Tier-Based Strictly Local Stringsets: Perspectives from Model and Automata Theory

Dakotah Lambert
Stony Brook University
Department of Linguistics
dakotah.lambert@stonybrook.edu

James Rogers
Earlham College
Department of Computer Science
jrogers@cs.earlham.edu

Abstract

Defined by Heinz et al. (2011) the Tier-Based Strictly Local (TSL) class of stringsets has not previously been characterized by an abstract property that allows one to prove a stringset's membership or lack thereof. We provide here two such characterizations: a generalization of suffix substitution closure and an algorithm based on deterministic finite-state automata (DFAs). We use the former to prove closure properties of the class. Additionally, we extend the approximation and constraint-extraction algorithms of Rogers and Lambert (2019a) to account for TSL constraints, allowing for free conversion between TSL logical formulae and DFAs.

1 Tier-Based Strict Locality

The class of Strictly k-Local stringsets (SL$_k$), first described by McNaughton and Papert (1971), is well known, with learning algorithms from Garcia et al. (1990) and a decision algorithm stemming from Caron (1998) that led to constraint-extraction algorithms from Rogers and Lambert (2019a). A superclass of this, defined by the application of a Strictly k-Local grammar to the output of an erasing homomorphism (which may be the identity map) was introduced by Heinz et al. (2011) as the Tier-Based Strictly k-Local sets of strings.

In this paper, we introduce a purely relational view of TSL. From this, we derive a generalization of the abstract characterization of the Strictly k-Local stringsets for their tier-based cousins, extend the known approximation and constraint-extraction algorithms to this class, and introduce a type of alphabet-agnostic finite-state automaton, and operations thereon, useful in building representations of stringsets from logical formulae.

In demonstration of the abstract characterization of the class, we prove that TSL is not, in general, closed under any of the Boolean operations. We demonstrate in contrast that intersection closure does hold when the tier alphabets are the same. We then investigate and classify some specific linguistic examples, namely the one-stress constraint, the liquid dissimilation of Latin, and the backness harmony of Uyghur.

2 Relational Word Models

We begin by defining a relational word model in the same way as Rogers and Lambert (2019b). A relational structure in general is a set of domain elements, D, augmented with a set of relations of arbitrary arity, $R_i \subseteq D^{n_i}$. Let w be a string over some alphabet Σ. Then a *word model* for w is a structure:

$$\mathcal{M}_{\Sigma}^{R_i}(w) \triangleq \langle D_w, \sigma_w, \rtimes_w, \ltimes_w, R_i \rangle_{\sigma \in \Sigma}.$$

where D_w is isomorphic to an initial segment $\langle 0, 1, \ldots, |w| + 1 \rangle$ of the natural numbers and represents the positions in $\rtimes w \ltimes$, each σ_w (in addition to $\rtimes_w$ and $\ltimes_w$) is a unary relation that holds for all and only those positions at which σ (or $\rtimes$ or $\ltimes$, respectively) occurs, and the remaining R_i are the other salient relations, such as the standard successor or precedence relations (denoted in this paper by $\triangleleft$ and $<$, respectively). Note that the set $\{\sigma_w, \rtimes_w, \ltimes_w\}_{\sigma \in \Sigma}$ is a partition of D_w. As a minor abuse of notation, we allow symbols to refer to their associated relations, and we allow sets of relations of the same arity to be read as the disjunction of their pointwise application. Figure 1 shows three different word models for the string "abab", where each cell represents a domain element, each cell's label is the alphabetic unary relation that element satisfies, and the edges represent the indicated relation. The tier-successor relation, $\triangleleft^{\tau}$, will be defined shortly hereafter.

The class of Strictly k-Local stringsets over a tier $\tau \subseteq \Sigma$, TSL$_k^{\tau}$, was originally described as

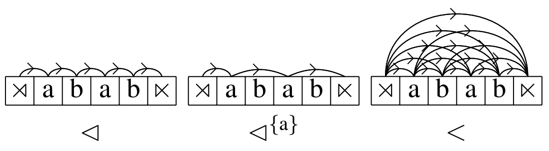

Figure 1: Three word models for the string "abab", the first variant under the standard successor relation, the second under the tier-successor relation on the alphabet $\{a\}$, and the last under the precedence relation.

those characterized by a set of Strictly k-Local constraints on the output of an erasing homomorphism (Heinz et al., 2011). Note that it can be assumed that the tier alphabet always contains $\rtimes$ and $\ltimes$. Here we suggest an alternative perspective based on relational word models and define a relation appropriate for describing this class.

The standard successor relation is the transitive reduction of the precedence relation and is first-order definable from the latter as follows:

$$x \lhd y \triangleq (x < y) \wedge \big(\forall z\big)\big[\neg(x \leq z \leq y)\big].$$

With minor modification, we can instead use the restriction of the precedence relation to the intended tier-alphabet and derive a similar relation:

$$\begin{aligned} x \lhd^{\tau} y \triangleq\ & T(x) \wedge T(y) \wedge (x < y) \\ & \wedge \big(\forall z\big)\big[\neg\big(T(z) \wedge (x \leq z \leq y)\big)\big]. \end{aligned}$$

This definition is equivalent to the standard successor relation after erasing symbols not in the intended tier alphabet, and through this equivalence we use this tier-successor relation as our basis for describing TSL stringsets and constraints. By extension, this relation should be useful in describing the yet unexplored Boolean closure of TSL$^{\tau}$ formulae, which we call Tier-Based k-Locally Testable analogously to the Locally Testable and Piecewise Testable classes characterized by McNaughton and Papert (1971) and Simon (1975), respectively. We will revisit this in section 8.

In order to avoid doubling sub- and superscripts, the tier-successor relation over tier τ is written $\lhd[\tau]$ when it appears in such a position.

3 Windows and Factors

Given a homogeneous relation R of arity a, the set

$$W_a^R(x_1) \triangleq \big\{x_1 \ldots x_a : \langle x_1, \ldots, x_a\rangle \in R\big\}$$

is the set of windows of length a (a-windows) that begin with x_1. The set of windows of length $n > a$ is defined inductively:

$$\begin{aligned} W_{i+1}^R(x_1) \triangleq \big\{ & x_1 \ldots x_{i+1} : \\ & x_1 \ldots x_i \in W_i^R(x_1) \text{ and} \\ & \langle x_{i-a+2}, \ldots, x_{i+1}\rangle \in R\big\}. \end{aligned}$$

Informally, each n-window is a sequence of positions that can be formed from a sequence of overlapping a-windows, the latter being sequences formed directly from the tuples in R. In order to discuss windows shorter than the arity of their defining relation, we say that any of the affixes of an n-window of length $m < n$ is an m-window from an appropriate starting point. Let the first position of a string x be denoted by p_0 and the final one by p_f, then define the length of x under the relation R as the size of the largest window that can be formed in x:

$$|x|^R \triangleq \max\Big\{n : (\exists v)\,\big[vp_f \in W_n^R(p_0)\big]\Big\}.$$

If R is a binary relation for which the transitive closure is asymmetric, such as the $<$ relation or its reductions used in this paper, $|x|^R$ is finite whenever x is itself finite.

Let $\hat{\Sigma} = \Sigma \cup \{\rtimes, \ltimes\}$. A string $s = \hat{\sigma}_1\hat{\sigma}_2 \ldots \hat{\sigma}_k$ for $\hat{\sigma}_i \in \hat{\Sigma}$ is a *k-factor* of a string t under the relation R, $s \sqsubseteq^R t$, iff for some position $p \in D_t$ there is some k-window $w_1 w_2 \ldots w_k \in W_k^R(p)$ such that each $\hat{\sigma}_i$ holds for the corresponding w_i. For example, one can use Figure 1 to see that for both the $\lhd^{\{a\}}$ and $<$ relations, it holds that aa $\sqsubseteq$ ⋊abab⋉, but not for $\lhd$. Additionally, abb $\sqsubseteq$ ⋊abab⋉ for $<$ but neither for $\lhd$ nor for $\lhd^{\{a\}}$.

Define the set of all k-factors of w as follows:

$$F_k^R(w) \triangleq \big\{s : |s| = k \text{ and } s \sqsubseteq^R w\big\}.$$

Additionally, define the set of factors of width at most k as one would expect:

$$F_{\leq k}^R(w) \triangleq \bigcup_{1 \leq i \leq k}\big(F_i^R(w)\big).$$

Note that a window is distinct from a factor in that the former is a sequence of positions while the latter describes a string of symbols that occupies such a sequence of positions.

Following Rogers and Lambert (2019b), we say a function $f\colon X^n \to X$ is *conservative* iff f preserves well-formedness of its inputs and it holds that for all possible inputs:

$$F_k^R\big(f(x_1, \ldots, x_n)\big) \subseteq \bigcup_{1 \leq i \leq n}\big(F_k^R(x_i)\big).$$

For strings inserting and deleting symbols other than end-markers preserves well-formedness. Note that conservativity of an operation depends on R, k, and the domain; for example, while inserting or deleting symbols not in τ is conservative under $\lhd^\tau$ (since $\lhd^\tau$ ignores them), the insertion is not conservative under $<$.

A factor f may be taken as a logical proposition that f occurs. A word model $\mathcal{M}(w)$ satisfies such a proposition, $\mathcal{M}(w) \models f$, iff $f \sqsubseteq w$. Satisfaction of a set of factors is considered disjunctively, and the Boolean connectives hold their usual meaning.

If φ is an arbitrary logical sentence using these constructions, the *models* of φ are the structures:

$$\mathrm{Mod}(\varphi) \triangleq \{\mathcal{M} : \mathcal{M} \models \varphi\},$$

and one can say that φ represents the stringset:

$$L(\varphi) \triangleq \{w : \mathcal{M}(w) \in \mathrm{Mod}(\varphi)\}.$$

Any stringset definable in this way is said to be *locally definable* under the relations in question, as an extension of the notion of locality used by McNaughton and Papert (1971). A logic further restricted to φ of the form:

$$\varphi = \bigwedge(\neg f_i)$$

where each f_i is a factor (a conjunction of negative literals) characterizes those stringsets that are *locally definable in the strict sense*.

The Strictly k-Local stringsets and their tier-based cousins are definable by a set of permitted k-factors over the appropriate relation $G \subseteq \hat{\Sigma}^{\leq k}$. We call such G a *grammar*. Since for a finite alphabet there are only finitely many k-factors, we could equivalently use the complement of G, denoted $\overline{G}$. Then the stringset is locally definable in the strict sense by taking $\varphi = \bigwedge(\neg f \in \overline{G})$.

Any stringset locally definable in the strict sense is closed under any operation conservative under the appropriate relations and factor width, because if no factor of any input is forbidden and the operation does not introduce new factors, the output cannot contain a forbidden factor.

4 Substitution of (Preprojective) Suffixes

A property is said to *characterize* a class iff all members of the class have the property and all objects that have the property are members of the class. For example, the Strictly k-Local stringsets are characterized by closure under substitution of suffixes (Rogers and Pullum, 2011). When two strings in an SL_k set share a factor of width $k-1$, the portions following this shared factor in each may be swapped to obtain new strings in the set. In order to describe an analogous property for TSL, first define the *projection* of w onto τ as follows:

$$\pi_\tau(w) \triangleq F^{\lhd[\tau]}_{|w|^{\lhd[\tau]}}(w).$$

In other words, $\pi_\tau(w)$ is the set of $\lhd^\tau$ factors in w the same length as the longest such factor. It can be shown that this is singleton and equivalent to the standard projection operation. We omit tier specifications when they are clear from context. Following mathematical tradition, we abuse notation and use $\pi_\tau(w)$ to refer to its single element.

To move freely between strings and projections, we note the following:

Lemma 1. *If a stringset L over some alphabet Σ is closed under insertion and deletion of symbols outside of some $\tau \subseteq \Sigma$, then $w \in L$ iff $\pi_\tau(w) \in L$.*

Proof. Let L be so closed. If w in L, then by closure under deletion, $\pi_\tau(w) \in L$. If $\pi_\tau(w) \in L$, then by closure under insertion, $w \in L$. □

Definition 1 (Preprojective Suffix Substitution)**.** Let Σ be an alphabet and $\tau \subseteq \Sigma$ a tier-alphabet. Let $w_1 = u_1x_1v_1$ and $w_2 = u_2x_2v_2$ be strings over Σ^* such that $\pi_\tau(x_1) = \pi_\tau(x_2)$. We then say the substrings x_1 and x_2 are *projectively shared factors* of size $k = |x_1|^{\lhd[\tau]}$ and the string $w_3 = u_1x_1v_2$ is formed by *τ-preprojective suffix substitution*.

For strings on τ^*, preprojective suffix substitution is identical to the standard suffix substitution under which SL stringsets are closed. Further, recall that insertion and deletion of symbols outside of τ is conservative, and so TSL stringsets are closed under these operations. Preprojective suffix substitution is equivalent to projecting onto τ, performing suffix substitution on the restricted domain, then doing an inverse projection by reinserting the symbols that were removed earlier. Since each step is conservative, preprojective suffix substitution is as well, so TSL stringsets are closed thereunder. More interesting is the following:

Theorem 1. *All stringsets closed both under insertion and deletion of symbols outside of some tier alphabet τ and under τ-preprojective suffix substitution for some factor size k are* TSL^τ.

Proof. Let L be a stringset so closed. Since L is closed under τ-preprojective suffix substitution, its projection to τ ($\pi_\tau(L)$) is closed under suffix substitution and is thus SL_k. Further, for any $w \in \Sigma^*$ such that $\pi_\tau(w)$ is in $\pi_\tau(L)$, Lemma 1 guarantees that w is itself in L (and vice versa). Thus by definition, L is TSL^τ_k. $\square$

Since all TSL stringsets are closed under these operations and all stringsets so closed are TSL, this combination of closures characterizes TSL.

5 Closure Properties

One constraint that is nearly universal in phonotactics is that one and only one syllable with primary stress ($\acute{\sigma}$) occurs in a given word (Hyman, 2009). Despite the fact that this constraint as a whole is neither Strictly Local nor Strictly Piecewise, it is $\mathrm{TSL}^{\{\acute{\sigma}\}}_2$, as witnessed by the following formula:

$$\neg \rtimes\ltimes \wedge \neg\acute{\sigma}\acute{\sigma}.$$

While similar formulae show that TSL^τ_{n+1} can require that n instances of arbitrary elements from τ occur, we can prove, for example, that no TSL stringset can recognize exactly the set of strings containing both 'a' and 'b'. Since TSL is closed under deletion of non-tier symbols and "ab" is in L but neither "a" nor "b" is itself in L, it is necessarily the case that both symbols would have to be on the tier alphabet for any TSL grammar that recognizes L. Using strings formed from these symbols alone, we can demonstrate failure of preprojective suffix substitution closure for TSL_3:

$$\begin{aligned} w_1 &= \rtimes\boxed{\text{aa}}\text{b}\ltimes \in \rtimes L\ltimes \\ w_2 &= \rtimes\text{b}\boxed{\text{aa}}\ltimes \in \rtimes L\ltimes \\ w_3 &= \rtimes\boxed{\text{aa}}\ltimes \notin \rtimes L\ltimes. \end{aligned}$$

In fact, by making the shared 'a' factor be of width $k-1$ rather than 2, it can be shown that no TSL_k grammar can describe exactly the set of strings containing both 'a' and 'b'. This is despite the fact that each can be required individually by a TSL_2 grammar over an appropriate tier. In other words, TSL is not closed under intersection when the tier alphabets may differ. Interestingly, the set of strings containing exactly one instance of both 'a' and 'b' is recognized by a TSL_3 grammar since its projection to the $\{\text{a}, \text{b}\}$ tier is finite and thus SL:

$$\bigwedge\{\neg\rtimes\ltimes, \neg\rtimes\text{a}\ltimes, \neg\rtimes\text{b}\ltimes, \neg\text{aa}, \neg\text{bb}, \neg\text{aba}, \neg\text{bab}\}.$$

Although TSL is not in general closed under intersection, the following holds:

Theorem 2. *If $L_1 \in \mathrm{TSL}^\tau_{k_1}$ and $L_2 \in \mathrm{TSL}^\tau_{k_2}$, then the intersection $L_1 \cap L_2 \in \mathrm{TSL}^\tau_{\max(k_1,k_2)}$.*

Proof. Let L_1 and L_2 be as stated, and further let $L = L_1 \cap L_2$ and $k = \max(k_1, k_2)$. Then L is closed under insertion and deletion of symbols outside of τ because for any $w \in L$, by definition $w \in L_1$ and $w \in L_2$, and both of these sets are so closed. L is closed under substitution of preprojective suffixes by the same reasoning. Then by Theorem 1, L is TSL^τ_k. $\square$

This theorem fails to hold for intersections of TSL stringsets over different tiers because the closure properties do not hold on both operands.

We can also show that TSL is not closed under union by demonstrating that the set of strings where all instances of 'a' precede all those of 'b' is TSL ($\neg$ba), and that where all instances of 'b' precede all those of 'a' is TSL ($\neg$ab), but their union is not:

$$\begin{array}{r} \rtimes\text{b}\boxed{\text{a}^{k-1}}\ltimes\phantom{\text{b}} \\ \rtimes\boxed{\text{a}^{k-1}}\text{b}\ltimes \\ \hline \rtimes\text{b}\boxed{\text{a}^{k-1}}\text{b}\ltimes. \end{array}$$

In general, to prove that a stringset is TSL one needs only provide the grammar. To show that a stringset cannot be TSL, one can use insertion or deletion closure to determine some symbols that must be on the tier alphabet and then use strings formed from only those symbols to demonstrate a failure of closure under substitution of preprojective suffixes. We leave as an exercise for the reader to show that TSL is not closed under complement, nor (since Σ^* is TSL) under relative complement.

6 Linguistic Examples

There are several TSL linguistic phenomena. Any Strictly k-Local pattern over an alphabet Σ can be described by a TSL^Σ_k grammar as well. Of course, this is uninteresting as we generally want to describe these phenomena with a lowest measure of complexity. The TSL class is motivated by the set of patterns that it can capture that SL does not.

One such pattern is the one-stress constraint described at the beginning of the previous section. The two sub-constraints that comprise it, namely that some syllable with primary stress occurs and that no more than one such syllable occurs, are coSL_1 (coSP_1) and $\mathrm{LTT}_{1,2}$ (SP_2), respectively, under standard adjacency and precedence accounts.

Though this constraint is neither purely SP nor purely SL, it can be described using TSL alone.

This particular kind of TSL constraint demonstrates applicability when long-distance dependencies are in effect. As another example, let us consider the simple SL_2 constraint that is alternation:

$$\bigwedge\{\neg\mathrm{ll}, \neg\mathrm{rr}\}.$$

If this constraint is applied on the tier of liquids (here only "l" and "r"), then the result is a dissimilation constraint like that of Latin as described by Cser (2010). The pattern described by Cser is a bit more involved, though: the dissimilation is blocked by non-coronal consonants. A TSL description accounts for these blockers: in addition to the liquids, the non-coronal consonants are on the tier as well.

Let us look now at an attested non-TSL pattern. In the previous section we described a method to prove that a given stringset is not TSL, and we will apply that here to the backness harmony in Uyghur as described by Mayer and Major (2018). The cited paper contains a full description of the pattern, but a simplification is as follows. Vowels and consonants both have harmonizing and transparent instances. Here we consider only the front/back vowel pair "y" and "u" and the back consonant "q". A suffix harmonizes to the rightmost harmonizing vowel if there is one, else with the rightmost harmonizing consonant if there is one, else the result is unspecified. We will ignore the final clause here.

We can prove that each of "y", "u", and "q" must be on the tier by constructing a stem of transparent segments and an affix that contains a harmonizing vowel. Inserting a harmonizing segment of mismatched backness into the stem causes an otherwise acceptable word to be rejected, and thus each of these three segment types must be on the tier. The following demonstrates a failure of closure under substitution of preprojective suffixes:

$$\frac{\begin{array}{c}\rtimes\mathrm{y}\,\boxed{\mathrm{q}^k}\,\mathrm{y}\ltimes\\ \rtimes\mathrm{u}\,\boxed{\mathrm{q}^k}\,\mathrm{u}\ltimes\end{array}}{\rtimes\mathrm{y}\,\boxed{\mathrm{q}^k}\,\mathrm{u}\ltimes}$$

It then follows that this pattern is not TSL for any choice of parameters.

In this section, we have again shown that the one-stress constraint and Latin liquid dissimilation are TSL by providing grammars with appropriate parameters, and we have used the methodology of the previous section to prove that backness harmony in Uyghur is not TSL for any parameters.

7 Multiple Relations, Additional Tiers

We proved earlier that TSL^{τ} is closed under intersection, but TSL in general is not. In this section, we discuss the intersection of TSL stringsets of incompatible relations (i.e. unequal tier alphabets). This is the complexity class inhabited by those stringsets that can be described by a coöccurrence of several TSL constraints operating over tier alphabets that are not necessarily equal.

The intersection of TSL^{τ_i} stringsets ($1 \leq i \leq n$) is locally definable in the strict sense when each forbidden factor is considered with respect to its own relation. Operationally this would be equivalent to using n distinct projective tiers, a concept explored by De Santo and Graf (2019) and referred to as MTSL. For $T = \bigcup_{1\leq i\leq n}(\tau_i)$, it is clear that insertion and deletion of symbols outside of T remains conservative. Yet T-preprojective suffix substitution no longer is; a slight modification is required in order to obtain this property:

Definition 2 (Generalized Preprojective Suffix Substitution)**.** For two strings $w_1 = u_1x_1v_1$ and $w_2 = u_2x_2v_2$ where:

$$\begin{aligned}(\forall i)\Big[\quad &(|x_1|^{\lhd[\tau_i]} \geq k-1\\ &\quad\vee \pi_{\tau_i}(x_1) = \pi_{\tau_i}(w_1))\\ \wedge\, &(|x_2|^{\lhd[\tau_i]} \geq k-1\\ &\quad\vee \pi_{\tau_i}(x_2) = \pi_{\tau_i}(w_2))\\ \wedge\, &\pi_{\tau_i}(x_1) = \pi_{\tau_i}(x_2) \qquad\Big],\end{aligned}$$

the string $w_3 = u_1x_1v_2$ is formed by the more general $\{\tau_i\}$*-preprojective suffix substitution.*

In words, on each tier, x_1 and x_2 have equal projections, which are either of length at least $k-1$ or equal to the projection of each word. This generalized operation is conservative, as the shared x substrings are guaranteed to have sufficiently many tier-symbols to allow for suffix-substitution on each projected tier. Therefore closure under this operation, and under insertion and deletion of symbols outside of the union of the tier alphabets, is necessary for a stringset to be in MTSL. Like the pumping lemma for Regular stringsets, lack of these closures can then be used to disprove class membership. It is provably not a characterization, which, like the Myhill-Nerode theorem, would allow the closures to constitute proof of membership.

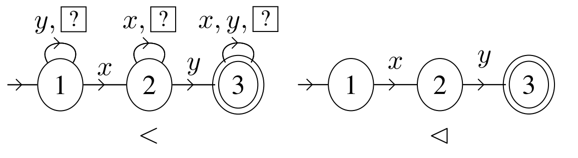

Figure 2: The factor $\rtimes xy \ltimes$ under multiple relations.

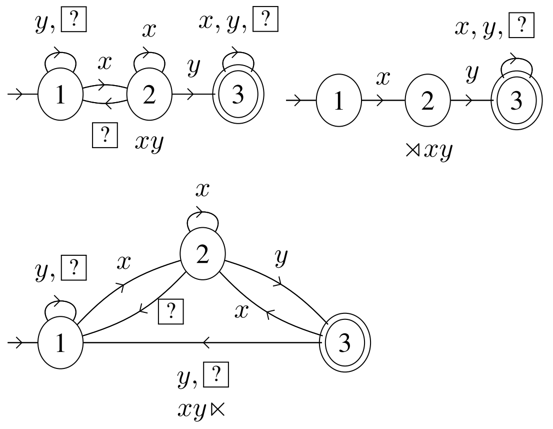

Figure 3: The factors xy, $\rtimes xy$, and $xy \ltimes$ under $\triangleleft$.

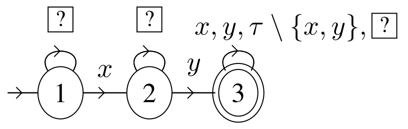

Figure 4: The factor $\rtimes xy$ under $\triangleleft^\tau$.

8 Logical Formulae and Automata

In this section, we briefly discuss the construction of finite-state automata for locally definable stringsets under each of the $\triangleleft$, $<$, and $\triangleleft^\tau$ relations (defining Local, Piecewise, and Tier-Local classes, respectively). In building automata that represent arbitrary logical formulae, one could either determine an appropriate alphabet beforehand or construct automata in such a way that only necessary symbols are considered. Here we use the latter approach, introducing a placeholder $\boxed{?}$ for potential other symbols. We define a DFA by a five-tuple $\mathcal{A} = \langle \Sigma, Q, \delta, q_\rtimes, F \rangle$ where Σ is an alphabet, Q a set of states, δ a (partial) transition function, $q_\rtimes$ an initial state, and F a set of final states.

The simplest case is the Piecewise formulae, as anchors do not affect $<$. For a string $\sigma_1 \ldots \sigma_n$ related by $<$, define $\Sigma = \{\sigma_1, \ldots, \sigma_n, \boxed{?}\}$ and construct a set of states $\{q_1, \ldots, q_{n+1}\}$ and a transition function of the form:

$$\delta(q_i, \sigma) = \begin{cases} q_{i+1} & \text{if } \sigma = \sigma_i \\ q_i & \text{otherwise.} \end{cases}$$

For $q_\rtimes = q_1$ and $F = \{q_{n+1}\}$, this reflects our intention, that the factor $\sigma_1 \ldots \sigma_n$ under $<$ occurs. Figure 2 shows the automaton constructed for the factor $\rtimes xy \ltimes$.

For factors defined using adjacency instead of precedence, we begin with fully anchored factors of the form $\rtimes \sigma_1 \ldots \sigma_n \ltimes$. The construction is the same as for Piecewise factors, except that the transition function is only defined for (q_i, σ_i). For factors that are not fully anchored, concatenate Σ^* to the side(s) missing an anchor (and determinize and minimize as appropriate). Figure 3 shows the less-anchored versions of $\rtimes xy \ltimes$.

In order to transform an adjacency factor into a tier-adjacency factor, note that the former is simply the projective image of the latter. Since the $\triangleleft^\tau$ relation does not attend to non-tier symbols, insertion of such a symbol at a given state must lead to a Nerode-equivalent state. Since the DFAs we are using here are minimal, it follows that each state should have a self-loop on all non-tier symbols. Thus we can first replace all instances of $\boxed{?}$ by parallel edges on each symbol in $\tau \setminus \{x, y\}$, and then add a self loop on $\boxed{?}$ to each state to account for symbols not on the tier. Figure 4 shows this transformation applied to the factor $\rtimes xy$.

Given these constructions for individual factors, unary operations such as the complement or iteration-closure are the standard automata-theoretic operations. For binary operations, given automata $\mathcal{A}_1$ and $\mathcal{A}_2$ whose alphabets are Σ_1 and Σ_2, add transitions on $\Sigma_2 \setminus \Sigma_1$ to $\mathcal{A}_1$ in parallel to all existing transitions on $\boxed{?}$ and similarly on $\Sigma_1 \setminus \Sigma_2$ to $\mathcal{A}_2$, then apply the standard automata-theoretic operation as usual. This use of a distinct placeholder symbol allows constraints to be defined by automata of minimal alphabet that expand in a way that preserves their semantics.

With these constructions, we can create DFAs for any stringsets definable by Boolean combinations of SL, SP, and TSL formulae, including among other things MTSL. Concatenation of automata for sequences of (Tier-)Local factors yields Piecewise-(Tier-)Local ones (Rogers and Lambert, 2019b). Boolean operations on these would yield Multi-Tier-Based Piecewise-Locally Testable stringsets: Boolean combinations of factors defined by occurrence, in order if not adjacently, of blocks of symbols on any of a number of projective tiers.

9 Deconstructing Automata

Since TSL^{τ} stringsets are closed under insertion of symbols not in τ, any transition on such a symbol from a given state must lead to a Nerode-equivalent state. Thus in a minimal DFA, such transitions are necessarily self-loops. Let $\mathcal{A} = \langle \Sigma, Q, \delta, q_{\rtimes}, F \rangle$ be a minimal DFA and define:

$$\bar{\tau} = \Big\{ \sigma : (\forall q) \left[\delta(q, \sigma) = q \right] \Big\}.$$

The projection of $\mathcal{A}$ to τ ($\pi_{\tau}(\mathcal{A})$) is the result of replacing all transitions on symbols from $\bar{\tau}$ by transitions on ε, and since these transitions are all self-loops, this is equivalent to simply removing them. Then $\mathcal{A}$ represents a TSL_k^{τ} stringset iff this projection represents an SL_k one. The algorithms of Rogers and Lambert (2019a) can then be used to extract SL constraints from the projection, which of course are the TSL^{τ} constraints of $\mathcal{A}$ itself. Use of these algorithms provides a simple way to test whether an arbitrary Regular stringset is TSL_k^{τ}, and if so, for which parameters k and τ and even which grammar.

On the other hand, if $L(\mathcal{A})$ was not TSL, then since the extracted SL constraints describe the smallest SL superset of $L\big(\pi_{\tau}(\mathcal{A})\big)$, it follows that they then also describe the smallest TSL^{τ} superset of $L(\mathcal{A})$. That said, there may be smaller TSL supersets over different tiers.

10 Conclusions

The Tier-Based Strictly k-Local (TSL_k) class of stringsets was introduced by Heinz et al. (2011) and the question of what an abstract characterization for the class might be has remained open until now. We introduced here an abstract characterization, which can be used to provably state whether or not a given stringset is in the class. We then used this to prove various closure properties of the class itself. As TSL is not closed under intersection (but TSL^{τ} for fixed τ is), we discussed its intersection closure (MTSL) and provided a property that is necessary to be in MTSL. Failure to satisfy this property thus proves that a stringset is not in this class.

Further, to better integrate the TSL class with the other Piecewise-Local classes on the Subregular hierarchy, we introduced a tier-successor relation and associated logical formulae. We then described a method to construct deterministic finite-state automata from such formulae in order to harness the plentiful library of existing automata-theoretic tools. Finally, we used our abstract characterization to demonstrate a method of factoring a TSL automaton into individual constraints and a method of finding the constraints that produce the smallest TSL superset of a given non-TSL automaton. This provides a means to determine whether an arbitrary regular stringset is TSL_k^{τ}, and if so, for which parameters.

11 Future Work

We would like to explore linguistic applications of the Tier-Based extensions to the other classes in the piecewise-local subregular hierarchy, such as the Tier-Based Locally Testable stringsets hinted at in section 2 or the arbitrary formulae from section 8. For example, it would appear that Uyghur backness harmony might be MTLT, where the existence of harmonizing vowels can turn off the constraint referencing consonants.

Acknowledgments

The authors would like to thank Jeffrey Heinz at Stony Brook University for his contributions to the study of piecewise-local subregularity. We would also like to thank the anonymous reviewers for their detailed and useful feedback.

References

Pascal Caron. 1998. LANGAGE: A Maple package for automaton characterization of regular languages. In Derick Wood and Sheng Yu, editors, *Automata Implementation*, volume 1436 of *Lecture Notes in Computer Science*, pages 46–55. Springer Berlin / Heidelberg.

András Cser. 2010. The -alis/-aris allomorphy revisited. In Franz Rainer, Wolfgang Dressler, Dieter Kastovsky, and Hans Christian Luschützky, editors, *Variation and Change in Morphology: Selected Papers from the 13th International Morphology Meeting*, pages 33–52. John Benjamins Publishing Company, Vienna, Austria.

Aniello De Santo and Thomas Graf. 2019. Structure sensitive tier projection: Applications and formal properties. In Raffaella Bernardi, Greg Kobele, and Sylvain Pogodalla, editors, *Formal Grammar 2019*, volume 11668 of *Lecture Notes in Computer Science*, pages 35–50. Springer Verlag.

Pedro Garcia, Enrique Vidal, and José Oncina. 1990. Learning locally testable languages in the strict sense. In *Proceedings of the 1st International Workshop on Algorithmic Learning Theory*, pages 325–338, Tokyo, Japan.

Jeffrey Heinz, Chetan Rawal, and Herbert G. Tanner. 2011. Tier-based strictly local constraints for phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Short Papers*, volume 2, pages 58–64, Portland, Oregon. Association for Computational Linguistics.

Larry M. Hyman. 2009. How (not) to do phonological typology: The case of pitch-accent. *Language Sciences*, 31(2–3):213–238.

Connor Mayer and Travis Major. 2018. A challenge for tier-based strict locality from Uyghur backness harmony. In Annie Foret, Greg Kobele, and Sylvain Pogodalla, editors, *Formal Grammar 2018*, volume 10950 of *Lecture Notes in Computer Science*, pages 62–83.

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

James Rogers and Dakotah Lambert. 2019a. Extracting Subregular constraints from Regular stringsets. *Journal of Language Modelling*, 7(2):143–176.

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

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.

Imre Simon. 1975. Piecewise testable events. In Helmut Brakhage, editor, *Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 214–222. Springer Verlag, Berlin.