Proceedings of Machine Learning Research 153:66–80, 2021 Proceedings of the 15th ICGI

Learning Multiple Independent Tier-based Processes

Phillip Burness PBURN036@UOTTAWA.CA
Kevin McMullin KEVIN.MCMULLIN@UOTTAWA.CA
University of Ottawa

Editors: Jane Chandlee, Rémi Eyraud, Jeffrey Heinz, Adam Jardine, and Menno van Zaanen

Abstract

Work on the learnability of tier-based string-to-string functions has so far been limited to those that operate over a single tier. Such functions are, however, generally incapable of modelling multiple simultaneous processes. It is relatively easy to define a class of multi-tiered functions that can handle more than one process at a time, but doing so has negative consequences for learnability. Namely, we conjecture that it is difficult if not impossible to efficiently learn any arbitrary set of tiers from positive data alone. We thus describe the *strongly target-specified* subclass of multi-tiered functions, whose tiers can be efficiently identified from positive data. In these functions, each input element is associated with a single tier that on its own can fully determine what the element is mapped to. When the tiers act independently in this way, we can learn them in isolation from each other. A transducer representation of the target function can then be constructed using the discovered tiers.

Keywords: long-distance phonology; subregular functions; locality; tiers

1. Introduction

Much recent work in computational phonology has focused on representing phonological patterns using hierarchies of *subregular* formal languages and functions. At the bottom of the language hierarchy are the Strictly Local (SL) languages, which model local phonotactic restrictions by disallowing specific contiguous strings of symbols (Rogers and Pullum, 2011; Rogers et al., 2013). These SL languages form the basis of the Input Strictly Local (ISL) and Output Strictly Local (OSL) functions, which model processes with local triggering contexts (Chandlee, 2014; Chandlee et al., 2014, 2015). The SL languages cannot, however, model non-local phonotactics, nor can the ISL or OSL functions model non-local processes. To address this limitation, a class of languages known as the Tier-based Strictly Local (TSL) languages were developed, which operate like SL languages except over a tier projection, allowing them to enforce non-local restrictions (Heinz et al., 2011; McMullin and Hansson, 2016). These TSL languages were in turn given functional extensions that can compute processes with non-local contexts (Burness and McMullin, 2019; Hao and Andersson, 2019; Hao and Bowers, 2019). An important result upon which this paper will expand comes from Burness and McMullin (2019). They showed that any TSL function can be efficiently learned from positive data when the tier is known *a priori*, and furthermore showed that the tier itself can be efficiently learned from positive data for total TSL functions operating over a window of size 2.

Despite the successes of the TSL languages and functions, they suffer from a major drawback. Specifically, the standard definitions of TSL languages and functions allow one and only one tier. Consequently, standard TSL languages and functions are ill-equipped to deal with multiple, simultaneous long-distance dependencies. Recent work by Aksënova and Deskmukh (2018) and McMullin et al. (2019) has looked at the properties and learnability of TSL languages operating over multiple tiers. This paper will show that the tiers of a multi-tiered function are efficiently learnable from positive data provided that all tiers operate over a window of size 2, and provided that the output corresponding to a given input element is determined by exactly one of the multiple tiers.

The rest of this paper is structured as follows. Section 2 introduces the notation that will be used throughout the paper. Section 3 presents the TSL functions as they are currently defined. Section 4 presents the Muliple Tiered-based Strictly Local (MTSL) functions, as well as the subclass thereof that our algorithm is designed to learn. Section 5 discusses the properties of the target MTSL subclass that allow for efficient learning. Section 6 presents our algorithm for learning multiple tiers and demonstrates that it is efficient in terms of its run time and the amount of data it needs to succeed. Finally, Section 7 concludes and provides directions for future research.

2. Preliminaries

To start, let Σ be an alphabet of symbols. A string w is a finite contiguous sequence of symbols from Σ, and $|w|$ denotes the length of w. We write λ for the unique "empty" string of length 0. We use Σ^* to denote the set of all strings of any length that can be made from elements in Σ, including the empty string. Given two strings u and v, we write $u \cdot v$ to denote their concatenation, though will often simply write uv when context permits. A prefix of some string $w \in \Sigma^*$ is any string p such that $w = p \cdot x$ and $x, p \in \Sigma^*$. A suffix of some string $w \in \Sigma^*$ is any string s such that $w = x \cdot s$ and $x, s \in \Sigma^*$. When $|w| \geq n$, $\texttt{suff}^n(w)$ denotes the unique suffix of w with a length of n; when $|w| < n$, it simply denotes w itself. We write $\texttt{prefixes}(w)$ to refer to the set of all prefixes in w. Given a string w and one of its prefixes u we write $u^{-1} \cdot w$ to denote w with u removed from its front. For example, $(ab)^{-1} \cdot abcde = cde$. Finally, given a set of strings S, we write $\texttt{lcp}(S)$ to denote the *longest common prefix* of S, which is the string u such that u is a prefix of every $w \in S$, and there exists no other string v such that $|v| \geq |u|$ and v is also a prefix of every $w \in S$.

A string-to-string function is a relation that pairs every $w \in \Sigma^*$ with one $y \in \Delta^*$, where Σ and Δ are the input alphabet and output alphabet respectively. Given a set of input strings $I \subseteq \Sigma^*$ and a string-to-string total function f, $f(I) = \bigcup_{i \in I}\{f(i)\}$ is the set of all outputs associated to at least one of the inputs. An important concept is that of the *tails* of an input string w with respect to a function f. In words, $\texttt{tails}_f(w)$ pairs every possible string $y \in \Sigma^*$ with the portion of $f(wy)$ that is *directly attributable* to y. That is, the tails of w are its effect on the output of any subsequent string of input symbols. We say that two strings w_1 and w_2 are *tail-equivalent* with respect to f when $\texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$. Note that tail equivalency acts as a partition on Σ^*.

Definition 1 Tails (Oncina and Garcia, 1991)
Given a string-to-string function f and an input $w \in \Sigma^*$:

$$\texttt{tails}_f(w) = \{(y, v) \mid f(wy) = uv \ \wedge \ u = \texttt{lcp}(f(w\Sigma^*))\}$$

Definition 2 Tail equivalency
Given a string-to-string function f, two string $w_1, w_2 \in \Sigma^*$ are tail-equivalent *with respect to f if and only if $\texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$*

Throughout the rest of this paper, we will need to be able to pick out the portion of the output that corresponds to actual input material. Viewed from another perspective, we need to be able to ignore the portion of the output that would correspond to a word-end symbol. To make this distinction, Chandlee et al. (2015) defined the prefix function f^p associated with a function f as shown below. An example where $f(w)$ and $f^p(w)$ differ would be a function that appends a to the end of every input string. In this case, f^p is simply the identity map, so $f^p(abc) = abc$ whereas $f(abc) = abca$.

Definition 3 Prefix function (Chandlee et al., 2015)
Given a function f, its associated prefix function f^p is such that $f^p(w) = \texttt{lcp}(f(w\Sigma^*))$.

3. SL and TSL functions

Like their name suggests, the Input Strictly Local (ISL) and Output Strictly Local (OSL) functions take the Strictly Local (SL) languages as their base. The property of the SL languages that allowed for the jump to ISL and OSL functions is known as Suffix Substitution Closure (SSC; Rogers and Pullum, 2011; Rogers et al., 2013). Informally, if two grammatical strings share a middle portion of at least length $k-1$ (i.e., if $w_1 = axb$, $w_2 = cxd$, and $|x| \geq k-1|$), then we can substitute the suffixes that come after the overlap without causing ungrammaticality. A corollary of SSC is that any two grammatical strings from an SL$_k$ language that end in the same $k-1$ (or more) symbols can be legally continued by the exact same set of strings (Chandlee et al., 2015). The definitions of the ISL and OSL functions take this corollary and adapt it so that it applies to functional tails. Informally, a function f is ISL if the tail-equivalence classes of f correspond to input suffixes, and a function f is OSL if the tail-equivalence classes of f correspond to suffixes of f^p. In the interest of space, we henceforth focus entirely on output-oriented functions. The learning results below also hold for input-oriented functions with minimal changes.

Definition 4 Output Strictly k-Local Functions (Chandlee et al., 2015)
A function f is OSL$_k$ if for all pairs w_1, w_2 in Σ^*:

$$\texttt{suff}^{k-1}(f^p(w_1)) = \texttt{suff}^{k-1}(f^p(w_2)) \implies \texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$$

Chandlee (2014) and Chandlee et al. (2014, 2015) show that most iterative phonological processes can be modelled with an OSL function, an important exception being long-distance iterative processes like consonant harmony. This is parallel to the fact that long-distance phonotactics cannot be represented with an SL stringset, which motivated Heinz et al. (2011) to define the Tier-based Strictly Local (TSL) languages—stringsets that are SL after an erasure function has applied, masking all symbols that are irrelevant to the restrictions that the language places on its strings (i.e., all symbols that do not belong to the specified tier). Note that the erasure function is sometimes called the *projection* function.

Definition 5 Erasure function
Given a tier $T \subseteq \Sigma$, the erasure function *applied by T on Σ^* is such that:*

$$\begin{array}{lcll} \mathtt{erase}_T(\lambda) & = & \lambda & \\ \mathtt{erase}_T(w) & = & \mathtt{erase}_T(u) \cdot \sigma & \text{if} \quad w = u \cdot \sigma \wedge \sigma \in T \\ \mathtt{erase}_T(w) & = & \mathtt{erase}_T(u) & \text{if} \quad w = u \cdot \sigma \wedge \sigma \notin T \end{array}$$

As it turns out, the TSL languages also exhibit a form of Suffix Substitution Closure (Lambert and Rogers, 2020). In light of this fact, the legal continuations of any string w in a TSL$_k$ language can be inferred simply by looking at the $k-1$ suffix of $\mathtt{erase}_T(w)$. Just as we did for the SL functions, then, we can define the TSL functions according to how they partition Σ^* into tail-equivalence classes. Informally, a function f is TSL if the tail-equivalence classes of f correspond to tier suffixes (where the tier is a subset of the relevant alphabet Δ). For convenience, we will write $\mathtt{suff}^n_T(w)$ to mean $\mathtt{suff}^n(\mathtt{erase}_T(w))$ in what follows.

Definition 6 Output Tier-based Strictly k-Local Functions (Burness and McMullin, 2019)
A function f is OTSL$_k$ if there is a tier $T \subseteq \Delta$ such that for all w_1, w_2 in Σ^*:

$$\mathtt{suff}^{k-1}_T(f^p(w_1)) = \mathtt{suff}^{k-1}_T(f^p(w_2)) \implies \mathtt{tails}_f(w_1) = \mathtt{tails}_f(w_2)$$

4. Multi-tiered Functions

As discussed in Burness and McMullin (2019), the TSL functions are quite versatile, being able to model long-distance harmony and long-distance dissimilation, both with and without blocking effects. This is, however, only the case when we model each phonological process of a language in isolation. Consider the Kikongo language, which contains a process of progressive height harmony affecting vowels and a process of progressive nasal harmony affecting /d/ (Aksënova and Deskmukh, 2018; Ao, 1991; Hyman, 1998). The processes apply independently, but may potentially co-occur, as seen in the behaviour of the perfective active suffix /-idi/. When the suffix attaches to a base that contains neither a [−high] vowel nor a [+nasal] stop, it surfaces faithfully as in /suk-idi/ → [suk-idi] 'wash-PERF.ACT'. When the suffix attaches to a base that contains a [+nasal] stop but does not contain a [−high] vowel, it undergoes nasal harmony but not height harmony as in /nik-idi/ → [nik-ini] 'ground-PERF.ACT'. Finally, when the suffix attaches to a base containing both a [−high] vowel and a [+nasal] consonant, it undergoes both harmonies simultaneously as in /meng-idi/ → [meng-ene] 'hate-PERF.ACT'.

The combination of height harmony and nasal harmony cannot be computed by a single TSL function. To see why, consider what happens when we try with an OTSL$_2$ function whose tier consists of vowels and nasal consonants. Producing a vowel will push the most recent nasal consonant (if any) out of the $k-1$ window, and producing a nasal consonant will push the most recent vowel (if any) out of the $k-1$ window. At any given point, then, we can only know how to correctly map an input vowel or only know how to correctly map an input /d/. Increasing k does not eliminate the issue, since any number of vowels can in principle occur between two nasal consonants, and any number of nasal consonants can in principle occur between two vowels. There is no reason, however, why we cannot let a single function operate over multiple tiers. Doing so very easily allows us to model the separate height harmony and nasal harmony at the same time. Burness and McMullin (2020) defined such Multi-Tiered Strictly k-Local (MTSL$_k$) functions as follows.

Definition 7 Output Multi-Tiered Strictly k-Local functions
A function f is OMTSL$_k$ if there is a finite set Θ of tiers $T \subseteq \Delta$ such that for all $w_1, w_2 \in \Sigma^*$:

$$[\bigwedge_{T\in\Theta}[\mathtt{suff}_T^{k-1}(f^p(w_1)) = \mathtt{suff}_T^{k-1}(f^p(w_2))]] \Rightarrow [\mathtt{tails}_f(w_1) = \mathtt{tails}_f(w_2)]$$

In words, there is a finite set of tiers such that if two input strings share a $k-1$ suffix *on all tiers*, then they will have the same tails. For reasons of space we do not provide an automata-theoretic characterization of this function class, although it is worth a brief discussion. Chandlee et al. (2015) showed that the OSL$_k$ functions exactly correspond to finite-state transducers wherein the current state acts as a record of the last up to $k-1$ symbols written to the output string. Burness and McMullin (2019) showed that to generalize this result to the OTSL$_k$ functions, we need only ensure that the current state acts as a record of the last up to $k-1$ *tier* symbols written to the output string. Generalizing to the OMTSL$_k$ functions requires, as one might expect, that the current state acts as a record of the last up to $k-1$ symbols *on each tier*. Where each state label is a single raw string in the OSL case and a single tier string in the OTSL case, each state label in an OMTSL transducer would be a collection of tier strings, one for each of the given tiers. A reviewer also points out that tier projection is reminiscent of indexed grammars and asks whether the tier-projection mechanism could be better reflected by more sophisticated automata. One possibility would be to use a registered automaton (Cohen-Sygal and Wintner, 2006) with one register per tier. Such a transducer would write to these registers throughout the course of a derivation, updating them to reflect the suffix on the relevant tier, essentially constructing the state space of the naive transducer on the fly.

The above definition of OMTSL$_k$ functions allows any finite number of tiers and allows any conceivable relationship between their contents. We conjecture that this lack of restrictions renders efficient tier learning incredibly difficult, if not outright impossible. In order to restrict the hypothesis space in a manner conducive to efficient tier learning, we start by referencing the contribution of each $\sigma \in \Sigma$ separately, rather than referencing the entirety of $\mathtt{tails}_f$. Informally, the contribution of σ relative to w is the portion of $f(w\sigma)$ uniquely and directly attributable to a (e.g., it is what we would append to the output upon reading a in a transducer after having read w). We also consider the contribution of a special word-end symbol $\ltimes$ to handle functions where $f^p(w) \neq f(w)$.

Definition 8 Contribution
Given a string-to-string function f, and some $w \in \Sigma^*$:

- *for $\sigma \in \Sigma$, $\mathtt{cont}_f(\sigma, w) = \mathtt{lcp}(f(w\Sigma^*))^{-1} \cdot \mathtt{lcp}(f(w\sigma\Sigma^*)) = f^p(w)^{-1} \cdot f^p(w\sigma)$*
- *for $\ltimes \notin \Sigma$, $\mathtt{cont}_f(\ltimes, w) = \mathtt{lcp}(f(w\Sigma^*))^{-1} \cdot f(w) = f^p(w)^{-1} \cdot f(w)$*

Using this notion of single-element contributions, we can pick out a non-trivial subset of the OMTSL$_k$ functions that are efficiently learnable from positive data. For the functions in this subset, the outcome of a given singular input element depends on exactly one tier. Viewed from another perspective, each input element (i.e., each potential target of one or more processes) specifies the tier that it wants to track, and the contribution of an input element can always be determined by tracking only its specified tier; tracking other tiers provides information that is either redundant or irrelevant. Burness and McMullin (2020) singled out a similar but more permissive subset of the MTSL functions by defining a restriction that they call *target specification.* To fulfill their restriction, it must always be possible to link the behaviour of a given input element back to a fixed superset-subset hierarchy of tiers (in other words, each target specifies a fixed hierarchy to which it pays exclusive attention). The functions we define here are a special case of Burness and McMullin's (2020) restriction, since the single tier responsible for an input element's behaviour vacuously forms a superset-subset hierarchy. Accordingly, we call the functions to be learned by our algorithm below the *strongly* target-specified functions.

Definition 9 Strong target specification
An OMTSL$_k$ function f is strongly target specified *if for each $x \in \Sigma \cup \{\ltimes\}$ there is a tier $T \subseteq \Delta$ such that for all $w_1, w_2 \in \Sigma^*$:*

$$[\mathtt{suff}_T^{k-1}(f^p(w_1)) = \mathtt{suff}_T^{k-1}(f^p(w_2))] \Rightarrow [\mathtt{cont}_f(x, w_1) = \mathtt{cont}_f(x, w_2)]$$

The Kikongo data presented above can be analyzed using a strongly target-specified OMTSL$_2$ function. The input element /i/ is associated with (or specifies that it wishes to track) the tier T_i which contains all and only the vowels of the language. The input element /d/ is associated (or specifies that it wishes to track) the tier T_d which contains all and only the nasal consonants. The simultaneous computation of the two rules is shown pictorially in Figure 1 for the word /meng-idi/ → [meng-ene] 'hate-PERF.ACT'. The string in the center shows how each input element gets transformed by the function. Solid lines represent projection to an output tier and dashed lines represent an output tier element's influence on an input element.

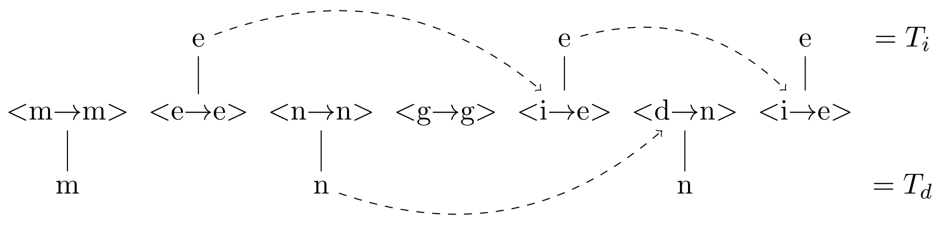

Figure 1: Simultaneous height harmony and nasal harmony in Kikongo.

5. Previous Learning Results

The tier-induction strategy of Burness and McMullin (2019) relied on some important properties of OTSL$_2$ functions. We outline these below and show how they transfer over to the strongly target-specified OMTSL$_2$ functions in such a way that the learning algorithm of Burness and McMullin (2019) can be extended to our particular multi-tiered case.

First, we note that many OTSL$_2$ functions can be described using a variety of tiers. For example, the identity map can be described using any subset of the output alphabet as the tier. Burness and McMullin (2019) showed that, given an OTSL$_2$ function, taking the union of two potential tiers will result in another potential tier (i.e., potential tiers can be freely combined). This property holds, with appropriate modifications, for the strongly target-specified OMTSL$_2$ functions as well. Let an x-tier be any tier that can uniquely determine the contribution of $x \in \Sigma \cup \{\ltimes\}$. The following lemma shows that, given a strongly target-specified OMTSL$_2$ function, combining any of its potential x-tiers will result in another potential x-tier.

Lemma 10 Free combination of tiers
Given a strongly target-specified OMTSL$_2$ function f, if $A \subseteq \Delta$ and $B \subseteq \Delta$ are both x-tiers for f, then $\Omega = A \cup B$ is also an x-tier for f.

Proof If $\texttt{suff}^1_\Omega(f^p(w_1)) = \texttt{suff}^1_\Omega(f^p(w_2)) = t$, then $t \in A$ or $t \in B$. If $t \in A$, then $\texttt{suff}^1_A(f^p(w_1)) = \texttt{suff}^1_A(f^p(w_2))$ and therefore $\texttt{cont}_f(x, w_1) = \texttt{cont}_f(x, w_2)$ since A is an x-tier for f. If $t \in B$, then $\texttt{suff}^1_B(f^p(w_1)) = \texttt{suff}^1_B(f^p(w_2))$ and therefore $\texttt{cont}_f(x, w_1) = \texttt{cont}_f(x, w_2)$ since B is an x-tier for f. Therefore, $\texttt{suff}^1_\Omega(f^p(w_1)) = \texttt{suff}^1_\Omega(f^p(w_2)) \Rightarrow \texttt{cont}_f(x, w_1) = \texttt{cont}_f(x, w_2)$, which means that Ω is an x-tier for f. ■

Burness and McMullin (2019) demonstrated that the OTSL$_2$ version of Lemma 10 (their Lemma 4) implies the existence of a unique largest tier for any OTSL$_2$ function that is a superset of its other possible tiers (if any others exist). Similarly, we point out that Lemma 10 implies that there exists a unique largest x-tier for any strongly target-specified OMTSL$_2$ function. Parallel to Burness and McMullin (2019), we call this the *canonical* x-tier for f.

Definition 11 Canonical x-tier
Given a strongly target-specified OMTSL$_2$ function f, the x-tier $T \subseteq \Delta$ is the canonical *x-tier for f if and only if there is no other x-tier $\Omega \subseteq \Delta$ for f such that $|\Omega| \geq |T|$.*

Burness and McMullin (2019) go on to show that, if one attempts to describe an OTSL$_2$ function using a superset of its canonical tier, then there will always be at least one input-output pair which acts as evidence that one of the superfluous tier elements cannot be a member of *any* tier for f. An analogous statement holds true for strongly target-specified OMTSL$_2$ functions, as follows.

Lemma 12 Absolute non-tier status
Let f be a strongly target-specified OMTSL$_2$ function where $T \subseteq \Delta$ is the canonical x-tier for $x \in \Sigma \cup \{\ltimes\}$. For every Ω such that $T \subset \Omega$ there will exist $a \in (\Omega - T)$ and $w_1, w_2 \in \Sigma^*$ such that $\texttt{suff}^1_\Omega(f^p(w_1)) = \texttt{suff}^1_\Omega(f^p(w_2)) = a$ and $\texttt{cont}_f(x, w_1) \neq \texttt{cont}_f(x, w_2)$.

Proof By contradiction. Suppose that the lemma is false. This means that for all symbols $a \in (\Omega - T)$ and for all pairs of words $w_1, w_2 \in \Sigma^*$ it is the case that $\texttt{suff}^1_\Omega(f(w_1)) = \texttt{suff}^1_\Omega(f(w_2)) = a$ implies that $\texttt{cont}_f(x, w_1) = \texttt{cont}_f(x, w_2)$. Now, since T is an x-tier for f, it is also the case that for all symbols $b \in T$ and for all pairs of words $w_1, w_2 \in \Sigma^*$ it is the case that $\texttt{suff}^1_\Omega(f(w_1)) = \texttt{suff}^1_\Omega(f(w_2)) = b$ implies that $\texttt{cont}_f(x, w_1) = \texttt{cont}_f(x, w_2)$.

Together these imply that for all symbols $c \in \Omega$ and for all pairs of words $w_1, w_2 \in \Sigma^*$ it is the case that $\texttt{suff}^1_\Omega(f(w_1)) = \texttt{suff}^1_\Omega(f(w_2)) = c$ implies that $\texttt{cont}_f(x, w_1) = \texttt{cont}_f(x, w_2)$. This means that Ω is an x-tier for f, but $|\Omega| > |T|$, contradicting the initial premise that T is the canonical x-tier for f. ■

Putting the above results together implies a method guaranteed to find the canonical x-tier for any strongly target-specified OMTSL$_2$ function and one of its input elements x. We begin by hypothesizing that the canonical tier is equal to the entire output alphabet Δ and look through our sample for evidence that some element cannot be on the tier. If such an element is found, we remove it from the hypothesized tier, and continue verifying the tier contents until we reach a point where none of the elements can be flagged for removal. At this point, we will have found the canonical tier we were looking for. This is precisely the strategy used by Burness and McMullin (2019) use to find the single canonical tier of a OTSL$_2$ function. The major difference in the multiple tier case is that we must implement the strategy once per input element.

6. Learning Multiple Tiers

This section presents a learning algorithm that identifies the canonical x-tiers of any total strongly target-specified OMTSL$_2$ function. Once the tiers are known, they can be used along with a training sample to construct a transducer that computes the target function, a step which we sketch further below. Our criterion for successful learning is exact identification from positive data (Gold, 1967) with polynomial bounds on time and data (de la Higuera, 1997). Before we can work towards identifying tiers, we must gain as much information as possible about the prefix function f^p corresponding to f, based only on the evidence provided in the training sample. To do this, the `estimate_fp` procedure of Burness and McMullin (2019), shown in Algorithm 1, goes through every string `pref` that is the prefix of at least one input string in the training data, and for every $\sigma \in \Sigma$, it checks whether $\texttt{pref} \cdot \sigma$ is also a prefix of some input string. If this is the case, there is enough information to determine $f^p(\texttt{pref})$, and the algorithm does so.

```
Function estimate_fp($S$):
    $P \leftarrow \emptyset$
    $X \leftarrow \{\texttt{pref} \mid \texttt{pref} \in \texttt{prefixes}(w)$ where $(w, u) \in S\}$
    $Y \leftarrow \{\texttt{pref} \in X \mid (\forall \sigma \in \Sigma)[\texttt{pref} \cdot \sigma \in X]\}$
    for each $y \in Y$ do
        $z \leftarrow \texttt{lcp}(\{u \mid (w, u) \in S$ where $y \in \texttt{prefixes}(w)\})$
        $P \leftarrow P \cup \{(y, z)\}$
    end
    return $P$
```

Algorithm 1: Prefix function estimation (Burness and McMullin, 2019)

Now we can determine the x-tiers independently of each other. This is accomplished by the function `get_tiers`, a modified version of the tier induction procedure in Burness and McMullin (2019). When determining T_x, it looks through the set P constructed by `estimate_fp` for any evidence that some $a \in T_x$ needs to be removed. To do this, it builds an auxiliary set `match` that contains every $(p, f^p(p)) \in P$ for which $\texttt{suff}^1_{T_x}(f^p(p)) = a$ under the current hypothesis for T_x. It then checks whether $\texttt{cont}_f(x, p)$ is the same for all $(p, f^p(p)) \in \texttt{match}$. If this is the case, a is added to the set `keep`. However, if there is more than one value found for the contribution of some $x \in \Sigma \cup \{\ltimes\}$, it will instead remove a from T_x. If at any point some symbol gets removed from a tier, the set `keep` is immediately emptied. The algorithm will continue working on T_x until every a in the current hypothesis for T_x gets added to the set `keep`. The paragraphs below establish that the `get_tiers` function identifies the canonical tiers of any total strongly target-specified OMTSL$_2$ function in polynomial time and data.

```
Function get_tiers(S):
    $P \leftarrow$ estimate_fp($S$)
    tiers $\leftarrow \emptyset$
    for each $x \in \Sigma \cup \{\ltimes\}$ do
        $T_x \leftarrow \Delta$
        keep $\leftarrow \emptyset$
        while keep $\neq T_x$ do
            for each $a \in T_x$ do
                match $\leftarrow \{(p, q) \in P \mid \texttt{suff}^1_{T_x}(q) = a\}$
                if $x \in \Sigma$ then
                    $C \leftarrow \{q^{-1} \cdot y \mid (p, q) \in \texttt{match} \wedge (px, y) \in P\}$
                else
                    $C \leftarrow \{q^{-1} \cdot y \mid (p, q) \in \texttt{match} \wedge (p, y) \in S\}$
                end
                if $|C| > 1$ then
                    $T_x \leftarrow T_x - \{a\}$
                    keep $\leftarrow \emptyset$
                end
                if $a \in T_x$ then
                    keep $\leftarrow$ keep $\cup \{a\}$
                end
            end
        end
        tiers $\leftarrow$ tiers $\cup \{T_x\}$
    end
    return tiers
```
Algorithm 2: Tier induction

Lemma 13 Polynomial time
For any input sample S, `get_tiers`(S) runs in time polynomial in the size of S.

Proof We begin by calling `estimate_fp`. This step is unchanged from Burness and McMullin (2019), who showed that its computation time is quartic in the size of the sample. We now run the portion of Burness and McMullin's (2019) algorithm that determines the tiers, to which we've made two changes. The first change is that some steps from within the "while" loop were removed. The second change is that the scope of "for each $x \in \Sigma \cup \{\ltimes\}$"

was enlarged to encompass the "while" loop, rather than being encompassed by it. These changes do not affect the time complexity of the procedure, which Burness and McMullin (2019) showed was quintic in the size of the sample. ■

The remaining lemmas of this section will show that for each total strongly target-specified OMTSL$_2$ function f, there is a finite kernel of data consistent with f that is a characteristic set for the algorithm (i.e., if the training set subsumes this kernel, the algorithm is guaranteed to succeed). The OMTSL$_k$ functions divide Σ^* into a finite number of equivalence classes according to sets of tails, meaning that the OMTSL$_k$ functions are also subsequential functions. Oncina and Garcia (1991) show how the finite partition of Σ^* lets us build the smallest finite-state transducer that computes a given subsequential function. Given a state q in this canonical transducer $\mathcal{M}$, we write w_q to denote the length-lexicographically earliest input string that reaches the state q.

Definition 14 Characteristic set
A sample S contains a characteristic set if it contains the following for each state q in $\mathcal{M}$:

1. *The input-output pair $(w_q, f(w_q))$.*
2. *For all triples $a, b, c \in \Sigma$:*
   i. *some input-output pair $(w_q a, f(w_q a))$,*
   ii. *some input-output pair $(w_q ab, f(w_q ab))$, and*
   iii. *some input-output pair $(w_q abcv, f(w_q abcv))$, where $v \in \Sigma^*$*

Lemma 15 Evidence availability
If a learning sample S contains a characteristic set, then `estimate_fp`(S) *provides sufficient information to determine the following for all $w \in \Sigma^*$ and all pairs $x, y \in \Sigma$:*

$$\texttt{cont}_f(x, w)$$

$$\texttt{cont}_f(\ltimes, w)$$

$$\texttt{cont}_f(y, wx)$$

$$\texttt{cont}_f(\ltimes, wx)$$

Proof For any input string $w \in \Sigma^*$, reading w will lead to some state q in $\mathcal{M}$. The target function is subsequential which means that that either $w = w_q$ or else can be replaced thereby since subsequentiality implies that $\texttt{cont}_f(i, w) = \texttt{cont}_f(i, w_q)$ for any $i \in \Sigma \cup \{\ltimes\}$. Now recall that $f^p(w) = \texttt{lcp}(\{u \mid u = f(wv) \wedge v \in \Sigma^*\})$. It is sufficient to use a set containing $f(w)$ and one $f(waz)$ for each $a \in \Sigma$ (where $z \in \Sigma^*$) because every member of Σ^* is either λ or begins with some $a \in \Sigma$. Let us call such a set a *support* for determining $f^p(w)$. If a support exists in the sample, `estimate_fp`(S) adds (p, r) to the set P, where $r = \texttt{lcp}(\{u \mid (w, u) \in S \wedge p \in \texttt{prefixes}(w)\}) = f^p(p)$. By the definition of the characteristic set, for every state q in the canonical transducer and for every triple $a, b, c \in \Sigma$, the learner will see w_q, $w_q a$, $w_q ab$, and $w_q abcv$ where $v \in \Sigma^*$. For any input string w, then, the characteristic set will contain everything necessary to build a support for determining $f^p(w_q)$, $f^p(w_q a)$

and $f^p(w_q ab)$ for all pairs $a, b \in \Sigma$. Finally, recall that $\texttt{cont}_f(\sigma, w) = f^p(w)^{-1} \cdot f^p(w\sigma)$ for $\sigma \in \Sigma$ and that $\texttt{cont}_f(\ltimes, w) = f^p(w)^{-1} \cdot f(w)$. For each pair $x, y \in \Sigma$, the algorithm can thus calculate:

$$\texttt{cont}_f(x, w_q) = f^p(w_q)^{-1} \cdot f^p(w_q x) = \texttt{cont}_f(x, w)$$

$$\texttt{cont}_f(\ltimes, w_q) = f^p(w_q)^{-1} \cdot f(w_q) = \texttt{cont}_f(\ltimes, w)$$

$$\texttt{cont}_f(y, w_q x) = f^p(w_q x)^{-1} \cdot f^p(w_q xy) = \texttt{cont}_f(y, wx)$$

$$\texttt{cont}_f(\ltimes, w_q x) = f^p(w_q x)^{-1} \cdot f(w_q x) = \texttt{cont}_f(\ltimes, wx)$$

■

Lemma 16 Tier convergence
If a learning sample S contains a characteristic set, then $\texttt{get\_tiers}(S)$ will return the canonical tiers of f.

Proof Let T be the canonical x-tier of f, and let H be the x-tier constructed by the algorithm. The algorithm begins with $H = \Delta$, and so either $H = T$ already, or else $H \supset T$. We know from Lemma 12 that if $H \supset T$, there will exist a pair of input strings w_1 and w_2 in the domain of f such that $\texttt{cont}_f(x, w_1) \neq \texttt{cont}_f(x, w_2)$ even though $\texttt{suff}^1_H(f^p(w_1))$ $= \texttt{suff}^1_H(f^p(w_2)) = a$ for some $a \in (H - T)$. We know from Lemma 15 that, given a characteristic set, the algorithm will be able to calculate and check all possible sequences of two contributions out of each tail-equivalence class in the target function. The algorithm will thus flag and remove at least one $a \in (H - T)$ when $H \supset T$. Conversely, there will be no pair of input strings w_3 and w_4 in the domain of f such that $\texttt{cont}_f(x, w_3) \neq \texttt{cont}_f(x, w_4)$ when $\texttt{suff}^1_H(f^p(w_3)) = \texttt{suff}^1_H(f^p(w_4)) = c$ for any $c \in T$. When $H = T$, then, the algorithm will add all $a \in H$ to $\texttt{keep}$, at which point $\texttt{keep} = H = T$. ■

Lemma 17 Polynomial data
There exists a characteristic set that is polynomial in the size of $\mathcal{M}$.

Proof Let Q be the set of states in $\mathcal{M}$. The cardinality of Q is finite and is treated a constant. For item 1 in Definition 14 there are $|Q|$ input-output pairs $(w_q, f(w_q))$ in a characteristic set. For each of these pairs, it is the case that $|w_q| \leq |Q|$ and it is the case that $|f(w_q)|$ is less than or equal to the summed length of the output edges of the machine's transitions. We write this sum as $x_\diamond$ and note that it is linear in the size of the transducer. The overall length of the inputs contributed by item 1 is thus in $\mathcal{O}(|Q|^2)$ and the overall length of the outputs contributed by item 1 is thus in $\mathcal{O}(|Q| \cdot x_\diamond)$. Both of these are quadratic in the size of $\mathcal{M}$.

For items 2i, 2ii, and 2iii in Definition 14, there are respectively $|\Sigma|$, $|\Sigma|^2$ and $|\Sigma|^3$ corresponding input-output pairs per state $q \in Q$. Factoring out the constants gives us $|Q|$ pairs. For the pairs contributed by item 2c, we restrict ourselves without loss of generality to pairs $(w_q abc, f(w_q abc))$. For each pair from item 2, the length of the input is less than or equal to $|Q| + 3$. For each pair from item 2, the length of the output is less than or equal to $x_\diamond$ plus three times the longest output edge in the machine. We write this quantity as

$y_\diamond$ and note that $y_\diamond$ is linear in the size of the transducer. The overall length of the inputs contributed by item 2 is in $\mathcal{O}(|Q|^2)$ and the overall length of the outputs contributed by item 2 is in $\mathcal{O}(|Q| \cdot y_\diamond)$. Both of these are quadratic in the size of $\mathcal{M}$. ■

Theorem 18 *`get_tiers`(S) identifies the canonical tiers of any total strongly target-specified OMTSL$_2$ function in polynomial time and data.*

Proof Immediate from Lemmas 13, 16, and 17. ■

Once the tiers are known, it is relatively easy to construct a finite-state transducer computing the target function. Chandlee et al. (2015) present an efficient and correct algorithm for bulding OSL$_k$ transducers from positive data. The algorithm operates in a breadth-first manner, computing all transitions out of one state before moving to the next, and since the algorithm assumes that the function to be learned is OSL$_k$, it labels the landing state of each transition with the appropriate output suffix. Burness and McMullin (2019) show that the same overall strategy can be used to construct OTSL$_k$ transducers once the tier is known, the only difference being that the state labelling step must use the appropriate *tier* suffix. It is almost certainly the case that with knowledge of the required tiers, we can also adopt the strategy for OMTSL$_k$ transducers if the state labelling step considers the appropriate collection of tier suffixes. The strategy takes time quadratic in the size of the sample, and the amount of necessary data is quadratic in the size of the target transducer (Chandlee et al., 2015).

To close this section, it is worth noting, as pointed out by a reviewer, that our learning result is not entirely novel since the OMTSL$_k$ functions are a strict subclass of the subsequential functions, and so the Onward Subsequential Transducer Inference Algorithm (OSTIA) developed by Oncina et al. (1993) is capable of learning them. The novelty of our algorithm instead lies in the fact that it is designed to learn a particular subclass of subsequential functions that are motivated by the typology of phonological processes. Also, while OSTIA runs in cubic time and would be more time efficient than discovering a set of tiers in quintic time and then building a transducer in quadratic time, there are to our knowledge no analyses of OSTIA's data complexity. It may well be that the reduction from quintic to cubic time that would come from adopting OSTIA over our algorithm is offset by an even larger increase in the size of the necessary data sample.

7. Discussion and Conclusion

The strongly target-specified OMTSL$_2$ functions introduced in this paper are capable of modelling multiple long-distance phonological processes simultaneously, provided that the processes do not interact with each other. We imposed this criterion of independence as it permits efficient identification of the requisite tiers from positive data. Once these tiers are known, the Output Strictly Local Function Inference Algorithm of Chandlee et al. (2015) can be modified to construct a transducer that computes the target function. We need only change the criteria for determining state labels and determining the states in which transitions land (see Burness and McMullin (2019) for a demonstration of how this holds for single-tiered functions).

Of course, phonological processes very often interact with each other, and long-distance processes are by no means exempt. To cite just one case, the Samala language is well-known for its process of regressive sibilant harmony, whereby the anteriority of the rightmost sibilant overrides the anteriority of all sibilants to the left (e.g., /s/ becomes [ʃ] if followed by [ʃ]). The language contains an additional rule affecting the sibilant /s/, according to which it palatalizes to [ʃ] when immediately followed by [t], [n], or [l] as in /s-niʔ/ → [ʃ-niʔ] 'his neck' (Applegate, 1972; McMullin, 2016). The long-distance process is given priority here, such that the local sequences [sn], [st] and [sl] are permitted precisely when palatalization would create a disharmonic sequence of non-local sibilants as in /s-net-us/ → [s-net-us] 'he does it to him' (Applegate, 1972; McMullin, 2016). We could try to model this with a strongly target-specified OMTSL$_2$ function reading from right to left, but we need at least two tiers for /s/. In order to know whether input /s/ should harmonize we need to ignore everything that is not a sibilant, and to know whether input /s/ should palatalize we we cannot ignore anything. Burness and McMullin (2020) defined a weaker form of target specification than the one used in this paper, and this weaker form is capable of modelling the Samala interaction, among other interesting cases. Future work will explore whether various means of permitting input elements to track more than one tier (such as the weaker form of target specification) can maintain efficient learnability.

Acknowledgments

This research was supported by the Social Sciences and Humanities Research Council of Canada.

References

Alëna Aksënova and Sanket Deskmukh. Formal restrictions on multiple tiers. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2018*, pages 64–73, 2018.

Benjamin Ao. Kikongo nasal harmony and context-sensitive underspecification. *Linguistic Inquiry*, 22:193–196, 1991.

Richard B. Applegate. *Ineseño Chumash Grammar.* Doctoral Dissertation, University of California, Berkeley, 1972.

Phillip Burness and Kevin McMullin. Efficient learning of Output Tier-based Strictly 2-Local functions. In *Proceedings of the 16th Meeting on the Mathematics of Language*, pages 78–90. Association for Computational Linguistics, 2019.

Phillip Burness and Kevin McMullin. Multi-tiered strictly local functions. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 245–255. Association for Computational Linguistics, 2020.

Jane Chandlee. *Strictly Local Phonological Processes.* Doctoral Dissertation, University of Delaware, 2014.

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

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. Output strictly local functions. In *Proceedings of the 14th Meeting on the Mathematics of Language (MOL 2015)*, pages 112–125, 2015.

Yael Cohen-Sygal and Shuly Wintner. Finite-state registered automata for non-concatenative morphology. *Computational Linguistics*, 32:49–82, 2006. doi: 10.1162/coli.2006.32.1.49.

Colin de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning Journal*, 27:125–138, 1997.

E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

Yiding Hao and Samuel Andersson. Unbounded stress in subregular phonology. In *Proceedings of the 16th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology and Morphology*, pages 135–143, Florence, Italy, 2019. Association for Computational Linguistics.

Yiding Hao and Dustin Bowers. Action-Sensitive Phonological Dependencies. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology and Morphology*, pages 218–228, Florence, Italy, 2019. Association for Computational Linguistics.

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

Larry Hyman. Positional prominence and the 'prosodic trough' in Yaka. *Phonology*, 15: 41–75, 1998.

Dakotah Lambert and James Rogers. Tier-Based Strictly Local Stringsets: Perspectives from Model and Automata Theory. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2020*, pages 330–337, New Orleans, Louisianna, 2020.

Kevin McMullin. *Tier-Based Locality in Long-Distance Phonotactics: Learnability and Typology.* Doctoral Dissertation, University of British Columbia, Vancouver, BC, 2016.

Kevin McMullin and Gunnar Ólafur Hansson. Long-distance phonotactics as Tier-based Strictly 2-Local Languages. In Adam Albright and Michelle A. Fullwood, editors, *Proceedings of the 2014 Annual Meeting on Phonology*, Washington, DC, 2016. Linguistic Society of America.

Kevin McMullin, Alëna Aksënova, and Aniello De Santo. Learning phonotactic restrictions on multiple tiers. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, volume 2, pages 377–378, 2019.

José Oncina and Pedro Garcia. Inductive learning of subsequential functions. Technical Report DSIC II-34, University Politecnia de Valencia, 1991.

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

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

James Rogers, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. Cognitive and sub-regular complexity. In *Formal Grammar*, number 8036 in Lecture Notes in Artificial Intelligence, pages 90–108. Springer, 2013.