Structure Sensitive Tier Projection: Applications and Formal Properties

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. Here, grounded in the fact that there are phonotactic processes that cannot be described with a simple tier-based account, we explore a new class of subregular languages obtained by relaxing a particular constraint on the tier-projection mechanism.

1 Introduction

The complexity of linguistic dependencies has been a topic of great interest in the last decade. While many dependencies in phonology are local, and thus can be described with n-gram models, unbounded dependencies have proven problematic. But in fact a minor extension of n-gram models — tier-based strictly local (TSL) grammars — is sufficient to capture a large number of these processes (Heinz et al., 2011; McMullin, 2016; McMullin and Hansson, 2016).

TSL grammars build on the idea of phonological tiers, pioneered in representational theories of phonology for the study of tone systems, among other things (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, with n-gram constraints acting only over that subset. This paper explores possible extensions of the TSL class, that are grounded in the need to describe problematic phonotactic patters. By modifying TSL grammars projection mechanism, we show how it is possible to capture outlier processes, and study the increase in generative power that arises even from these minor changes.

The paper is structured as follows. Section 2 presents phonotactic processes that are problematic for the TSL account. Section 3 introduces mathematical notation that is essential for studying subregular languages. This particularly weak subregion of the Chomsky hierarchy is discussed in detail in Section 4, with emphasis on TSL and its intersection closure. The central result of the paper is in Section 5, which presents new extensions of TSL, and relates them to the rest of the subregular hierarchy. Section 6 discusses the implications of these results for linguistics and learnability.

2 Linguistic Motivation

The primary motivation for studying the properties of TSL extensions comes from their 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 substrings 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 cannot be captured with local constraints, and have been characterized instead as *tier-based strictly local*. A full introduction to the formal properties of TSL is given in Section 4.2.

Intuitively, a TSL dependency is a dependency that is non-local in the input string, but local over a special structure called a *tier*. A tier is defined as the projection of a subset of the segments of the input string, and the grammar constraints are char-

acterized 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 from the input a tier T that only contains sibilants, and ban contiguous *ʒs and *sʒ on T (c.f. (Hayward, 1990)).

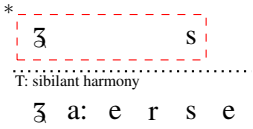

Figure 1: Example from Aari showing long-distance sibilant harmony as a local dependency over a sibilant tier.

However, several processes have been reported that cannot be described by the way TSL currently combines its projection mechanism — used to filter irrelevant segments — and strictly local dependencies. In particular, sometimes it is necessary to check some structural properties of a segment in the input string *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 that no TSL grammar is able to generate this pattern, which should allow for well-formed strings like **Ĺ**LLLLH while also ruling out ill-formed strings like *Ĺ**L**LLLHH.

In order to discriminate among those strings, the essential property to be considered is a syllable *non-finality*, which is a structural property and cannot be deduced from the tier — a final heavy syllable on a tier may be either non-final or final in the string the tier was projected from. The grammar would need to check the position of a syllable before projecting it on a tier. However, this is not something TSL is currently allowed to do.

Similarly, McMullin (2016) discusses 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, cf. (a)) 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] (cf. (b) vs (c)), unless there is an

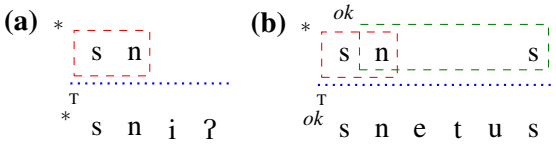

Figure 2: Example from Samala: (a) is ill-formed because of adjacent *sn; (b) is well-formed since sn is followed by another s later in the string, but it is still ruled out by the grammar.

[s] following in the string, in which case it surfaces as [sn] (cf. (d)):

a) *ok*kʃuʃojin *"I darken it"*

b) *ok*ʃniʔ *"his neck"*

c) *sniʔ *"his neck"*

d) *ok*snetus *"he does it to him"*

Figure 2 exemplifies why it is not possible to capture this overall pattern with a single TSL grammar. Since [sn] is sometimes observed in a string-adjacent context (as in (d)), it must be permitted as a 2-gram on a tier (even though it is only allowed when a segment such as [s] follows them later in the string). But then, a TSL grammar would have no means of banning *sn when there is no subsequent [s] in the string (as in (b) vs. (c)). Vice-versa — as shown is Figure 2 — if we ban *sn on T, then the grammar will not be able to allow it when another [s] follows on the tier.

A careful reader might point out that the difference between Figure 2.a and Figure 2.b can be resolved by extending the tier-grammar to consider 3-grams. However, in order to ban *sn, every occurrence of [n] in the input string must projected on the tier. Since the number of [n] segments between two sibilants is potentially unbounded, no TSL grammar can generally account for this pattern, independently of the dimension of the tier k-grams.

Another example of such processes can be found in Yaka, where [$+nasal$] segments start a harmony domain only if they are not immediately followed by voiced oral stops (Walker, 2000), or among culminativity patterns (cf. Heinz (2014)).

All the processes above are beyond the reach of TSL because projection of a segment is no longer determined solely by that segment's label, but also by other surrounding segments.

In Section 5, we demonstrate that a very minor and natural change in the definition of TSL grammars is sufficient to allow for this kind of structure-sensitive projection.

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. The concatenation of two languages $L_1L_2 = \{uv : u \in L_1 and v \in L_2\}$. For all strings w and non-empty strings u, $|w|$ denotes the length of the string, $|w|_u$ denotes the number of occurrences of u in w, and λ is the unique empty string. Left and right word boundaries are marked by $\rtimes, \ltimes \notin \Sigma$ respectively.

We define generalized regular expressions (GREs) recursively. GREs include λ, $\emptyset$, and every symbol $\sigma \in \Sigma$. If R and S are GREs, then RS, $R+S$, $R \times S$, and R^* are also GREs. The language of a GRE is defined as follows. $L(\emptyset) = \emptyset$. For all $\sigma \in \Sigma \cup \{\lambda\}$, $L(\sigma) = \{\sigma\}$. If R and S are regular expressions, then $L(RS) = L(R)L(S)$, $L(R+S) = L(R) \cup L(S)$, $L(R \times S) = L(R) \cap L(S)$. Also, $L(R^*) = L(R)^*$. A language is *regular* iff there is a GRE defining it.

A string u is a *k-factor* of a string w iff $\exists x, y \in \Sigma^*$ such that $w = xuy$ and $|u| = k$. The function F_k maps words to the set of k-factors within them.

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

For example, $F_2(aab) = \{aa, ab\}$.

The domain F_k is generalized to languages $L \subseteq \Sigma^*$ in the usual way: $F_k(L) = \bigcup_{w \in L} F_k(w)$. We also consider the function which counts k- factors up to some threshold t.

$$F_{k,t}(w) := \{(u,n) : u \text{ is a k-factor of } w \text{ and } n = |w|_u \text{ iff } |w|_u < t \text{ else } n = t\}$$

For example $F_{2,3}(aaaaab) = \{(aa,3),(ab,1)\}$.

The set of prefixes of w is $Pref(w) = \{p \in \Sigma^* | w = ps \text{ for some } s \in \Sigma^*\}$ and the set of suffixes is $Suff(w) = \{s \in \Sigma^* | w = ps \text{ for some } s \in \Sigma^*\}$. For all $w \in \Sigma^*$ and $n \in \mathbb{N}$, $Suff^n(w)$ is the single suffix of w of length n if $|w| \geq n$, and $Suff^n(w) = w$ otherwise. 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:

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

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

Following Oncina et al. (1993), a subsequential finite state transducer (SFST) is a 6-tuple $(Q, q_0, \Sigma, \Gamma, \delta, \sigma)$, where Q is a finite set of states, Σ and Γ are finite alphabets, $q_0 \in Q$ os the initial state, $\delta \subseteq Q \times \Sigma \times \Gamma^* \times Q$ is a set of edges, and $\sigma : Q \to \Gamma^*$ is the final output function that maps states to strings that are appended to the output if the input ends in that state. δ recursively defines a mapping $\delta^* : (q, \lambda, \lambda, q) \in \delta^*$; if $(q, u, v, q') \in \delta^*$ and $(q', \sigma, w, q'') \in \delta$ then $(q, u\sigma, vw, q') \in \delta^*$.

In order to simplify some proofs, we rely on first-order logic characterizations of certain string languages and string-to-string mappings. As usual, 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 $x \lhd y \leftrightarrow x \prec y \wedge \neg\exists z[x \prec z \wedge z \prec y]$.

We add a dedicated predicate for each label $\sigma \in \Sigma$ we wish to use: $\sigma(x)$ holds iff x is labelled σ, where x is a position in w. 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$(\sigma_1(\cdot), \ldots, \sigma_n(\cdot), \prec)$. If φ is a logical formula without any free variables, we use $L(\varphi) = \{w \in \Sigma^* | M_s \models \varphi\}$ as the stringset extension of φ .

4 The Subregular Hierarchy

As mentioned earlier, the TSL languages are part of the subregular hierarchy. Originally defined in McNaughton and Papert (1971) and subsequently extended in Rogers et al. (2010), the subregular hierarchy categorizes subclasses of the regular languages according to their complexity. The hierarchy has become very intricate in recent years, and the reader is referred to Rogers and Pullum (2011),

Rogers et al. (2013) for details. We only provide the basic definitions for a few relevant classes 4.1, and focus instead on TSL, as well as well as its intersection closure (MTSL, (De Santo, 2017)) since they are central to our results.

4.1 Basic Definitions

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

$$L = \{w \in \Sigma^* : F_k(\rtimes^{k-1} w \ltimes^{k-1}) \subseteq S\}$$

We also call S a strictly k-local grammar. A language L is strictly local iff it is strictly 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.

One note regarding left and right edge-markers: for S to be k-local, it needs to contain only factors of length k. Thus, strings are augmented with enough left and right edge-markers to ensure that this requirement is satisfied. However, it is often convenient to shorten the k-factors in the definition of strictly k-local grammars, and write down only one instance of each edge-marker, with the implicit understanding that it must be augmented to the correct amount. We adopt this simpler notation throughout the paper, unless required to make a definition clearer.

Strictly k-local languages are characterized by k-local suffix substitution closure. This makes it much easier to show that certain languages are not strictly local, which greatly simplifies some of the later proofs.

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 $u_1 \cdot x \cdot v_2 \in L$.

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

Strictly local languages are at the very bottom of the hierarchy. Closer to the top we find the class of locally threshold testable languages.

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 languages are those whose strings contain a restricted number of occurrences of any k-factor in a string. 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.

Properly included in LTT, *locally testable* (LT) languages are locally threshold testable with $t = 1$.

Higher in the hierarchy than TSL are also the piecewise testable languages discussed by Rogers and Pullum (2011).

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.

Piecewise languages are sensible to relationships between segments based on *precedence* (over arbitrary distances) rather than *adjacency* (immediate precedence).

The last subregular class relevant to our discussion is the class of *star-free* languages, a proper subset of regular languages closed under Boolean operations. Multiple characterizations are know for *star-free* languages. Here we prefer the one in terms of first-order logic (McNaughton and Papert, 1971), which is more helpful to some of the proofs in this paper.

Definition 5. (Star-Free) Star-free (SF) languages are those that can be described by first order logic with precedence.

In other words, SF languages are those languages that can be obtained from a set of unary predicates by using the operations of union, complement, and concatenation, and which contain no instances of the Kleene star $(^*)$.

4.2 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 symbols that are not in T.

Given some $\sigma \in \Sigma$, the erasing function $E_T : \Sigma \to \Sigma \cup \lambda$ maps σ to itself if $\sigma \in T$ and to λ otherwise.

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

We extend E_T from symbols to strings in the usual pointwise fashion.

Definition 6. (TSL Language) A language L is strictly k-local on tier T iff there exists a tier $T \subseteq \Sigma$ and a finite set $S \subseteq F_k(\rtimes^{k-1} T^* \ltimes^{k-1})$ such that

$$L = \{w \in \Sigma^* : F_k(\rtimes^{k-1} E_T(w) \ltimes^{k-1}) \subseteq S\}$$

We also call S the set of permissible k-factors on tier T.

As can be gleaned from Definition 6 states that a language L is TSL 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 7. (Input Strictly Local Function) A function f is input strictly k-local (ISL-k) iff there is some k such that f can be described with a SFST for which:

1. $Q = \Sigma^{\leq k-1}$ and $q_0 = \lambda$
2. $(\forall q \in Q, \forall a \in \Sigma, \forall u \in \Gamma^*)$
   $[(q,a,u,q') \in \delta \Rightarrow q' = Suff^{k-1}(qa)]$

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 $k = 1$.

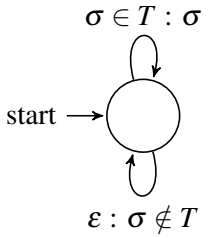

Figure 3: Transducer $\mathscr{D}$

A TSL grammar G can be decomposed into a cascade of three transducers. The first one is the ISL-1 transducer for the projection function E_T. The output of that transducer is filtered by the strictly local grammar S of G. The filtration step can be implemented by composing E_T with the identity function over $L(S)$. Finally, a third transducer computes the inverse of E_T, mapping every string s to some s' such that $E_T(s') = s$. The transducer $\mathscr{D}$ computing the inverse of E_T is shown in Figure 3.

The transducer perspective gives us interesting insights into the relation between tier-projection and strict locality, and further simplifies many proofs in the paper.

4.3 Multi-Tier Strictly Local Languages

As mentioned in Section 2, while TSL captures many unbounded dependencies in phonotactics, the conjunction of distinct TSL languages has been used in the literature to characterize more complex patterns.

However, De Santo (2017) shows that TSL languages are not closed under intersection. Then, to account for processes requiring more than one grammar, he introduces *multi-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 viewed as TSL languages projecting over multiple tiers and enforcing a — potentially different — set of constraints for each tier. Then, a string is in the language iff each substring is well-formed on every projected tier. Not too surprisingly, the MTSL class is exactly the intersection closure of TSL.

Definition 8. A multi-tier strictly k-local ((k,n)-MTSL) language L is the intersection of n distinct k-TSL languages $L_1, \ldots, L_n$, with $k, n \in \mathbb{N}$. For succinctness, we write that a language is k-MTSL$_n$, where k, n can be omitted when not relevant to exposition.

The following two theorems sum up most of the properties of MTSL languages that are relevant to the current work. The reader is referred to (De Santo, 2017) for proofs of these results.

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

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

4.4 Logical Definability and Subregularity

The variants of TSL discussed in this paper are properly contained in the class of star-free languages. These containment relations are obvious if one adopts a perspective grounded in first-order logic. Recall from Definition 5 that a language is star-free iff it can be defined in first-order

logic (FO). The TSL variants can easily be defined as highly restricted fragments of first-order logic, wherefore they are all star-free.

The logical characterization of SL as conjunctions of negative literals provides the general template which all TSL variants build on. A string satisfies the property p_S iff it contains the k-gram p of grammar S. For example, the strings that are well-formed with respect to $S = \{^*\rtimes b, ^*bb, ^*aa, ^*a\ltimes\}$ are those for which the following holds:

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

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

Therefore, a strictly k-local grammar S can be written as a formula

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

where

$$p := \exists x_1 \ldots x_k \left[\bigwedge_{1 \leq i < k} x_i \neq x_{i+1} \wedge x_i \lhd x_{i+1} \right]$$

It is easy to extend this characterization to TSL languages. We can define *tier-precedence* with respect to tier $T \subseteq \Sigma$ as the binary predicate:

$$x \lhd_T y \leftrightarrow T(x) \wedge T(y) \wedge x \prec y \wedge \neg \exists z [T(z) \wedge x \prec z \wedge z \prec y]$$

and

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

Thus, every TSL language is defined by an FO formula like ϕ above, except that immediate precedence is replaced by tier-precedence in the definition of each p. Since every MTSL language is the intersection of finitely many TSL languages, the formulae for MTSL languages are conjunctions of TSL formulae.

5 Structure-Sensitive TSL Languages

With all the preliminaries finally in place, we return to the initial question of how the projection mechanism of TSL can be extended to handle the problematic phenomena discussed in Section 2.

The projection mechanism of TSL languages is based on each segment's 1-local properties (i.e. its label and nothing else). This section presents a new class of grammars: structure-sensitive TSL (SS-TSL). The projection mechanism of SS-TSL grammars is generalized from strictly 1-local to strictly k-local, which makes it possible to project segments based on the structural context they appear in. In terms of the transducer perspective of TSL given at the end of Section 4.2, SS-TSL grammars are transducer cascades where the first transducer is expanded from ISL-1 to ISL-k.

5.1 Definition

We define a *context* $c \in C$ as triple (g, σ, g'), where $\sigma \in \Sigma$, g is an n-gram and g' an n'-gram, such that $0 \leq |g| + |g'| < m$.

Given $w \in \Sigma^*$, $T \subseteq \Sigma$ and a set of contexts C_T, the erasing function $E_T : \Sigma \to \Sigma \cup \lambda$ maps σ to itself iff $\exists (g, \sigma, g') \in C_T$ s.t. $g\sigma g' \in F_m(\rtimes^{m-1} w \ltimes^{m-1})$, and to λ otherwise. In other words, $E_T(\sigma)$ is an ISL-m function projecting a segment on a tier iff there is at least one matching context. A TSL erasing function with $m > 1$ can also be referred to as *structural projection*.

Then, the definition of SS-TSL languages mirrors the one for TSL.

Definition 9. (Structure-Sensitive TSL) A language L is structure-sensitive tier-based strictly local ((k,m)-SS-TSL) with *tier-sensitivity* m iff, given a set of contexts C_T and $T \subseteq \Sigma$, there exist an ISL-m erasing function E_T, and a finite set $S \subseteq F_k(\rtimes^{k-1} T^* \ltimes^{k-1})$ such that

$$L = \{w \in \Sigma^* : F_k(\rtimes^{k-1} E_T(w) \ltimes^{k-1}) \subseteq S\}$$

For succinctness, we write L is k-SS-TSL^m, where k and m can be omitted unless required from context.

As an example, assuming $\Sigma = \{a, b\}$, Figure 4 presents an ISL transducer with input and output alphabet $\{a, b\}$ that computes the erasing function E_T with structure sensitivity $m = 2$, projecting on the tier only a, bs in initial/final position.

In order to characterize SS-TSL in terms of FO, one has to modify the notion of tier-membership that is used to determine tier-precedence. For every segment, we need to re-define what it means to belong to a tier based on its m-local properties.

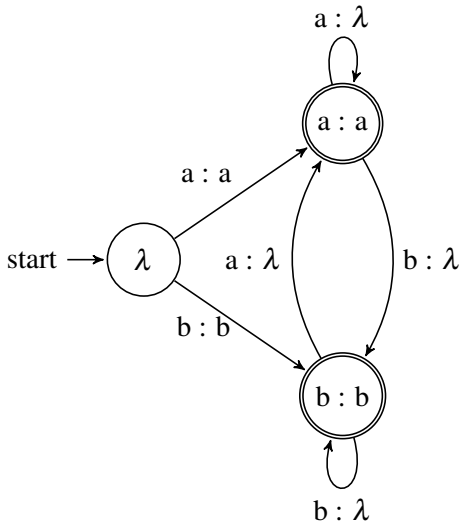

Figure 4: ISL_2 transducer for tier of first/last a, b

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

where γ is a FO predicate defining for each y_i a set of possible labels and precedence relationships to x, In other words, Γ is the set of contexts defined for tier T.

To give a practical example, suppose we want an SS-TSL^m language such that it projects a tier of initial/final as. Then:

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

For ease of exposition, we will refer to tier-precedence employing this notion of tier-membership with $\blacktriangleleft_T$. Then, a SS-TSL language is described by an FO formula just like 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, we now establish their position inside the subregular hierarchy. In particular, we show that SS-TSL generalizes TSL in a different ways than MTSL, while still remaining a proper subset of SF.

Theorem 4. *TSL $\subsetneq$ SS-TSL.*

Proof. The inclusion is a corollary of Proposition 1, which states that the erasing function of TSL languages is an ISL mapping with $m = 1$. For proper inclusion, consider the language $L = a\{a,b\}^*b \cup b\{a,b\}^*a$. L can be generated by an SS-TSL^2 grammar with $T = \{a : \rtimes a \vee a \ltimes, b : \rtimes b \vee b \ltimes\}$ and $S = \{^*aa, ^*bb\}$. We can show that L is not TSL. Consider $a\{a,b\}^*a \notin L$: to rule this string out, a TSL grammar would need to project every a on the tier and ban $^*aa$. However, this would also ban $a\{a,b\}^*b \in L$. Thus, L is SS-TSL^2 but not k-TSL for any k

□

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 once again consider the language $L = a\{a,b\}^*b \cup b\{a,b\}^*a$. As already stated in the proof of Theorem 4, L can be generated by an SS-TSL^2 grammar with the erasing function in Figure 4. Now, if L is MTSL_n, it is the intersection of n distinct TSL languages $L_1, \ldots, L_n$. Since $a\{a,b\}^*a \notin L$, there has to be at least one L_i projecting every a on the tier, and enforcing $^*aa$. But then, this language would also incorrectly rule out $a\{a,b\}^*b$. Thus, $L \notin \text{MTSL}_n$ for any number of intersecting TSL languages. □

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 $L = b^*a\{b,d\}^*cd^*$. This language is not closed under suffix substitution, thus it is not SL. But it can be generated by a MTSL_3 grammar with:

$$T_1 = \{a,c\}, S_{T_1} = \{^*\rtimes c, ^*a\ltimes, ^*\rtimes\ltimes\}$$

$$T_2 = \{b,c\}, S_{T_2} = \{^*cb\}$$

$$T_3 = \{a,d\}, S_{T_3} = \{^*da\}$$

Assume now that $L \in$ SS-TSL. To enforce the constraints of S_{T_1} and ban strings like $\{b,d\}^*, b^*a\{b,d\}^*, \{b,d\}^*cd^*$, the grammar needs to project a and c on a tier T. Since $^*cb$ sequences are also out, b is projected on the tier. Finally, to avoid $^*da$ sequences, d needs to be on the tier too. But now $T = \Sigma$, and S_T is reduced to a strictly local grammar over the input string. Since $\{a,b,c,d\}$ are all on the tier, S_T will not be able to

rule out strings like $\{b,d\}^+$ which clearly are not in L, while allowing the well-formed $a\{b,d\}^*c$. Hence, $L \notin$ 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. Corollary of Lemma 5 and Lemma 6. Moreover, the two classes are obviously not disjoint, since TSL languages are both MTSL and SS-TSL. □

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

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

To see why LT is not a subclass of SS-TSL, consider $\Sigma = \{a,b,c\}$ and a sentential logic formula $\varphi := aa \rightarrow bb$ defining a language $L = \{w \in \Sigma^* | w \vDash \varphi\}$. 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.

We can show that $L \notin$ SS-TSL^m, independently of the locality of the structural projection. Since strings like aa^+ are ill-formed, if $L \in$ SS-TSL, there is a tier T containing every a, and the grammar should ban $^*aa$. However, this also incorrectly rules out the well-formed aa^+bb. Given that a^+b is also not part of the language, we should also project every b on T. Since the number of as on the tier is potentially unbounded, banning a^+b strings will again result in blocking aa^+bb, or ab^+. Thus, L is LT but not SS-TSL.

For PT $\nsubseteq$ SS-TSL, we pick the same example as before and we assume that, in the formula $\varphi := aa \rightarrow bb$, aa and bb are predicates based on *precedence* (i.e. denoting subsequences) instead than based on *immediate precedence* (denoting substrings). This language is PT, but again 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 $\subsetneq$ SF*

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. Similarly, we can define a new class of languages that are the intersection of finitely many SS-TSL languages.

Definition 10. (Structure-Sensitive MTSL) Consider a finite set of (k,m)-SS-TSL languages $\mathscr{L} = \{L_1, \ldots, L_n\}$, with $n = |\mathscr{L}|$. A structure-sensitive multi-tier strictly local ((k,n,m)-SS-MTSL) language L is defined as $L := \bigcap_{1 \leq i \leq n} L_i$.
We write k-SS-MTSL^m_n instead of (k,n,m)-SS-MTSL. We omit k,n and m when convenient.

Clearly, it is also possible to characterize SS-MTSL languages in terms of FO logic, using *tier-precedence* $\blacktriangleleft_T$ as defined for SS-TSL. Since every SS-MTSL language is the intersection of finitely many TSL languages, the formulae for SS-MTSL languages are conjunctions of SS-TSL formulae.

We can now place the class of SS-MTSL languages with respect to the rest of the subregular hierarchy.

Lemma 10. *MTSL $\subsetneq$ SS-MTSL*

Proof. MTSL $\subseteq$ SS-MTSL follows from definition, TSL being the set of languages in SS-MTSL^m with $m = 1$. Proper inclusion is a corollary of Lemma 7, which states that MTSL and SS-TSL are incomparable. □

This result also clarifies that SS-TSL and SS-MTSL are in a proper subsumption relationship.

Theorem 11. *SS-TSL $\subsetneq$ SS-MTSL*

Proof. Inclusion is trivial. It is proper since MTSL and SS-TSL are incomparable, and MTSL $\subsetneq$ SS-MTSL. □

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

Lemma 12. *SS-MTSL $\subsetneq$ 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.*

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. □

5.4 Closure Properties

The current characterization of SS-TSL still lacks a discussion of its closure properties.

Clearly, SS-TSL is not closed under intersection This follows from the fact that SS-TSL is properly included in SS-MTSL, and from Definition 10.

Moreover, both SS-TSL and SS-MTSL are not closed under relabeling. Simply consider the SL (thus SS-TSL, SS-MTSL) 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.

To shown that SS-TSL is not closed under union, all we need is to show that the union of SS-TSL produces languages that cannot be captured by extending the locality of the ISL erasing function. In fact, we can exploit the counter-examples used in De Santo (2017)'s non-closure proofs for TSL and MTSL languages.

First, we need to introduce some additional notation. 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 4.2, and may be a proper superset of L.

Now, let $\Sigma := \{a,b,c\}$ and L_1 and L_2 the largest Σ-languages such that $\{a,b\}(L_1) := a^+b^+$ and $\{a,b\}(L_2) := b^+a^+$. L_1 and L_2 are 2-TSL (thus 2-SS-TSL[1]) and that $L_1 \cup L_2 = \downarrow T(L_1 \cup L_2)$ iff $T := \{a,b\}$. But $\{a,b\}(L_1 \cup L_2) = a^+b^+ \cup b^+a^+$, and it is not closed under k-local suffix substitution for any choice of k:

$$\begin{array}{llllll} & & \overbrace{}^{x} & & & \\ a & b\cdots b & b & & & \in L \\ & b\cdots b & a & a & & \in L \\ \hline a & a\ b\cdots b & a & a & & \notin L \end{array}$$

Thus, $L_1 \cup L_2$ is not TSL. The problem here is that the union of two TSL languages results is a tier-language that is not strictly local. This cannot be fixed by moving to an SS-TSL grammar. To rule out strings like $b^+, a^+, \{a,b\}^+a^+$, we still need to project a,b on the tier, independently of the locality of the ISL projection function.

Although this is not an exhaustive formal proof, the counter-example illustrates the essential insight: that the union of TSL includes languages that need every element of the alphabet on the tier, and therefore cannot be described by simply increasing the locality of the structural projection.

The actual proof is convoluted and not particularly informative, since it relies even more heavily on technical notation, in order to consider all possible combination of structural tier-projection. Thus, we omitted it here, preferring to outline the essential underlying intuitions instead.

Proposition 2. SS-TSL is not closed under intersection, union, and relabelings.

6 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. The final hierarchy of subregular classes as discussed in this paper is shown in Figure 5.

This figure also includes a class of languages — not discussed here — that further extends TSL by allowing projections of tiers from tiers as a cascade of ISL erasing functions. Preliminary work indicates that this extension (TESL) properly includes SS-TSL, and that its intersection closure (MTESL) subsumes even SS-MTSL languages. Future work on the properties of these classes might reveal whether they have any practical use for linguistics.

Furthermore, we could gain a more comprehensive understanding of the *refined* subregular hierarchy as developed in recent years by comparing SS-TSL and De Santo (2017)'s MTSL to the class of interval-based strictly piecewise languages introduced by Graf (2016) as an extension of TSL, SL and SP.

6.1 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 strings of 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

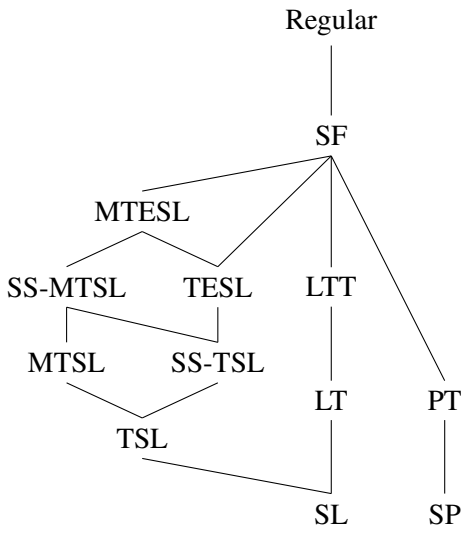

Figure 5: Proper inclusion relationships among subregular language classes. This paper establishes the SS-TSL and SS-MTSL. It also lays the ground for further explorations of TESL and MTESL.

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 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 14. *The classes of SS-TSL and SS-MTSL 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 TSL extensions. Although learners based on 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*. Jardine and McMullin (2016) further extend this result, and establish a learner that is guaranteed to induce a TSL_k grammar in polynomial time and data. The latter seems to be easily adaptable to SS-TSL, by inducing a tier of segments based on their k-local properties. Moreover, 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, McMullin and Allen (2015) propose a learner for conjoined TSL languages that exploits search over lattices. While their implementation still lacks generality, in theory it should also work for SS-MTSL.

Overall, the extensive amount of work on efficient learners for subregular classes encourages us to propose the following conjecture:

Conjecture 1. *The classes of SS-TSL and SS-MTSL are efficiently learnable from a polynomial sample size in polynomial time.*

Future work will focus on adapting some of the algorithms described above to SS-TSL, in order to test this conjecture and compare performances. Among alternative approaches worth exploring, Goldsmith and Riggle (2012) propose a tier-based learner for harmony patterns relying on mutual information.

6.2 Implications for Phonology

In Section 2, we argued that the classes discussed in this paper are motivated by the existence of phonotactic processes that exceed the expressive power of TSL. These patterns can easily be covered by SS-TSL languages, which provide an erasing function sensible to local properties of the segments in the string.

For instance, recall the harmony process in Samala, which combined a long-distance sibilant harmony with local dissimilation between /s/ and /n/. This kind of expressivity can now be accomplished by increasing the locality window of the *tier projection mechanism*.

For example, Figure 6 shows how, by increasing the locality of the projection to 2, we allow the grammar to project $[n]$ iff it is immediately preceded by a sibilant in the input string, and then use 3-local tier constraints to ban $\{{}^*sn(\neg s), {}^*\int ns\}$, in addition to the factors needed to enforce the usual sibilant harmony patterns.

This time, the possible unboundedness of /n/ is not a problem, since/n/ is now relevant for the projection only when adjacent to a sibilant.

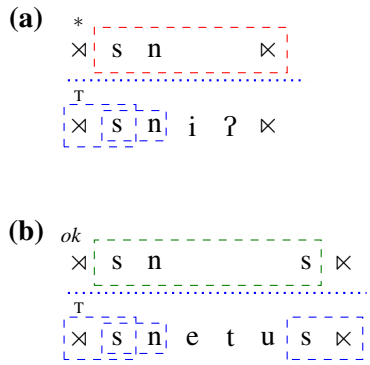

Figure 6: Example from Samala: (a) is ill-formed because of adjacent *sn; (b) is well-formed since [sn] is followed by another [s] later in the string. Note that [n] is projected on the tier only when adjacent to [s]

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. The grammar for Samala, for example, can easily be modified to enforce harmony between the first and last segment to the exclusion of any other material, by projecting only sibilants in initial/final position. Not only is such a pattern unattested, experimental evidence suggests that it is never entertained as a possible phonological dependency (Lai, 2015).

In future, we should 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 SS-TSL against the constraint-based approach to TSL proposed by McMullin (2016) along the lines of Optimality Theory (OT; (Prince and Smolensky, 2008)). The idea is to account for phonotactic patterns in which local and non-local dependencies interact, by representing each OT constraint as individual, ranked, and violable 2-TSL grammars.

Most of the patterns covered by SS-TSL can be captured this way. However, the operation of ranking TSL languages is still lacking a proper formalization, and it is not clear whether the generative complexity of the formalism is still subregular.

Interestingly, OT systems can be implemented through finite-state transducers, if each constraint distinguishes among only a finite set of equivalence classes of candidates (Frank and Satta, 1998). Then, this paper approach to TSL in terms of cascades of ISL functions should make it easier to study TSL rankings, once the effects of composing ISL functions are better understood.

Evidently, to really assess the linguistic usefulness of what we can now call the *TSL neighborhood*, there is need for a more careful typological exploration. In particular, moving to linguistic domains in which attempts to a subregular 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 the TSL extensions in Figure 5.

6.3 Implications for Syntax

Since the current consensus is that the requirements of syntax greatly exceed the computational power of regular languages[1], it is probably not particularly surprising that most of the work connecting notions in the subregular hierarchy to natural language processes has been done in phonology.

Recently, Graf (2014) proposed that syntactic dependencies 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 subregular 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 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. 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 Conclusions

A growing body of literature is exploring TSL languages as a good computational hypothesis for the complexity of phonotactic patterns. However,

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

the TSL class comes with particularly tight constraints on the projection function. Here we relax some of these constraints, allowing for more general definitions of tier-projection. The resulting new class naturally extends TSL, and easily captures patterns — previously problematic for TSL accounts — in which local and non-local dependencies interact. The fact that a few minor modifications to TSL allow us to cover previously unaccounted patterns, while keeping the generative power in check, supports future studies of this region. 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 subregular 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. Multi-tier strictly local languages. Ms., Stony Brook University.

Robert Frank and Giorgio Satta. 1998. Optimality theory and the generative complexity of constraint violability. *Comput. Linguist.*, 24(2):307–315, June.

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.

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

Adam Jardine and Kevin J. McMullin. 2016. Efficient learning of tier-based strictly k-local languages. In Frank Drewes, Carlos Martin-Vide, and Bianca Truthe, editors, *Proceedings of Language and Automata Theory and Applications, 11th International Conference*, Lecture Notes in Computer Science. Springer. To appear.

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

Regine Lai. 2015. Learnable vs. unlearnable harmony patterns. *Linguistic Inquiry*, 46(3):425–451.

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 McMullin and Gunnar Hansson. 2016. Long-distance phonotactics as tier-based strictly 2-local languages. *Proceedings of the Annual Meetings on Phonology*, 2(0).

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.

José Oncina, Pedro García, and Enrique Vidal. 1993. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):448–458.

Alan Prince and Paul Smolensky, 2008. *Optimality Theory: Constraint interaction in Generative Grammar*. Blackwell Publishing Ltd.

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.