Check for updates

Structure Sensitive Tier Projection: Applications and Formal Properties

Aniello De Santo(✉) and Thomas Graf

Department of Linguistics, Stony Brook University, Stony Brook, USA
aniello.desanto@stonybrook.edu, mail@thomasgraf.net

Abstract. The subregular approach has revealed that the phonological surface patterns found in natural language are much simpler than previously assumed. Most patterns belong to the subregular class of tier-based strictly local languages (TSL), which characterizes them as the combination of a strictly local dependency with a tier-projection mechanism that masks out irrelevant segments. Some non-TSL patterns have been pointed out in the literature, though. We show that these outliers can be captured by rendering the tier projection mechanism sensitive to the surrounding structure. We focus on a specific instance of these *structure-sensitive TSL* languages: input-local TSL (ITSL), in which the tier projection may distinguish between identical segments that occur in different local contexts in the input string. This generalization of TSL establishes a tight link between tier-based language classes and ISL transductions, and is motivated by several natural language phenomena.

Keywords: Subregular hypothesis · TSL · Phonotactics · Input strictly local functions · Generative capacity

1 Introduction

The subregular hypothesis ([16] and references therein) posits that every language's set of phonologically well-formed surface strings—its phonotactic patterns—belongs to a proper subclass of the regular languages. The class of tier-based strictly local languages (TSL) has been of particular interest in this respect [17]. TSL is inspired by autosegmental phonology [12] and combines two components: (I) an n-gram based mechanism to enforce local constraints on adjacent segments, and (II) a tier projection mechanism that "masks out" irrelevant parts of the string. Long-distance dependencies are thus reanalyzed as local dependencies over strings with masked out segments.

While TSL covers a wide range of data, recent literature has reported several instances of complex phenomena—from Samala sibilant harmony to unbounded tone plateauing—that cannot be characterized in these terms [14,15,24, a.o.]. We argue that all these counterexamples can be accounted for by extending the tier projection mechanism. We redefine TSL as a cascade of three string transductions, one of which is the tier projection mechanism. In standard TSL, the

R. Bernardi et al. (Eds.): FG 2019, LNCS 11668, pp. 35–50, 2019.
https://doi.org/10.1007/978-3-662-59648-7_3

tier projection is an input strictly local function of locality 1 (1-ISL) in the sense of Chandlee [5, Definition 4]. By allowing for more complex string transductions, one obtains the much more powerful class of *structure sensitive TSL* (SS-TSL). Within this wide range of options, we focus on the natural generalization from 1-ISL to n-ISL. This means that projection of a segment s does not merely depend on s alone but may also consider the locally bounded context $u_1 \cdots u_m \_ v_1 \cdots v_n$ in which s occurs. The resulting class of *input tier-based strictly local* (ITSL) languages greatly expands the empirical coverage of TSL while retaining essential formal properties.

The paper is structured as follows. Section 2 introduces mathematical notation that is essential for studying subregular languages. The fundamental properties of strictly local (SL) and tier-based strictly local (TSL) languages are presented in Sect. 3. There, we also introduce the first major innovation of this paper, the generalization from standard TSL to SS-TSL. We then define ITSL, the most natural subclass of SS-TSL. Section 4 studies the formal properties of ITSL, and relates it to the rest of the subregular hierarchy. We then expand on this with results on the intersection closures of TSL and ITSL, respectively (Sect. 5). Finally, Sect. 6 discusses the implications of these results for learnability.

2 Preliminaries

This paper discusses TSL and our generalization of its projection function. As we compare the resulting new languages to several subregular classes besides TSL, a fair amount of mathematical machinery is required. We assume familiarity with set notation on the reader's part.

Given a finite alphabet Σ, Σ^* is the set of all possible finite strings of symbols drawn from Σ. A language L is a subset of Σ^*. The concatenation of two languages $L_1L_2 = \{uv : u \in L_1 \textit{ and } v \in L_2\}$. For every string w and every non-empty string u, $|w|$ denotes the length of the string, $|w|_u$ denotes the number of occurrences of u in w, and ε is the unique empty string. Left and right string boundaries are marked by $\rtimes, \ltimes \notin \Sigma$ respectively.

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

$$F_k(w) := \{u : u \textit{ is a k-factor of } w \textit{ if } |w| \geq k, \textit{ else } u = w\}$$

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

$$F_{k,t}(w) := \{(u, n) : u \textit{ is a k-factor of } w \textit{ and } n = \min(|w|_u, t)\}$$

For example $F_{2,5}(aaaaab) = \{(aa, 4), (ab, 1)\}$, but $F_{2,3}(aaaaab) = \{(aa, 3), (ab, 1)\}$.

In order to simplify some proofs, we rely on first-order logic characterizations of certain string languages and string-to-string mappings. 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 $w \in \Sigma^*$. Note that $\prec$ is a strict total order.

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 $x \triangleleft y \Leftrightarrow x \prec y \wedge \neg\exists z[x \prec z \wedge z \prec y]$. 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 [26]. If $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$, then a string $w \in \Sigma^*$ can be represented as a structure M_w in the signature$(\sigma_1(\cdot), \ldots, \sigma_n(\cdot), \prec)$. If φ is a logical formula without any free variables, we use $L(\varphi) = \{w \in \Sigma^* \mid M_w \text{ satisfies } \varphi\}$ as the stringset extension of φ .

3 Structure-Sensitive TSL Languages

There is a rich literature exploring the subclasses that the regular languages can be divided into [4,9,27,32, a.o.]. Among these *subregular* classes, *tier-based strictly local* languages (TSL; [17]) have received particular attention due to their ability to provide natural descriptions of phonological well-formedness conditions (see also [13,19,29]). TSL extends the class of *strictly local* languages (SL) with a tier projection mechanism that renders non-local dependencies in a string local over tiers. The projection mechanism is very limited though, as it only considers a segment's label but not its structural context. This is too restrictive for phonology, which is why we extend TSL to a class of languages sensitive to structural information: TSL where tier projection can take local information into account.

3.1 Strictly Local and Tier-Based Strictly Local Languages

SL is the class of languages that can be described in terms of a finite number of forbidden substrings. Intuitively, SL languages describe patterns which depend solely on the relation between a bounded number of consecutive symbols in a string—there are no long-distance dependencies.

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

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

We also call S a strictly k-local grammar, and we also use $L(S)$ to indicate the language generated by S. A language L is strictly local iff it is SL_k for some $k \in \mathbb{N}$.

For example, $(ab)^n$ is a strictly 2-local language over alphabet $\{a, b\}$ because it is generated by the grammar $G := \{\rtimes b, bb, aa, a\ltimes\}$.[1]

Even though this paper is concerned with extensions of SL, many of our proofs make use of a particular characterization of SL in terms of k-local suffix substitution closure [30].

Definition 2 (Suffix Substitution Closure). *For any $k \geq 1$, a language L satisfies k-local suffix substitution closure iff for all strings u_1, v_1, u_2, v_2, for any string x of length $k-1$ if both $u_1 \cdot x \cdot v_1 \in L$ and $u_2 \cdot x \cdot v_2 \in L$, then $u_1 \cdot x \cdot v_2 \in L$.*

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

The language $L := a^*ba^*$, for example, is not SL because for any k we can pick two strings $a^m ba^k \in L$ and $a^k ba^n \in L$ and recombine them into $a^m ba^k ba^n \notin L$. However, this language is TSL.

TSL is an extension of SL where k-local constraints only apply to elements of a tier $T \subseteq \Sigma$. An erasing function (also called projection function) is introduced to delete all symbols that are not in T. Given some $\sigma \in \Sigma$, the erasing function $E_T : \Sigma \to \Sigma \cup \{\varepsilon\}$ maps σ to itself if $\sigma \in T$ and to *mptystring* otherwise.

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

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

Definition 3 (TSL). *A language L is* tier-based strictly k-local *(TSL_k) 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*

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

We also call S the set of forbidden k-factors on tier T, and $\langle S, T\rangle$ is a TSL_k grammar.

As can be gleaned from Definition 3, a language L is TSL iff it is strictly k-local on tier T for some $T \subseteq \Sigma$ and $k \in \mathbb{N}$. This will be important for many proofs.

For a concrete example, consider once more $L := a^*ba^*$ such that $aba, aabaa, aaaba \in L$ but $abaabaa, ababaa \notin L$. This language is generated by the TSL_2 grammar $\langle \{\rtimes\ltimes, bb\}, \{b\}\rangle$ over $\Sigma = \{a, b\}$, which bans every string whose tier is empty (no b) or contains more than one b.

[1] A comment regarding edge markers. For S to be k-local, it needs to contain only factors of length k. Thus, strings are augmented with enough 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. So $\rtimes \rtimes a$ is truncated to $\rtimes a$. We adopt this simpler notation throughout the paper, unless required to make a definition clearer.

3.2 Insufficiency of TSL

While TSL enjoys wide empirical coverage in phonology, some non-TSL phenomena have been pointed out in the literature [14,15,24]. As a concrete example, consider the case of sibilant harmony in Samala, where an unbounded dependency can override a local one (see [2] for the original data set and [24] for a subregular analysis). Samala displays sibilant harmony such that [s] and [ʃ] may not co-occur anywhere within the same word (cf. Ex. (1a)). There is also a ban against string-adjacent [st], [sn], [sl], which is resolved by dissimilation of [s] to [ʃ] (cf. Ex. (2a) and (2b)). However, dissimilation is blocked if the result would violate sibilant harmony. Thus /sn/ surfaces as [ʃn] unless the word contains [s] somewhere to the right, in which case it is realized as [s] (cf. Ex. (2a) and (3a)).

(1) a. /k-su-ʃojin/ → [kʃuʃojin]

(2) a. /s-niʔ/ → [ʃniʔ]
  b. /s-niʔ/ → *[sniʔ]

(3) a. /s-net-us/ → [snetus]
  b. /s-net-us/ → *[ʃnetus]

This pattern is not TSL. Pick some sufficiently large m and consider the strings [sne(ne)^{m}tus] and [ne(ne)^{m}tus], which are well-formed according to the generalization above. In stark contrast, the minimally different [sne(ne)^{m}tu] is ill-formed. In order to regulate this dependency, we need a TSL grammar whose tier contains at least [s] and [n]. But then the tiers of these three strings are of the form snn^{m}s, nn^{m}s, and snn^m, respectively. By suffix substitution closure, it is impossible for an SL grammar to allow the former two while forbidding the latter. But if the tier language is not SL, the original language is not TSL, either. Note that projecting additional symbols does not change anything with respect to suffix substitution closure, so the problem is independent of what subset of Σ one chooses as the tier alphabet.

The central shortcoming of TSL is that it only provides a choice between projecting no instance of [n], which is obviously insufficient, and projecting every instance of [n], which renders the dependency between sibilants non-local over tiers. But suppose that one could instead modify the projection function such that an [n] is projected iff it is immediately preceded by a sibilant. Then [sne(ne)^{m}tus] and [ne(ne)^{m}tus] have the tiers sns and s, whereas [sne(ne)^{m}tu] has the tier sn. An SL_3 grammar can easily distinguish between these, permitting the former two but not the latter. Such a modified version of TSL will also be able to block [snetu] while allowing for [senetu] as their respective tiers are sn and s. Apart from this Samala example, reported non-TSL patterns that can be accounted for by inspecting the local context of a segment before projecting it include nasal harmony in Yaka [33], unbounded stress of Classical Arabic (see [3] and references therein), Korean vowel harmony [14], and cases of unbounded tone plateauing [20, a.o.].

More recently, other patterns have been reported for which it seems to be necessary to extend TSL projections to consider more than just local contexts in the input string. Mayer and Major [23], based on a suggestion by Graf (p.c.), make tier-projection sensitive to preceding segments on the tier in order to capture backness harmony in Uyghur. Graf and Mayer [15] analyze Sanskrit retroflexion in terms of an even more general class whose projection function considers the local contexts in both the input string and the already constructed tier.

Crucially, all these extensions allow the erasing function E_T to consider additional structural factors. We call all languages in which the projection function has been extended along these lines *structure-sensitive* TSL. This is a very loosely defined class, but as we explain next the idea can be made more precise by viewing TSL-like grammars as a cascade of three string transductions.

3.3 TSL as the Composition of Three Transductions

For every TSL grammar $G := \langle S, T\rangle$, one can construct a sequence of transductions that generates exactly the same string language:

1. The *projection transduction* E_T rewrites every symbol $s \in T$ as s and deletes every $s' \notin T$.
2. The *grammar transduction* id_S is the identity function over $L(S)$.
3. The *filler transduction* F_T is the inverse of E_T.

Their composition $E_T \circ \mathrm{id}_S \circ F_T$ is a partial, non-deterministic finite-state transduction. The image of Σ^* under this transduction is exactly $L(G)$. All the recent extensions of TSL keep id_S the same, but they change the nature of E_T (and hence F_T). Without further limitations on E_T, every recursively enumerable string language can be generated this way. But from a linguistic perspective, this is immaterial as only very limited kinds of SS-TSL have been proposed. These classes generalize E_T to ISL or OSL functions as originally defined in [5]. We only consider the former here and leave the latter for future work.

3.4 Input-Sensitive TSL

Adding input-sensitivity to TSL only requires a minor change to the definition of E_T. In order to simplify the exposition later on, we take inspiration from [7] and define ISL projections in terms of local contexts.

Definition 4 (Contexts). *A* k-context *c over alphabet Σ is a triple $\langle\sigma, u, v\rangle$ such that $\sigma \in \Sigma$, $u, v \in \Sigma^*$ and $|u| + |v| \leq k$. A* k-context set *is a finite set of k-contexts.*

Definition 5 (ISL Projection). *Let C be a k-context set over Σ (where Σ is an arbitrary alphabet also containing edge-markers). Then the* input strictly k-local *(ISL-k) tier projection π_C maps every $s \in \Sigma^*$ to $\pi'_C(\rtimes^{k-1}, s\ltimes^{k-1})$, where $\pi'_C(u, \sigma v)$ is defined as follows, given $\sigma \in \Sigma \cup \{\varepsilon\}$ and $u, v \in \Sigma^*$:*

$$\begin{array}{ll} \varepsilon & \textit{if } \sigma a v = \varepsilon, \\ \sigma\pi'_C(u\sigma, v) & \textit{if } \langle\sigma, u, v\rangle \in C, \\ \pi'_C(u\sigma, v) & \textit{otherwise.} \end{array}$$

Note that an ISL-1 tier projection only determines projection of σ based on σ itself, just like E_T does for TSL. This shows that ISL-k-tier projections are a natural generalization of E_T even though they are no longer defined in terms of some $T \subseteq \Sigma$. The definition of ITSL languages then closely mirrors the one for TSL.

Definition 6 (ITSL). *A language L is m-*input local *k-*TSL *(m-ITSL$_k$) iff there exists an m-context set C and a finite set $S \subseteq \Sigma^k$ such that*

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

A language is input-local TSL *(ITSL) iff it is m-ITSL$_k$ for some $k, m \geq 0$. We call $\langle S, C\rangle$ an ITSL grammar.*

Let us return to the interaction of local dissimilation and non-local harmony in Samala. This process can be handled by an 2-ITSL$_3$ grammar $\langle S, C\rangle$ with

- $S := \{\text{s}\int, \int\text{s}, \text{snx}\}$ where $\text{x} \in \{\Sigma - s\}$,
- C contains all of the following contexts, and only those:
  - $\langle s, \varepsilon, \varepsilon\rangle$
  - $\langle S, \varepsilon, \varepsilon\rangle$
  - $\langle n, s, \varepsilon\rangle$

Since this phenomenon could not be handled with TSL, ITSL properly extends TSL.

Theorem 2. *TSL $\subsetneq$ ITSL*

For the sake of rigor, we also provide a formal proof.

Proof. TSL $\subseteq$ ITSL is trivial. Now consider the language $L = a\{a, b\}^*b \cup b\{a, b\}^*a$ over alphabet $\Sigma = \{a, b\}$. It is generated by the 2-ITSL$_2$ grammar $\langle S, C\rangle$ with $S = \{aa, bb, \rtimes\ltimes\}$ and $C := \{\langle\sigma, \rtimes, \varepsilon\rangle, \langle\sigma, \varepsilon, \ltimes\rangle \mid \sigma \in \Sigma\}$. But L is not TSL. Pick some arbitrary TSL$_k$ grammar $\langle S, T\rangle$ and strings $s := a^m b^n \in L$, $t := b^n a^o \in L$, and $u := a^m b^n a^o \notin L$ $(m, n, o > k)$. These three strings witness that no matter how one chooses $T \subseteq \Sigma$, the resulting tier language is not closed under suffix substitution closure. Thus, L is not k-TSL for any k.

ITSL is clearly more powerful than TSL, but the question is how much additional power the move to ISL projections grants us. We do not want ITSL to be too powerful as it should still provide a tight characterization of the limits of natural language phonology. The next section shows that ITSL is still a very conservative extension of TSL that is subsumed by the star-free languages and largely incomparable to any other subregular classes.

4 Formal Analysis

It is known that TSL is a proper subclass of the star-free languages (SF) and is incomparable to the classes locally testable (LT), locally threshold-testable (LTT), strictly piecewise (SP), and piecewise testable (PT) [17]. In addition, TSL is not closed under intersection, union, complement, concatenation, or relabeling (this is common knowledge but has not been explicitly pointed out in the literature before). The same holds for ITSL. This is not too surprising as ITSL is a fairly minimal extension of TSL, and many of the proofs in this section are natural modifications of the corresponding proofs for TSL.

4.1 Relations to Other Subregular Classes

First we have to provide basic definitions for subregular classes we wish to compare to ITSL.

Definition 7 *(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 *locally threshold testable* (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, the *locally testable* (LT) languages are *locally threshold testable* with $t = 1$.

We show that LT and ITSL are incomparable. Since TSL and LTT are known to be incomparable [17], the incomparability of LTT is an immediate corollary.

Theorem 3. *ITSL is incomparable to LT and LTT.*

Proof. That ITSL is no subset of LT or LTT follows from the fact that ITSL subsumes TSL, which is incomparable to both.

We now show that LT $\nsubseteq$ ITSL. Let L be the largest language over $\Sigma = \{a, b, c\}$ such that a string contains the substring aa only if it also contains the substring bb. This language is LT but cannot be generated by any m-ITSL$_k$ grammar G, irrespective of the choice of k and m.

Suppose G generates at least strings of the form $c^*aac^*bbc^* \in L$ and $c^*bbc^* \in L$, but not $c^*aac^* \notin L$. Then G must project both aa and bb, wherefore c^*aac^* and c^*bbc^* each license projection of aa and bb, respectively (projection of one of a or b cannot depend on the other because the number of cs between the two is unbounded). But then strings of the form $(c^*aac^*)^+bb(c^*aac^*)^+ \in L$ yield a tier language $(aa)^+bb(aa)^+$. By suffix substitution closure, G also accepts any tier of the form $(aa)^+$. Therefore, $L(G) \ni (c^*aac^*)^+ \notin L$.

Next consider the *strictly piecewise* (SP) and *piecewise testable* (PT) languages [10,28,31]. These are already known to be incomparable with SL, TSL, and LTT. For any given string w, let $P_{\leq k}(w)$ be a function that maps w to the set of subsequences up to length k in w.

Definition 8 *(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.*

Properly included in PT, SP languages mirror the definition of SL languages by replacing $F_k(w)$ with $P_k(w)$ in Definition 1. In short, piecewise languages are sensible to relationships between segments based on *precedence* (over arbitrary distances) rather than *adjacency* (immediate precedence).

Theorem 4 *ITSL is incomparable to SP and PT.*

Proof ITSL $\not\subseteq$ SP, PT follows from the fact that ITSL includes TSL, which is incomparable to both. In the other direction, consider the SP language L that consists of all strings over $\Sigma = \{a, b, c, d, e\}$ that do not contain the subsequences ac or bd. This language is not ITSL. In order to correctly ban both ac and bd, at least one instance of a, b, c, and d must be projected in each string. Consequently, for each symbol there must be some fixed context that triggers its projection. Assume w.l.o.g. that one of these contexts is $\langle b, u, v \rangle$. Consider the strings $s := a(e^m ubv)^n \in L$, $t := (e^m ubv)^n c \in L$, and $u := a(e^m ubv)^n c \notin L$, for sufficiently large m and n. The respective tiers are $s' := ab^n$, $t' := b^n c$, and $u' := ab^n c$. By suffix substitution closure, no SL language can contain s' and t' to the exclusion of u', wherefore L is SP (and PT) but not ITSL.

The last subregular class relevant to our discussion is SF. Multiple characterizations are known, but we will use the one in terms of first-order logic as it greatly simplifies the proof that ITSL is subsumed by SF.

Definition 9 *(Star-Free)***.** *Star-free (SF) languages are those that can be described by first order logic with precedence.*

Theorem 5. *ITSL $\subsetneq$ SF.*

Proof. Subsumption follows from the fact that every ITSL language can be defined in first-order logic with precedence. Proper subsumption then is a corollary of LT, PT $\subseteq$ SF together with Theorems 3 and 4.

We briefly sketch the first-order definition of ITSL. First, the successor relation $\triangleleft$ is defined from precedence in the usual manner. Then, for every context $c := \langle \sigma, u_1 \cdots u_m, u_{m+1} \cdots u_n \rangle$ one defines a predicate $C(x)$ as

$$\exists y_1, \ldots, y_{m+n} \Big[\sigma(x) \wedge \bigwedge_{1 \leq i < m} y_i \triangleleft y_{i+i} \wedge y_m \triangleleft x \wedge x \triangleleft y_{m+1} \wedge \bigwedge_{m+1 \leq i < n} y_i \triangleleft y_{i+i} \wedge \bigwedge_{1 \leq i \leq n} u_i(y_i) \Big]$$

The context predicates form the basis for the ITSL tier predicate

$$T(x) \Leftrightarrow \bigvee_{C \text{ is a context predicate}} C(x)$$

which in turns allows us to relativize precedence to symbols on the tier:

$$x \triangleleft_T y \Leftrightarrow T(x) \wedge T(y) \wedge x \prec y \wedge \neg\exists z[T(z) \wedge x \prec z \wedge z \prec y]$$

The set of forbidden k-factors then is just a conjunction of negative literals with $\triangleleft_T$ as the basic relation.

4.2 Closure Properties

The previous section established that ITSL is a natural generalization of TSL in the sense that it displays the same (proper) subsumption and incomparability relations with respect to other classes. We now show that this parallelism between TSL and ITSL also carries over to the standard closure properties. Just like TSL, ITSL is not closed under intersection, union, complement, concatenation, or relabeling.

We start with non-closure under intersection.

Lemma 1. *ITSL is not closed under intersection.*

Proof. Consider again the SP language L that consists of all strings over $\{a, b, c, d, e\}$ that do not contain the subsequences ac or bd. As shown in Theorem 4, this language is not ITSL. But L is the intersection of two TSL (and hence ITSL) languages L_1 and L_2 s.t. $T_1 = \{a, c\}$, $S_1 = \{ac\}$ and $T_2 = \{b, d\}$, $S_2 = \{bd\}$. Thus closure under intersection does not hold.

Lemma 2. *ITSL is not closed under concatenation.*

Proof. Let L be the union of $ab\{a, b, c\}^* a$ and $ba\{a, b, c\}^* b$. This language is ITSL. The context set is $C := \{\langle\sigma, \rtimes, \varepsilon\rangle, \langle\sigma, \varepsilon, \ltimes\rangle, \langle\sigma, \rtimes\sigma', \varepsilon\rangle \mid \sigma, \sigma' \in a, b, c\}$, and the only allowed k-factors are $\rtimes aba \ltimes$ and $\rtimes bab \ltimes$. Now consider the string $s_1 := abc^k bc^k b$, which is not in the concatenation closure of L. Nor is its iteration s_1^m. But the concatenation closure of L does contain $s_2 := s_1^m abs_1^m$, as this is an instance of $ab\{a, b, c\}^* a$ concatenated with $ba\{a, b, c\}^* b$. Every k-context of s_1^m is also a k-context of s_2. Hence every m-factor of s_1^m is also an m-factor of s_2. Therefore it is impossible for any k-ITSL$_m$ grammar G to contain s_2 to the exclusion of s_1. It follows that the concatenation closure of L is not k-ITSL for any k.

Lemma 3. *ITSL is not closed under union.*

Proof. Let $C := \{\langle a, \varepsilon, \varepsilon\rangle, \langle b, \varepsilon, \varepsilon\rangle\}$ and consider the SL$_2$ languages a^+b^+ and b^+a^+. Let L_{ab} and L_{ba} be the respective images of these languages under π_C^{-1} given alphabet $\{a, b, c\}$. That is to say, $L_{ab} := (c^*a)^+(c^*b)^+c^*$ and $L_{ba} := (c^*b)^+(c^*a)^+c^*$. By definition, L_{ab} and L_{ba} are ITSL languages, but their union L is not. Note that $s_1 := (c^k a)^m c^k \notin L$, whereas $s_2 := s_1^m (c^k b^k)^m c^k \in L$ and $s_3 := (c^k b^k)^m s_1^m c^k \in L$. Every k-context of s_1 also occurs in s_2 and s_3. This implies that no matter what k-context set one picks, all the m-factors of the tier of s_1 are also m-factors of the tiers of s_2 or s_3. As with concatenation closure, this makes it impossible to ban s_1 while allowing for s_2 and s_3.

The same string embedding strategy can also be used for relative complement.

Lemma 4. *ITSL is not closed under relative complement.*

Proof. For simplicity, we only prove non-closure under complement relative to Σ^* (this suffices because Σ^* is ITSL). Let C be as before, and consider the SL_2 language a^+b. The image under π_C^{-1} is the ITSL language $L := (c^*a)^+c^*bc^*$. Consider the string $s_1 := (c^ka)^m c^k bc^k \in L$. The complement $\overline{L}$ of L does not contain s_1, but it contains its mirror immage $s_{-1} := c^kbc^k(ac^k)^m$ and the concatenation of s_1 with itself: $s_{11} := (c^ka)^m c^kbc^k(c^ka)^m c^kbc^k \in \overline{L}$. But as before, every conceivable k-context of s_1 is also a k-context of and s_{-1} and s_{11}. Any illicit m-factor in the tier of s_1 will also occur in the tier of s_{-1} or s_{11}. Again once cannot rule out s_1 without also ruling out s_{-1} or s_{11}, which proves that $\overline{L}$ is not ITSL.

For non-closure under relabeling, a much simpler strategy suffices. Simply consider the SL (and thus ITSL) language $L_{ab} = (ab)^+$. A relabeling that replaces b by a maps L_{ab} to $L_{aa} = (aa)^+$, which isn't even star-free.

Theorem 6. *ITSL is not closed under intersection, union, relative complement, concatenation, and relabelings.*

While these closure properties may seem unappealing from a mathematical perspective, they mirror exactly the closure properties of TSL. This confirms our original claim that ITSL is a natural generalization of TSL. In addition, the lack of most of the canonical closure properties is welcome from a linguistic perspective because natural languages do not seem to display these closure properties either. That said, closure under intersection is a linguistically important property, which is why we explore it in depth in the next section.

5 Intersection Closure of TSL and ITSL

Lack of closure under intersection is problematic as it entails that the complexity of phonological dependencies is no longer constant under factorization. Depending on whether one treats a constraint as a single phenomenon or the interaction of multiple phenomena, the upper bound for phonological complexity will shift. Neither TSL nor ITSL are closed under intersection, yet they both are reasonable formal approximations of phonological dependencies. In order to understand what (I) TSL claims about individual phenomena imply about the complexity of phonology as a whole, we need a good formal understanding of the intersection closure of TSL (Sect. 5.1) and ITSL (Sect. 5.2).

5.1 Intersection Closure of TSL Languages

The intersection of two TSL languages can be regarded as a language that is produced by a single TSL grammar that projects multiple tiers. For this reason,

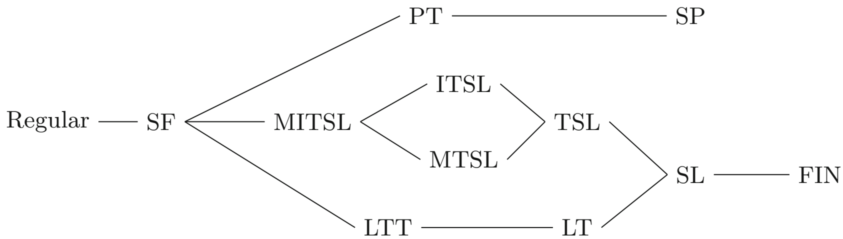

Fig. 1. Proper inclusion relationships of subregular classes. Subsumption goes left-to-right. We establish MTSL, ITSL, and MITSL.

we refer to the intersection closure of TSL as multi-TSL (MTSL). We write n-MTSL$_k$ to indicate a grammar where n is the number of tiers and k is the locality of the tier-constraints. Note that we frequently omit k and n to reduce clutter.

Definition 10. *An n-tier strictly k-local (n-MTSL$_k$) language L is the intersection of n distinct k-TSL languages ($k, n \in \mathbb{N}$).*

MTSL is a proper superclass of TSL, which is witnessed by the language we used to prove non-closure under intersection for ITSL. This also shows that MTSL is not subsumed by ITSL. The opposite does not hold either.

Lemma 5. *ITSL $\nsubseteq$ MTSL.*

Proof. Assume $\Sigma = \{a, b\}$, and consider the language $L_{FL} = a\{a, b\}^*b \cup b\{a, b\}^*a$. This language is ITSL. Suppose L_{FL} were the intersection of n distinct TSL languages $L_1, \ldots, L_n$. Since $a\{a, b\}^*a \notin L$, there would have to be at least one L_i projecting every a on the tier, and banning aa. But then this language also incorrectly rules out aa^+b. Thus, $L \notin n$-MTSL for any number of intersecting TSL languages.

Theorem 7. *MTSL and ITSL are incomparable.*

Regarding the place of MTSL with respect to the other subregular classes, we can reuse most of the previous results. That MTSL $\nsubseteq$ LTT, PT is entailed by TSL $\nsubseteq$ LTT, PT. To see why LTT $\nsubseteq$ MTSL, consider $\Sigma = \{a, b, c\}$ and a sentential logic formula $\varphi : aa \rightarrow bb$ s.t. $L = \{w \in \Sigma^* \mid w \models \varphi\}$. Following the same reasoning as in the proof for Theorem 3, it is easy to see that this language is 2-LT (thus, LTT) but not MTSL$_n$. For PT $\nsubseteq$ MTSL, we take the same example and assume that the predicates in φ are based on *precedence* instead of *immediate precedence*. Again following the reasoning in Theorem 3, this language is PT, but not n-MTSL for any n. Finally, MTSL $\subsetneq$ SF follows trivially from the fact that every TSL language is SF [17] and that SF languages are closed under finite intersection.

Theorem 8. *MTSL is incomparable to LT and PT, and MTSL $\subsetneq$ SF.*

5.2 Intersection Closure of ITSL Languages

The definition of MTSL extends in the expected manner to ITSL.

Definition 11 *(MITSL).* *A* multiple m-input local TSL *(*(m, n)*-MITSL*$_k$*) language is the intersection of* n *distinct* m*-ITSL*$_k$ *languages (*$k, m, n \in \mathbb{N}$*).*

Since ITSL is not closed under intersection, we have ITSL $\subsetneq$ MITSL, which in turn implies MTSL $\subsetneq$ MITSL because MTSL and ITSL are incomparable. Just like TSL, MTSL, and ITSL, MITSL is incomparable to LTT and PT. That MITSL $\nsubseteq$ LTT, PT follows from their incomparability to TSL, ITSL and MTSL, which MITSL properly subsumes. For the other direction, we can simply refer to the counter-examples used in Theorem 7, which are not MITSL irrespective of the number of tiers projected by the grammar.

Theorem 9. *MITSL is incomparable to LTT and PT.*

The incomparability to LTT and PT also entails MITSL $\subsetneq$ SF (MITSL $\subseteq$ SF follows from the FO definability of ITSL and the closure of SF under intersection).

Lemma 6. *ITSL* $\subsetneq$ *MITSL* $\subsetneq$ *SF.*

This shows that MTSL, ITSL, and MITSL are all natural generalizations of TSL that preserve the relation to other language classes. This extends even to their closure properties: TSL and ITSL have exactly the same closure properties with respect to intersection, union, complement, concatenation, and relabeling, and the multi-tier variants only gain closure under intersection (the proofs for ITSL carry over with simple modifications). In addition, TSL is the natural special case of MITSL with only one tier and ISL$_1$ tier projection.

From a linguistic perspective, this means that even though TSL is inadequate in multiple respects, the insights it yields are preserved with only minor modifications. TSL is not sufficiently expressive for all phonotactic dependencies, but the move from TSL to ITSL is conceptually natural and does not affect common closure properties. TSL complexity results also do not carry over from individual processes to the whole system, but the extension of TSL to MTSL via multiple tiers is linguistically appealing and once again does not affect closure properties or the relation to other language classes. Quite simply, TSL is but one particular point in a whole region of TSL-like classes, all of which behave very similar with respect to closure properties and their relative place in the subregular hierarchy.

6 Learnability Considerations

In this paper we have explored the effects of generalizing the tier projection function for TSL languages to allow for structure-sensitivity. As long as one limits structure-sensitivity to locally bounded contexts, the shift is very natural and mathematically well-behaved. In particular, ITSL allows for additional expressivity while still excluding many unnatural patterns from the classes LT, LTT, SP, PT, and SF (Fig. 1 on page 12).

But generative capacity is not the only linguistically relevant property of language classes. Learnability is also crucial and has profound implications for natural language acquisition [18]. The extensions we have proposed in this paper do not alter the learnability of TSL in the limit from positive text. While the whole class of TSL is not learnable in this paradigm because it properly subsumes the class FIN of all finite languages, TSL_k for $k \geq 0$ is finite and thus learnable [11]. This finiteness also holds for our extensions of TSL as long as all parameters are bounded.

Theorem 10. *Given fixed k, m, and n, (n, m)-MITSL_k languages are learnable in the limit from positive text.*

This still leaves open, though, whether these languages are efficiently learnable. We expect this to be the case given the existence of efficient learners for ISL and TSL [6,21,22]. Moreover, [25] propose an efficient algorithm for MTSL_2 building on the notion of a *2-path* exploited by [21]. In a similar fashion, it should be possible to infer local contexts in the projection of tier-segments.

Conjecture 1. (n, m)-MITSL_k languages are efficiently learnable from a polynomial sample size in polynomial time.

The phonotactic phenomena studied so far suggest tight bounds on m, n, k as relevant to the class of human languages [1,15]. Typological explorations thus offer important insights into human learning abilities [8,30].

7 Conclusions

TSL languages have been proposed as a good computational hypothesis for the complexity of phonotactic patterns. However, their tier projection function is too limited because it is context agnostic. A wide range of empirical phenomena can be captured if one equips TSL with an input-strictly local projection mechanism in the sense of Chandlee [5]. The resulting new class of ITSL has the same closure properties as TSL and extends generative capacity only by a small amount. In particular, ITSL occupies a similar position to TSL in the subregular hierarchy.

This paper has explored but one point in a whole region of TSL-like language classes. For instance, we completely omitted OTSL [23] and IOTSL [15]. We also limited ourselves to comparisons to well-established classes such as LTT, ignoring more recently defined classes [13,34]. One major reason for this limit in scope is the lack of fertile characterizations of TSL and ITSL languages. Whereas suffix substitution closure makes it very easy to show that a string language is not strictly local, TSL and ITSL introduce the additional parameter of tiers and contexts that are hard to quantify over in practice. We were able to use string embeddings to create subsumption relations between the contexts and k-factors of specific strings, but this technique is not nearly as versatile as suffix substitution closure. The lack of an equally elegant characterization of TSL and its variants is a serious impediment to a full exploration of the TSL region.

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grant No. BCS-1845344.

References

1. Aksenova, A., Deshmukh, S.: Formal restrictions on multiple tiers. Proc. Soc. Comput. Linguist. (SCiL) **20**(18), 64–73 (2018)
2. Applegate, R.: Ineseno chumash grammar. Ph.D. thesis, UC Berkeley (1972)
3. Baek, H.: Computational representation of unbounded stress patterns: tiers with structural features. In: Proceedings of the 53rd Meeting of the Chicago Linguistic Society (CLS53) (2017)
4. Brzozowski, J.A., Knast, R.: The dot-depth hierarchy of star-free languages is infinite. J. Comput. Syst. Sci. **16**(1), 37–55 (1978)
5. Chandlee, J.: Strictly local phonological processes. Ph.D. thesis, University of Delaware (2014)
6. Chandlee, J., Eyraud, R., Heinz, J.: Learning strictly local subsequential functions. Trans. ACL **2**, 491–503 (2014)
7. Chandlee, J., Heinz, J.: Strict locality and phonological maps. Linguist. Inq. **49**, 23–60 (2018)
8. De Santo, A.: Commentary: developmental constraints on learning artificial grammars with fixed, flexible, and free word order. Front. Psychol. **9**, 276 (2018)
9. Eilenberg, S.: Automata, Languages, and Machines. Academic Press, Inc., Cambridge (1974)
10. Fu, J., Heinz, J., Tanner, H.G.: An algebraic characterization of strictly piecewise languages. In: Ogihara, M., Tarui, J. (eds.) TAMC 2011. LNCS, vol. 6648, pp. 252–263. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20877-5_26
11. Gold, E.M.: Language identification in the limit. Inf. Control **10**(5), 447–474 (1967)
12. Goldsmith, J.: Autosegmental phonology. Ph.D. thesis, MIT, Cambridge (1976)
13. Graf, T.: The power of locality domains in phonology. Phonology **34**(2), pp. 385–405 (2017). https://doi.org/10.1017/S0952675717000197
14. Graf, T.: Locality domains and phonological c-command over strings. In: 2017 Proceedings of NELS (2018). http://ling.auf.net/lingbuzz/004080
15. Graf, T., Mayer, C.: Sanskrit n-retroflexion is input-output tier-based strictly local. In: 2018 Proceedings of SIGMORPHON (2018)
16. Heinz, J.: The computational nature of phonological generalizations. In: Hyman, L., Plank, F. (eds.) Phonological Typology, chap. 5, pp. 126–195. Phonetics and Phonology, Mouton De Gruyter (2018)
17. Heinz, J., Rawal, C., Tanner, H.: Tier-based strictly local constraints for phonology. In: Proceedings of the ACL 49th: Human Language Technologies: Short Papers - vol. 2, pp. 58–64 (2011). http://dl.acm.org/citation.cfm?id=2002736.2002750
18. Heinz, J., Riggle, J.: Learnability. In: van Oostendorp, M., Ewen, C., Hume, B., Rice, K. (eds.) Blackwell Companion to Phonology. Wiley-Blackwell, Hoboken (2011)

19. Jäger, G., Rogers, J.: Formal language theory: refining the chomsky hierarchy. Philos. Trans. R. Soc. B: Biol. Sci. **367**(1598), 1956–1970 (2012)
20. Jardine, A.: Computationally, tone is different. Phonology (2016). http://udel.edu/~ajardine/files/jardinemscomputationallytoneisdifferent.pdf
21. Jardine, A., Heinz, J.: Learning tier-based strictly 2-local languages. Trans. ACL **4**, 87–98 (2016). https://aclweb.org/anthology/Q/Q16/Q16-1007.pdf
22. Jardine, A., McMullin, K.: Efficient learning of tier-based strictly k-local languages. In: Drewes, F., Martín-Vide, C., Truthe, B. (eds.) LATA 2017. LNCS, vol. 10168, pp. 64–76. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-53733-7_4
23. Mayer, C., Major, T.: A challenge for tier-based strict locality from Uyghur backness harmony. In: Foret, A., Kobele, G., Pogodalla, S. (eds.) FG 2018. LNCS, vol. 10950, pp. 62–83. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-57784-4_4
24. McMullin, K.: Tier-based locality in long-distance phonotactics?: learnability and typology. Ph.D. thesis, University of British Columbia, February (2016). https://doi.org/10.14288/1.0228114
25. McMullin, K., Aksënova, A., De Santo, A. (2019): Learning phonotactic restrictions on multiple tiers. Proc. SCiL **2**(1), pp. 377–378 (2019). https://doi.org/10.7275/s8ym-bx57
26. McNaughton, R., Papert, S.: Counter-Free Automata. MIT Press, Cambridge (1971)
27. Pin, J.E.: Varieties of Formal Languages. Plenum Publishing Co., New York (1986)
28. Rogers, J., et al.: On languages piecewise testable in the strict sense. In: Ebert, C., Jäger, G., Michaelis, J. (eds.) MOL 2007/2009. LNCS (LNAI), vol. 6149, pp. 255–265. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14322-9_19
29. Rogers, J., Heinz, J., Fero, M., Hurst, J., Lambert, D., Wibel, S.: Cognitive and subregular complexity. In: Morrill, G., Nederhof, M.-J. (eds.) FG 2012-2013. LNCS, vol. 8036, pp. 90–108. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39998-5_6
30. Rogers, J., Pullum, G.K.: Aural pattern recognition experiments and the subregular hierarchy. J. Logic Lang. Inf. **20**(3), 329–342 (2011)
31. Simon, I.: Piecewise testable events. In: Brakhage, H. (ed.) GI-Fachtagung 1975. LNCS, vol. 33, pp. 214–222. Springer, Heidelberg (1975). https://doi.org/10.1007/3-540-07407-4_23
32. Thomas, W.: Languages, automata, and logic. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, pp. 389–455. Springer, Heidelberg (1997). https://doi.org/10.1007/978-3-642-59126-6_7
33. Walker, R.: Yaka nasal harmony: spreading or segmental correspondence? Annu. Meet. Berkeley Linguist. Soc. **26**(1), 321–332 (2000). https://doi.org/10.3765/bls.v26i1.1164
34. Yli-Jyrä, A.: Contributions to the theory of finite-state based linguistic grammars. Ph.D. thesis, University of Helsinki (2005). http://www.ling.helsinki.fi/~aylijyra/dissertation/contribu.pdf