruce Hayes. 2011. Learning and learnability in phonology.

R.B. Applegate. 1972. *Ineseno Chumash grammar*. Ph.D. thesis, University of California,Berkeley.

Hyunah Baek. 2016. Computational representation of unbounded stress patterns: tiers with structural features. Ms., Stony Brook University.

Jane Chandlee, Remi Eyraud, and Jeffrey Heinz. 2014. Learning strictly local subsequential functions. *Transactions of the Association for Computational Linguistics*, 2:491–503.

Jane Chandlee. 2014. *Strictly Local Phonological Processes*. Ph.D. thesis, University of Delaware.

Aniello De Santo. 2017. Multiple-tier strictly local languages. Ms., Stony Brook University.

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

John A Goldsmith and Indiana University (Bloomington). Linguistics Club. 1976. Autosegmental phonology.

John Goldsmith and Jason Riggle. 2012. Information theoretic approaches to phonological structure: the case of finnish vowel harmony. *Natural Language and Linguistic Theory*, 30(3):859–896.

Thomas Graf and Jeffrey Heinz. 2015. Commonality in disparity: The computational view of syntax and phonology. Slides of a talk given at GLOW 2015, April 18, Paris, France.

Thomas Graf and Jeffrey Heinz. 2016. Tier-based strict locality in phonology and syntax. Ms., Stony Brook University and University of Delaware.

Thomas Graf. 2014. Beyond the apparent: Cognitive parallels between syntax and phonology. In Carson T. Schütze and Linnaea Stockall, editors, *Connectedness: Papers by and for Sarah van Wagenen*, volume 18 of *UCLA Working Papers in Linguistics*, pages 161–174.

Thomas Graf. 2016. The power of locality domains in phonology. Ms., Stony Brook University.

Thomas Graf. 2017. The subregular complexity of monomorphemic quantifiers. Ms., Stony Brook University.

R. J. Hayward. 1990. Notes on the Aari language. In R. J. Hayward, editor, *Omotic language studies*, pages 425–493. School of Oriental and African Studies, University of London, London, UK.

Jeffrey Heinz and Jason Riggle. 2011. Learnability. In Marc van Oostendorp, Colin Ewen, Beth Hume, and Keren Rice, editors, *Blackwell Companion to Phonology*. Wiley-Blackwell.

Jeffrey Heinz, Chetan Rawal, and Herbert Tanner. 2011. Tier-based strictly local constraints for phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 58–64, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeffrey Heinz. 2014. Culminativity times harmony equals unbounded stress. In Harry van der Hulst, editor, *Word Stress: Theoretical and Typological Issues*, chapter 8. Cambridge University Press, Cambridge, UK.

Jeffrey Heinz. 2015. The computational nature of phonological generalizations. Ms., University of Delaware.

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

Gregory M. Kobele. 2006. *Generating Copies: An Investigation into Structural Identity in Language and Grammar*. Ph.D. thesis, UCLA.

Kevin J. McMullin and Blake H. Allen. 2015. Phonotactic learning and the conjunction of tier-based strictly local languages. In *aper presented at LSA 89th Annual Meeting*, page 137. To appear.

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

Robert McNaughton and Seymour Papert. 1971. *Counter-Free Automata*. MIT Press, Cambridge.

J. Oncina and P. García, 1991. *Inductive learning of subsequential functions*. Univ. Politecnica de Valencia, Tech. Rep., DSIC II-34.

James Rogers and Geoffrey K. Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*, 20(3):329–342.

James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel, 2010. *On Languages Piecewise Testable in the Strict Sense*, chapter Lecture Notes in Computer Science, pages 255–265. Springer.

James Rogers, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel, 2013. *Cognitive and Sub-regular Complexity*, chapter Formal Grammar, pages 90–108. Springer.

Stuart M. Shieber. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–343.

Edward P. Stabler. 1997. Derivational minimalism. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95. Springer, Berlin.

Rachel Walker. 2000. Yaka nasal harmony: Spreading or segmental correspondence? *Annual Meeting of the Berkeley Linguistics Society*, 26(1):321–332.