Action-Sensitive Phonological Dependencies

Yiding Hao
Yale University
New Haven, CT, USA
yiding.hao@yale.edu

Dustin Bowers
University of Arizona
Tucson, AZ, USA
bowersd@email.arizona.edu

Abstract

This paper defines a subregular class of functions called the *tier-based synchronized strictly local* (TSSL) functions. These functions are similar to the the tier-based input–output strictly local (TIOSL) functions, except that the locality condition is enforced not on the input and output streams, but on the computation history of the minimal subsequential finite-state transducer. We show that TSSL functions naturally describe rhythmic syncope while TIOSL functions cannot, and we argue that TSSL functions provide a more restricted characterization of rhythmic syncope than existing treatments within Optimality Theory.

1 Introduction

The subregular program in phonology seeks to define subclasses of the regular languages and finite-state functions that describe attested phonotactic constraints and phonological processes. These subclasses provide a natural framework for typological classification of linguistic phenomena while allowing for the development of precise theories of language learning and processing. The traditional view in subregular phonology is that most phonotactic dependencies are described by *tier-based strictly local* languages (TSL, Heinz et al., 2011; McMullin and Hansson, 2016; McMullin, 2016), while most phonological process are described by *strictly local* functions (Chandlee, 2014; Chandlee et al., 2015, In prep). These classes of languages and functions are defined by a principle known as *locality*—that dependencies between symbols must occur over a bounded distance within the string. To account for longer-distance dependencies, Heinz et al. (2011) proposes a *tier projection* mechanism that allows irrelevant intervening symbols to be exempt from the locality condition.

Recent work in subregular phonology has identified a number of exceptions to the traditional view. On the language side, unbounded culminative stress systems (Baek, 2018), Uyghur backness harmony (Mayer and Major, 2018), and Sanskrit n-retroflexion (Graf and Mayer, 2018) have been shown to lie outside the class of TSL languages. These observations have led to an enhancement of Heinz et al.'s (2011) tier projection system. On the function side, a number of processes, including bidirectional harmony systems (Heinz and Lai, 2013) and certain tonal processes (Jardine, 2016), have been shown to be not subsequential, and therefore not strictly local. At least two proposals, both known as the *weakly deterministic* functions, have been made in order to capture these processes (Heinz and Lai, 2013; McCollum et al., 2018).

This paper identifies *rhythmic syncope* as an additional example of a phonological process that is not strictly local. In rhythmic syncope, every second vowel of an underlying form is deleted in the surface form, starting with either the first or the second vowel. While rhythmic syncope cannot be expressed as a local dependency between symbols, it can be viewed as a local dependency between *actions* in the computation history of the minimal subsequential finite-state transducer (SFST). We formalize such dependencies by proposing the *tier-based synchronized strictly local* functions (TSSL). See Bowers and Hao (To appear) for a discussion of TSSL functions oriented towards the phonological literature.

This paper is structured as follows. Section 2 enumerates standard definitions and notation used throughout the paper, while Section 3 reviews existing work on strictly local functions. Section 4 introduces rhythmic syncope and shows that it is not strictly local. Section 5 presents two equivalent definitions of the TSSL functions—an al-

gebraic definition and a definition in terms of a canonical SFST. Section 6 develops some formal properties of the TSSL functions, showing that they are incomparable to the full class strictly local functions. Section 7 compares our proposal to existing OT treatments of rhythmic syncope, and Section 8 concludes.

2 Preliminaries

As usual, $\mathbb{N}$ denotes the set of nonnegative integers. Σ and Γ denote finite alphabets not including the left and right word boundary symbols $\rtimes$ and $\ltimes$, respectively. The length of a string x is denoted by $|x|$, and λ denotes the empty string. Alphabet symbols are identified with strings of length 1, and individual strings are identified with singleton sets of strings. For $k \in \mathbb{N}$, α^k denotes α concatenated with itself k-many times, $\alpha^{<k}$ denotes $\bigcup_{i=0}^{k-1} \alpha^i$, α^* denotes $\bigcup_{i=0}^{\infty} \alpha^i$, and α^+ denotes $\alpha\alpha^*$. The *longest common prefix* of a set of strings A is the longest string $\mathrm{lcp}(A)$ such that every string in A begins with $\mathrm{lcp}(A)$. The k-*suffix* of a string x, denoted $\mathrm{suff}^k(x)$, is the string consisting of the last k-many symbols of $\rtimes^k x$.

A *subsequential finite-state transducer* (SFST) is a 6-tuple $T = \langle Q, \Sigma, \Gamma, q_0, \rightarrow, \sigma \rangle$, where

- Q is the set of *states*, with $q_0 \in Q$ being the *start state*;

- Σ and Γ are the *input* and *output alphabets*, respectively;

- $\rightarrow : Q \times \Sigma \rightarrow Q \times \Gamma^*$ is the *transition function*; and

- $\sigma : Q \rightarrow \Gamma^*$ is the *final output function*.

For $x \in \Sigma^*$; $y \in \Gamma^*$; and $q, r \in Q$, the notation $q \xrightarrow{x:y} r$ means that T emits y to the output stream and transitions to state r if it reads x in the input stream while it is in state q. Letting $f : \Sigma^* \rightarrow \Gamma^*$, we say that T *computes* f if for every $x \in \Sigma^*$, $f(x) = y\sigma(q)$, where $q_0 \xrightarrow{x:y} q$. A function is *subsequential* if it is computed by an SFST.

An SFST $T = \langle Q, \Sigma, \Gamma, q_0, \rightarrow, \sigma \rangle$ is *onward* if for every state q other than q_0,

$$\mathrm{lcp}\left(\left\{ y \middle| \exists x \exists r.q \xrightarrow{x:y} r \right\} \cup \{\sigma(q)\}\right) = \lambda.$$

Putting T in onward form allows us to impose structure on the timing with which SFSTs produce output symbols.

Definition 1. Let $f : \Sigma^* \rightarrow \Gamma^*$. We define the function $f^{\leftarrow} : \Sigma^* \rightarrow \Gamma^*$ by

$$f^{\leftarrow}(x) := \mathrm{lcp}\left(\{f(xy) | y \in \Sigma^*\}\right).$$

For any $x, y \in \Sigma^*$, $f_x^{\rightarrow}(y)$ denotes the string such that $f(xy) = f^{\leftarrow}(x)f_x^{\rightarrow}(y)$. We refer to $f_x^{\rightarrow}$ as the *translation of f by x* and to $f^{\leftarrow}$ as f *top*.[1]

Suppose T computes f. The following facts are apparent.

- Fix $w, x \in \Sigma^*$ and write $q_0 \xrightarrow{x:y} q$ and $q_0 \xrightarrow{x:z} r$. If $q = r$, then $f_w^{\rightarrow} = f_y^{\rightarrow}$.

- T is onward if and only if for all $q \in Q \backslash \{q_0\}$, if $q_0 \xrightarrow{x:y} q$, then $y = f^{\leftarrow}(x)$.

These observations allow us to construct the *minimal SFST for f* by identifying each state with a possible translation $f_x^{\rightarrow}$ (Raney, 1958).

Let A and B be alphabets that are possibly infinite. A function $h : A^* \rightarrow B^*$ is a *homomorphism* if for every $x, y \in A^*$, $h(xy) = h(x)h(y)$.

3 Background

The *strictly local functions* are classes of subsequential functions proposed by Chandlee (2014), Chandlee et al. (2015), and Chandlee et al. (In prep) as transductive analogues of the strictly local languages (McNaughton and Papert, 1971). Whereas phonotactic dependencies can usually be described using *tier-based strictly local* languages (Heinz et al., 2011; McMullin and Hansson, 2016; McMullin, 2016), Chandlee (2014) has argued that local phonological processes can be modelled as strictly local functions when they are viewed as mappings between underlying representations and surface representations. A survey overview of the related literature can be found in Heinz (2018).

Intuitively, strictly local functions are functions computed by SFSTs in which each state represents the $i - 1$ most recent symbols in the input stream and the $j - 1$ most recent symbols in the output stream along with the current input symbol, for some parameter values i, j fixed. Such functions are "local" in the sense that the action performed on each input symbol depends only on information about symbols in the input and output streams

---

[1]This terminology follows Sakarovitch (2009, pp. 692–693). In the transducer inference literature, Oncina et al. (1993) refer to $f_x^{\rightarrow}$ as the *tails of x in f*, and Chandlee et al. (2015) refer to $f^{\leftarrow}$ as the *prefix function associated to f*.

within a bounded distance. In this paper, we augment strictly local functions with *tier projection*, a mechanism introduced by Heinz et al. (2011) and elaborated by Baek (2018), Mayer and Major (2018), and Graf and Mayer (2018) that allows the locality constraint to bypass irrelevant alphabet symbols, extending the distance over which dependencies may be enforced.

Definition 2. For any alphabet Σ, a *tier on Σ* is a homomorphism $\tau : \Sigma^* \to \Sigma^*$ such that for each $a \in \Sigma$, either $\tau(a) = a$ or $\tau(a) = \lambda$. In the former case, we say that a is *on τ*; in the latter case, we say that a is *off τ*.

Chandlee (2014), Chandlee et al. (2015), and Chandlee et al. (In prep) give two definitions of the strictly local functions. Firstly, they state the locality condition in terms of the algebraic representation of minimal SFSTs.

Definition 3. Fix $i, j > 0$ and let τ be a tier on $\Sigma \cup \Gamma$. A function $f : \Sigma^* \to \Gamma^*$ is *i, j-input–output strictly local on tier τ* (i, j-TIOSL) if for all $w, x \in \Sigma^*$, if

- $\mathrm{suff}^{i-1}(\tau(w)) = \mathrm{suff}^{i-1}(\tau(x))$ and

- $\mathrm{suff}^{j-1}(\tau(f^{\leftarrow}(w))) = \mathrm{suff}^{j-1}(\tau(f^{\leftarrow}(x)))$,

then $f_w^{\rightarrow} = f_x^{\rightarrow}$. A function is *$i$-input strictly local on tier τ* (i-TISL) if it is $i, 1$-TIOSL on tier τ, and it is *j-output strictly local on tier τ* (j-TOSL) if it is $1, j$-TIOSL on tier τ.

Secondly, they define strictly local functions in terms of canonical SFSTs that directly encode $(i-1)$-suffixes of the input stream and $(j-1)$-suffixes of the output stream in their state names.

Definition 4. Fix $i, j > 0$ and let τ be a tier on $\Sigma \cup \Gamma$. An SFST $T = \langle Q, \Sigma, \Gamma, q_0, \to, \sigma \rangle$ is *i, j-input–output strictly local on tier τ* (i, j-TIOSL) if the following conditions hold.

- $Q = (\{\rtimes\} \cup \Sigma)^{i-1} \times (\{\rtimes\} \cup \Gamma)^{j-1}$ and $q_0 = \langle \rtimes^{i-1}, \rtimes^{j-1} \rangle$.

- If $\langle a, b \rangle \xrightarrow{x:y} \langle c, d \rangle$, then $c = \mathrm{suff}^{i-1}(\tau(ax))$ and $d = \mathrm{suff}^{j-1}(\tau(by))$.

An SFST is *i-input strictly local on tier τ* (i-TISL) if it is $i, 1$-TIOSL on tier τ, and it is *j-output strictly local on tier τ* (j-TOSL) if it is $1, j$-TIOSL on tier τ.

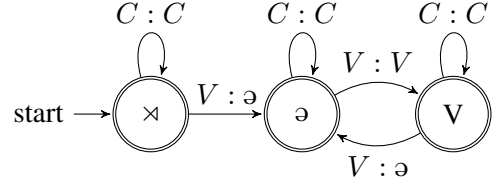These definitions turn out to be equivalent when the canonical SFSTs are required to be onward.

Figure 1: An SFST for rhythmic reduction.

Theorem 5 (Chandlee, 2014; Chandlee et al., 2015, In prep). *A function is i, j-TIOSL on tier τ if and only if it is computed by an onward SFST that is i, j-TIOSL on tier τ.*

Example 6. *Rhythmic reduction* is a phonological process in which alternating vowels in a word undergo reduction. The examples in (7) show rhythmic reduction in the Odawa variety of Ojibwe circa 1912, as documented by Edward Sapir. In our representation of reduction, vowels are reduced to ə, starting from the first vowel. There is no reason to believe that ə appears in underlying forms.

(7)   Rhythmic reduction in Ojibwe circa 1912 (Rhodes et al., 2012)

    a.   /mʌkɪzɪnʌn/ ⤳ [məkɪzənʌn] 'shoes'

    b.   /gʊtɪgʊmɪnʌgɪbɪnaːd/ ⤳ [gətɪgəmɪnəgɪbənaːd] 'if he rolls him'

Figure 1 shows an SFST that implements the rhythmic reduction pattern illustrated in (7). We represent the pattern using an alphabet of three symbols: C, representing consonants; V, representing vowels that have not been reduced; and ə, representing vowels that have been reduced. Observe that this SFST is onward and 2-TOSL, with C off the tier: each state represents the most recent vowel in the ouput stream.[2]

4   Rhythmic Syncope

Rhythmic syncope is a phonological process in which every second vowel in a word is deleted. The examples of (8) show rhythmic syncope in Macushi, in which deletion begins with the first vowel.[3]

(8)   Rhythmic syncope in Macushi (Hawkins, 1950)

a. /piripi/ $\rightsquigarrow$ [pripi] 'spindle'

b. /wanamari/ $\rightsquigarrow$ [wnamri] 'mirror'

In this section, we show that rhythmic syncope is not TIOSL. To see this, we formalize rhythmic syncope as a function over two alphabet symbols: C, representing consonants, and V, representing vowels. This idealization does not affect the argument that rhythmic syncope is not TIOSL, presented in Proposition 10.

Definition 9. The *rhythmic syncope function* ρ : $\{C, V\}^* \rightarrow \{C, V\}^*$ is defined as follows. For $c_0, c_1, \ldots, c_n \in C^*$,

$$\rho(c_0 V c_1 V c_2 \ldots V c_n) = c_0 v_1 c_1 v_2 c_2 \ldots v_n c_n,$$

where for each i, $v_i = V$ if i is even and $v_i = \lambda$ if i is odd.

The intuition underlying the argument presented below is that $(i-1)$-suffixes of the input and $(j-1)$-suffixes of the output do not contain information about whether vowels occupy even or odd positions within the input and output strings. Therefore, while an i, j-TIOSL SFST can record the most recent vowels read from the input stream and emitted to the output stream, this information is not sufficient for determining whether or not the SFST should delete a vowel.

Proposition 10. *The rhythmic syncope function is not i, j-TIOSL on tier τ for any $i, j > 0$ and any* $\tau : \{C, V\}^* \rightarrow \{C, V\}^*$.

Proof. Let $k > i$ be even. Consider the strings $w := V^k$ and $x := V^{k+1}$. Observe that $\rho^{\leftarrow}(w) = \rho^{\leftarrow}(x) = V^{k/2}$; thus $\text{suff}^{j-1}(\tau(\rho^{\leftarrow}(w))) = \text{suff}^{j-1}(\tau(\rho^{\leftarrow}(x)))$. Now, if V is on τ, then $\text{suff}^{i-1}(\tau(w)) = V^{i-1} = \text{suff}^{i-1}(\tau(x))$, and if V is off τ, then $\text{suff}^{i-1}(\tau(w)) = \rtimes^{i-1} = \text{suff}^{i-1}(\tau(x))$. Thus, if ρ is i, j-TIOSL on tier τ, then $\rho_w^{\rightarrow} = \rho_x^{\rightarrow}$. However, letting $y := V^{k/2}$, observe that

$$y = \rho(wV) = \rho^{\leftarrow}(w)\, \rho_w^{\rightarrow}(V) = y\, \rho_w^{\rightarrow}(V)$$
$$yV = \rho(xV) = \rho^{\leftarrow}(x)\, \rho_x^{\rightarrow}(V) = y\, \rho_x^{\rightarrow}(V).$$

This means that $\rho_w^{\rightarrow}(V) = \lambda$ but $\rho_x^{\rightarrow}(V) = V$, so ρ is not i, j-TIOSL on tier τ. $\qquad\square$

5 Synchronized Strictly Local Functions

Proposition 10 raises the question of how to characterize the kind of computation that effects rhythmic syncope. To investigate this question, Figure 2
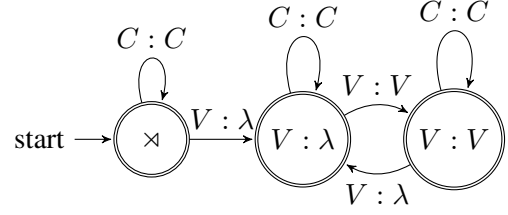

Figure 2: An SFST for rhythmic syncope.

shows a natural SFST implementation of rhythmic syncope. The states in this SFST record the most recent action performed by the SFST. If the most recent action was to delete a vowel ($V : \lambda$), then the next vowel the SFST encounters is not deleted ($V : V$); otherwise, the next vowel is deleted. This SFST is strikingly similar to the rhythmic reduction SFST in Figure 1. There, the special symbol ə, which is not part of the input alphabet, indicates the location of a reduced vowel, effectively recording the previous action in the output. Since there is no way to mark the location of a deleted symbol, the SFST in Figure 2 explicitly records its previous action in its state names. Thus, the rhythmic syncope SFST may be seen as a generalization of the rhythmic reduction SFST. The goal of this section is to define a class of functions, known as the *tier-based synchronized strictly local* (TSSL) functions, based on this intuition. Following Section 2, we begin by defining the TSSL functions algebraically in terms of the minimal SFST, and then we define a canonical SFST format for the TSSL functions.

Recall that at each time step, an SFST must read exactly one input symbol while producing an output string of any length. Since the minimal SFST for a function f must produce $f^{\leftarrow}(z)$ after reading the input string z, we can determine the possible actions of f by comparing $f^{\leftarrow}(z)$ with $f^{\leftarrow}(zx)$ for arbitrary $z \in \Sigma^*$ and $x \in \Sigma$.

Definition 11. Let $f : \Sigma^* \rightarrow \Gamma^*$. The *actions of f* are the alphabet $\mathcal{A}(f) \subseteq \Sigma \times \Gamma^*$ defined as follows.

$$\mathcal{A}(f) := \{\langle x, y \rangle | \exists z \in \Sigma^*. f^{\leftarrow}(zx) = f^{\leftarrow}(z)y\}$$

We denote elements $\langle x, y \rangle$ of $\mathcal{A}(f)$ by $x : y$.

Strings over $\mathcal{A}(f)$ represent computation histories of the minimal SFST for f.

Definition 12. Let $x \in \Sigma^*$ and let $f : \Sigma^* \rightarrow \Gamma^*$. The *run of f on input x* is the string $f^{\Leftarrow}(x) \in \mathcal{A}(f)^*$ defined as follows.

- If $|x| \leq 1$, then $f^{\Leftarrow}(x) := x : f^{\leftarrow}(x)$.

- If $x = yz$, where $|y| \geq 1$ and $|z| = 1$, then $f^{\Leftarrow}(x) := f^{\Leftarrow}(y)(z : w)$, where w is the unique string such that $f^{\leftarrow}(x) = f^{\leftarrow}(y)w$.

The notation $f^{\Leftarrow}$ allows us to define the TSSL functions in a straightforward manner, highlighting the analogy to the TIOSL functions.

Definition 13. Fix $k > 0$ and let τ be a tier on $\Sigma \times \Gamma^*$. A function $f : \Sigma^* \to \Gamma^*$ is k-*synchronized strictly local on tier* τ (k-TSSL) if for all $x, y \in \Sigma^*$, if $\mathrm{suff}^{k-1}(\tau(f^{\Leftarrow}(x))) = \mathrm{suff}^{k-1}(\tau(f^{\Leftarrow}(y)))$, then $f_x^{\to} = f_y^{\to}$.

Now, let us define the canonical SFSTs for TSSL functions. We define the actions of an SFST to be its possible transition labels.

Definition 14. Let $T = \langle Q, \Sigma, \Gamma, q_0, \to, \sigma \rangle$ be an SFST. The *actions of* T are the alphabet

$$\mathcal{A}(T) := \{\langle x, y \rangle | \exists q \exists r. \to(q, x) = \langle r, y \rangle\}.$$

We denote elements $\langle x, y \rangle$ of $\mathcal{A}(T)$ by $x : y$.

Again, the definition of the TSSL SFSTs is directly analogous to that of the TIOSL SFSTs.

Definition 15. Fix $k > 0$ and let τ be a tier on $\Sigma \times \Gamma^*$. An SFST $T = \langle Q, \Sigma, \Gamma, q_0, \to, \sigma \rangle$ is k-*synchronized strictly local on tier* τ (k-TSSL) if the following conditions hold.

- $Q = (\{\rtimes\} \cup \mathcal{A}(T))^{k-1}$ and $q_0 = \rtimes^{k-1}$.

- For every $q \in Q$, if $\to(q, x) = \langle r, y \rangle$, then

$$r = \mathrm{suff}^{k-1}(\tau(q(x : y))).$$

As is the case with TIOSL SFSTs, TSSL SFSTs compute exactly the class of TSSL functions when they are required to be onward.

Theorem 16. Fix $k > 0$, and let τ be a tier on $\Sigma \times \Gamma^*$. A function is k-TSSL on tier τ if and only if it is computed by an onward SFST that is k-TSSL on tier τ.

We leave the proof of this fact to Appendix A.

6 Properties of TSSL Functions

Having now defined the TSSL functions, this section investigates some of their formal properties. Subsection 6.1 compares the TSSL functions to the TISL, TOSL, and TIOSL functions. Subsection 6.2 observes that TSSL SFSTs compute a large class of functions when they are not required to be onward.

6.1 Relation to TIOSL Functions

A natural first question regarding the TSSL functions is that of how they relate to previously-proposed classes of subregular functions. We know from the discussion of rhythmic syncope that the TSSL functions are not a subset of the TIOSL functions: we have already seen that the rhythmic syncope function is 2-TSSL but not i, j-TIOSL for any i, j. We will see in this subsection that the TIOSL functions are not a subset of the TSSL functions, though both function classes fully contain the TISL and TOSL functions. Therefore, the two function classes are incomparable, and offer two different ways to generalize the TISL and TOSL functions.

The fact that the TSSL functions contain the TISL and TOSL functions follows from the observation that actions contain information about input and output symbols. Remembering the i most recent actions automatically entails remembering the i most recent input symbols, and the j most recent output symbols can be extracted from the j most recent actions if deletions are ignored.

Proposition 17. *Fix $k > 0$. Every k-TISL function and every k-TOSL function is k-TSSL.*

Proof. Let $f : \Sigma^* \to \Gamma^*$, and let τ be a tier on $\Sigma \cup \Gamma$. First, suppose that f is k-TISL on tier τ. Let υ be a tier on $\Sigma \times \Gamma^*$ defined as follows: an action $x : y$ is on υ if and only if x is on τ. Now, suppose $w, x \in \Sigma^*$ are such that $\mathrm{suff}^{k-1}(\upsilon(f^{\Leftarrow}(w))) = \mathrm{suff}^{k-1}(\upsilon(f^{\Leftarrow}(x)))$. Write

$$\upsilon(f^{\Leftarrow}(w)) = (w_1 : y_1)(w_2 : y_2) \ldots (w_n : y_n)$$
$$\upsilon(f^{\Leftarrow}(x)) = (x_1 : z_1)(x_2 : z_2) \ldots (x_n : z_n).$$

Then, we have $\tau(w) = w_1 w_2 \ldots w_n$ and $\tau(x) = x_1 x_2 \ldots x_n$. For all $i > n - k + 1$, $w_i : y_i = x_i : z_i$, and therefore $w_i = x_i$. But this means that $\mathrm{suff}^{k-1}(\tau(w)) = \mathrm{suff}^{k-1}(\tau(x))$, and since f is k-TISL on tier τ, $f_w^{\leftarrow} = f_x^{\leftarrow}$. We conclude that f is k-TSSL on tier υ.

Next, suppose that f is k-TOSL on tier τ. Let φ be a tier on $\Sigma \times \Gamma^*$ defined as follows: an action $x : y$ is on φ if and only if $\tau(y) \neq \lambda$. Now, suppose $w, x \in \Sigma^*$ are such that $\mathrm{suff}^{k-1}(\varphi(f^{\Leftarrow}(w))) = \mathrm{suff}^{k-1}(\varphi(f^{\Leftarrow}(x)))$. Write

$$\varphi(f^{\Leftarrow}(w)) = (w_1 : y_1)(w_2 : y_2) \ldots (w_n : y_n)$$
$$\varphi(f^{\Leftarrow}(x)) = (x_1 : z_1)(x_2 : z_2) \ldots (x_n : z_n).$$

Now, $\tau(f^{\leftarrow}(w)) = y_1 y_2 \ldots y_n$ and $\tau(f^{\leftarrow}(x)) = z_1 z_2 \ldots z_n$. Again, for all $i > n - k + 1$ we have
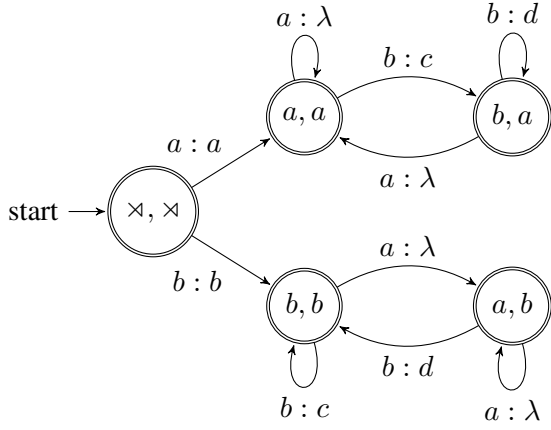
Figure 3: An onward $2, 2$-TIOSL SFST computing a function that is not k-TSSL for any k.

$w_i : y_i = x_i : z_i$, so $y_i = z_i$. Observe that

$$\text{suff}^{k-1}(\tau(f^{\leftarrow}(w))) = \text{suff}^{k-1}(y_j y_{j+1} \ldots y_n)$$
$$= \text{suff}^{k-1}(z_j z_{j+1} \ldots z_n) = \text{suff}^{k-1}(\tau(f^{\leftarrow}(x))),$$

where $j = n - k + 2$. Since f is k-TOSL on tier τ, $f_w^{\rightarrow} = f_x^{\rightarrow}$, so f is k-TSSL on tier φ. $\qquad \square$

This intuition does not carry over to the TIOSL functions. In Proposition 17, the proposed action tiers ignore symbols off the input and output tiers, thus ensuring that the relevant input and output symbols can always be recovered from the computation history. This approach encounters problems when an onward TIOSL SFST deletes symbols on the tier. Such SFSTs perform actions of the form $x : \lambda$, where x is on the tier. These actions do not record any output symbols, but they must be kept on the tier in a TSSL implementation so that the input symbol x can be recovered. If too many $(x : \lambda)$s are performed consecutively, they can overwhelm the memory of a TSSL SFST, causing it to forget the most recent output symbols. The following construction features exactly this kind of behavior.

Proposition 18. *There exists a function that is i, j-TIOSL for some i, j but not k-TSSL for any k.*

Proof. Let T be the SFST shown in Figure 3, and let $f : \{a, b\}^* \rightarrow \{a, b, c, d\}^*$ be the function computed by T.[4] Observe that T is onward and $2, 2$-TIOSL on tier τ, where a and b are on τ but c and d are not, so f is $2, 2$-TIOSL on tier τ. T always copies the first symbol of its input to the

---

[4]The angle brackets are omitted from the state names.

output. Thereafter, T behaves as follows: all as are deleted; a b is changed to a c if the most recent input symbol is the same as the first input symbol; a b is changed to a d otherwise. For example, $f(baabb) = bdc$.

Let $k > 0$, and let υ be a tier on $\{a, b\} \times \{a, b, c, d\}^*$. Suppose that either $k = 1$ or $a : \lambda$ is not on υ, and consider the strings $w := ba$ and $x := b$. Observe that $f^{\leftarrow}(w) = (b : b)(a : \lambda)$ and $f^{\leftarrow}(x) = (b : b)$. Either $\text{suff}^{k-1}(\upsilon(f^{\leftarrow}(x))) = \rtimes^{k-2}(b : b) = \text{suff}^{k-1}(\upsilon(f^{\leftarrow}(x)))$ if $k > 1$ and $b : b$ is on υ, or $\text{suff}^{k-1}(\upsilon(f^{\leftarrow}(x))) = \rtimes^{k-1} = \text{suff}^{k-1}(\upsilon(f^{\leftarrow}(x)))$ if $k = 1$ or $b : b$ is not on υ. However, $f_w^{\rightarrow}(b) = d$ but $f_x^{\rightarrow}(b) = c$, so f cannot be k-TSSL on tier υ.

Next, suppose that $k > 1$ and $a : \lambda$ is on υ. Consider the input strings $w := a^{k+1}$ and $x := ba^k$. Observe that $f^{\leftarrow}(w) = (a : a)(a : \lambda)^k$ and $f^{\leftarrow}(x) = (b : b)(a : \lambda)^k$, thus

$$\text{suff}^{k-1}(\upsilon(f^{\leftarrow}(w))) = (a : \lambda)^{k-1}$$
$$= \text{suff}^{k-1}(\upsilon(f^{\leftarrow}(x))).$$

However, $f_w^{\rightarrow}(b) = c$ but $f_x^{\rightarrow}(b) = d$, so f is not k-TSSL on tier υ. $\qquad \square$

6.2 Non-Onward TSSL SFSTs

The equivalence between the two definitions of the TSSL functions presented in Section 5 crucially depends on the criterion that TSSL SFSTs be onward. In this subsection we show that without this criterion, TSSL SFSTs compute a rich class of subsequential functions. To illustrate how this is possible, let us consider an example that witnesses the separation between TSSL functions and TSSL SFSTs.

Proposition 19. *There exists a 2-TSSL SFST that computes a function that is not k-TSSL for any k.*

Proof. Consider the SFST in Figure 4. This SFST is clearly 2-TSSL on a tier containing all actions, and the function it computes is given by $f(xy) = xyx$, where $x \in \{a, b\}$ and $y \in \{a, b\}^*$. Observe that for any $z \in \{a, b\}^*$, $f^{\leftarrow}(z) = z$. Therefore, writing $z = z_1 z_2 \ldots z_n$ with $|z_i| = 1$ for each i,

$$f^{\leftarrow}(z) = (z_1 : z_1)(z_2 : z_2) \ldots (z_n : z_n).$$

We need to show that f is not k-TSSL for any $k > 0$ and for any tier τ over $\{a, b\} \times \{a, b\}^*$.

Fix k and τ. Suppose $a : a$ is on τ, and consider the input strings $w = a^{k+1}$ and $x = ba^k$. Observe
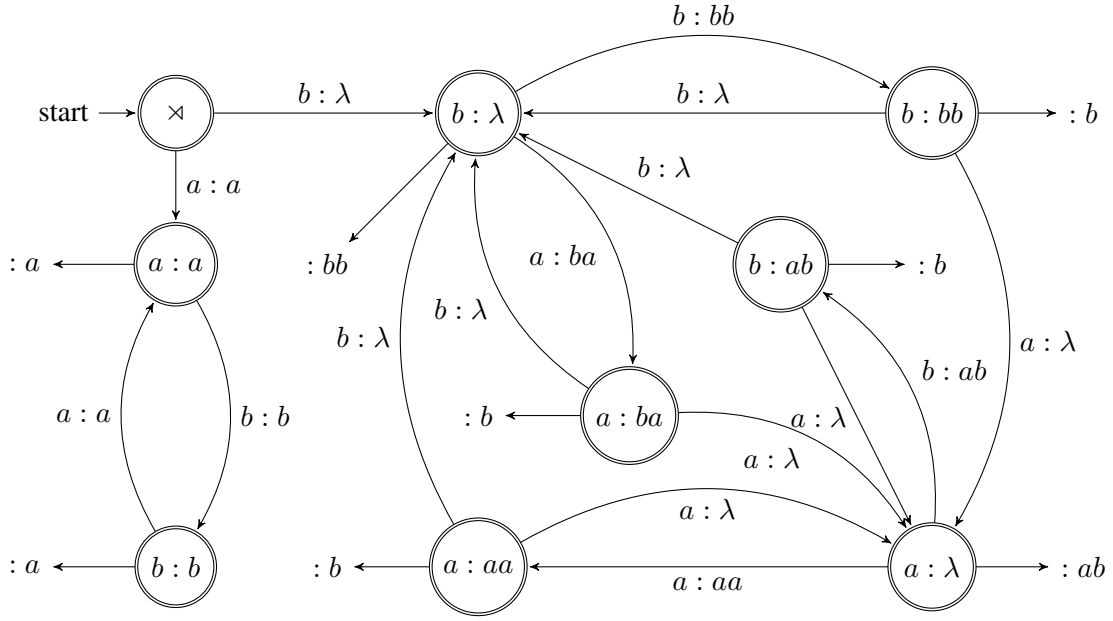
Figure 4: A non-onward 2-TSSL SFST computing a function that is not k-TSSL for any k.

that $f^{\Leftarrow}(w) = (a : a)^{k+1}$ and $f^{\Leftarrow}(x) = (b : b)(a : a)^k$, so

$$\operatorname{suff}^{k-1}(\tau(f^{\Leftarrow}(w))) = (a : a)^{k-1}$$
$$= \operatorname{suff}^{k-1}(\tau(f^{\Leftarrow}(x))).$$

However, $f_w^{\rightarrow}(\lambda) = a$ but $f_x^{\rightarrow}(\lambda) = b$, so f is not k-TSSL on tier τ.

Next, suppose $a : a$ is not on τ, and consider the input strings $w = b$ and $x = ab$. We have $f^{\Leftarrow}(w) = b : b$ and $f^{\Leftarrow}(x) = (a : a)(b : b)$, so

$$\operatorname{suff}^{k-1}(\tau(f^{\Leftarrow}(w))) = \operatorname{suff}^{k-1}(\tau(b : b))$$
$$= \operatorname{suff}^{k-1}(\tau(a : a)\tau(b : b))$$
$$= \operatorname{suff}^{k-1}(\tau((a : a)(b : b)))$$
$$= \operatorname{suff}^{k-1}(\tau(f^{\Leftarrow}(x))).$$

However, $f_w^{\rightarrow}(\lambda) = b$ but $f_x^{\rightarrow}(\lambda) = a$, so f is not k-TSSL on tier τ. $\square$

Let f be the function described in Proposition 19. As discussed in the proof, an onward SFST computing f must copy the current input symbol to the output stream during each time step. At the end of the computation, the final output function is responsible for adding the first input symbol to the end of the output string. Any onward TSSL SFST that attempts to compute f will eventually forget the identity of the first input symbol, so the final output function cannot determine what to add to the output. The SFST T in Figure 4 avoids this problem by exploiting its non-onwardness. If the

first symbol of its input is an a, then T behaves in an onward manner, copying the current input symbol at each time step. This can be seen in the left column of the state diagram. If the first symbol of T's input is a b, then T alternates between producing no output and producing two symbols of output. Every time T performs a non-deleting action $x : y$, y contains both the symbol that the onward SFST would produce at the current time step and the symbol that the onward SFST would have produced at the previous time step. This way, T encodes the identity of the first symbol of its input using the manner in which it produces output—if T produces output at every time step, then the first symbol is an a, and if it produces output every two time steps, then the first symbol is a b. In general, this kind of encoding trick can be applied to a wide range of SFSTs, including all SFSTs V that do not perform deletions. Informally, we enumerate the states of V by $\{q_0, q_1, \ldots, q_n\}$, and we construct a TSSL SFST S that simulates V by producing output at various frequencies. For each i, S produces output every $i + 1$ time steps if V is in state q_i. If S remembers at least $2(n + 1)$-many actions, then it can always deduce V's state at any point in the computation, allowing it to simulate V.

7 Rhythmic Syncope in Phonology

The view of rhythmic syncope we have presented here differs substantially in approach from existing treatments of rhythmic syncope in phonolog-

ical theory. McCarthy (2008) identifies two major approaches to rhythmic syncope in Optimality Theory. In the *pseudo-deletion* approach (e.g., Kager, 1997), the locations of symbols deleted by syncope are marked with blank symbols. This essentially makes rhythmic syncope identical to rhythmic reduction, which we have seen is 2-TOSL. McCarthy himself proposes a *Harmonic Serialism* approach in which rhythmic syncope is implemented in multiple steps. Firstly, stress is assigned to every second vowel in the underlying form. Then, the unstressed vowels are deleted, resulting in syncope. This kind of derivation is illustrated in (20).

(20) Rhythmic syncope in Harmonic Serialism
      (McCarthy, 2008)[5]
      | /wanamari/ | Underlying Form |
      | wanámarí | Stress |
      | [wnámrí] | Syncope |

In both approaches, rhythmic syncope is decomposed into a 2-TOSL function and a homomorphism. In the pseudo-deletion approach, the 2-TOSL function is rhythmic reduction, and the homomorphism removes the əs. In (20), the rhythmic stress step is 2-TOSL, while the syncope step is a homomorphism. In general, this kind of approach is extremely powerful.

Proposition 21. *Every subsequential function f can be written in the form $f = h \circ g$, where g is 2-TOSL and h is a homomorphism.*

Proof. Let $T = \langle Q, \Sigma, \Gamma, q_0, \rightarrow, \sigma \rangle$ be the minimal SFST for f. Define g as follows. Let $g(\lambda) := \langle \sigma, f(\lambda) \rangle$. For $x_1, x_2, \ldots, x_n \in \Sigma$, write

$$q_0 \xrightarrow{x_1:y_1} q_1 \xrightarrow{x_2:y_2} q_2 \xrightarrow{x_3:y_3} \ldots \xrightarrow{x_{n-1}:y_{n-1}} q_n.$$

Then, $g(x_1 x_2 \ldots x_n) := \langle q_1, y_1 \rangle \langle q_2, y_2 \rangle \ldots \langle q_n, y_n \rangle \langle \sigma, \sigma(q_n) \rangle$. Next, define h so that for any $\langle q, y \rangle$, $h(\langle q, y \rangle) = y$. It is clear that $f(x) = h(g(x))$ for every x. We now show that g is 2-TOSL on a tier containing the full output alphabet.

Fix $w, x \in \Sigma^*$. Observe that for all $z \in \Sigma^*$, $g^{\leftarrow}(z) \in (Q \times \Gamma^*)^*$. Therefore, suppose that $\text{suff}^1(g^{\leftarrow}(w)) = \text{suff}^1(g^{\leftarrow}(x)) = \langle q, y \rangle$. This means that $q_0 \xrightarrow{w:u} q$ and $q_0 \xrightarrow{x:v} q$ for some $u, v \in \Gamma^*$, so $g_u^{\rightarrow} = g_v^{\rightarrow}$ by definition. $\qquad \square$

---

[5]The full derivation proposed by McCarthy (2008) includes syllabification and footing steps, which are omitted here for simplicity.

In both pseudo-deletion and Harmonic Serialism, non-segmental phonological symbols are used to encode state information in the output, making rhythmic syncope 2-TOSL. Proposition 21 shows that this technique can be applied to arbitrary SFSTs, and therefore results in massive overgeneration. By contrast, we have already seen that the TSSL functions are a proper subset of the subsequential functions, making action-sensitivity a more restrictive alternative to current approaches to rhythmic syncope.

8   Conclusion

The classic examples of TIOSL phenomena in phonology are local processes and unidirectional spreading processes (Chandlee, 2014). Rhythmic syncope is qualitatively different from these phenomena in that it leaves no evidence that the process has occurred. As we have seen in Section 4, the fact that rhythmic syncope is not TIOSL is a consequence of this property. In defining the TSSL functions, we have proposed that rhythmic syncope should be viewed as a dependency between incremental steps in a derivation, here formalized as the actions of the minimal SFST.

A potential risk of such an analysis is that the notion of "action" is specific to the computational system used to implement rhythmic syncope, and therefore potentially subject to a broad range of interpretations. In this paper, we have used onwardness and the existence of the minimal SFST to formulate a notion of "action-sensitivity" that is both formalism-independent and implementation-independent. In Subsection 6.2, we have seen that action-sensitivity can be made very powerful if we relax our assumptions about the nature of the computation. This means that if action-sensitivity is to be incorporated into phonological analyses of rhythmic syncope, then care should be taken to avoid loopholes like the one featured in Proposition 19. Based on Proposition 21, a similar warning can be made regarding the composition of phonological processes. When decomposing phonemena into several processes, as McCarthy (2008) does in the Harmonic Serialism analysis, care should be taken to ensure that theoretical proposals do not allow for overgeneration.

Outstanding formal questions regarding the TSSL functions include their closure properties and the complexity of learning TSSL functions. We leave such questions to future work.

References

Hyunah Baek. 2018. Computational representation of unbounded stress: Tiers with structural features. In *Proceedings of CLS 53 (2017)*, volume 53, pages 13–24, Chicago, IL, USA. Chicago Linguistic Society.

Dustin Bowers. To appear. The Nishnaabemwin Restructuring Controversy: New Empirical Evidence. *Phonology*.

Dustin Bowers and Yiding Hao. To appear. Rhythmic Syncope in Subregular Phonology. In *Proceedings of the 42nd Annual Penn Linguistics Conference*, volume 26.1 of *Penn Working Papers in Linguistics*, Philadelphia, PA, USA. Penn Graduate Linguistics Society.

Jane Chandlee. 2014. *Strictly Local Phonological Processes*. PhD Dissertation, University of Delaware, Newark, DE, USA.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. 2015. Output Strictly Local Functions. In *Proceedings of the 14th Meeting on the Mathematics of Language*, pages 112–125, Chicago, IL, USA. Association for Computational Linguistics.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. In prep. Input–output strictly local functions and their efficient learnability.

Thomas Graf and Connor Mayer. 2018. Sanskrit n-Retroflexion is Input–Output Tier-Based Strictly Local. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 151–160, Brussels, Belgium. Association for Computational Linguistics.

W. Neil Hawkins. 1950. Patterns of Vowel Loss in Macushi (Carib). *International Journal of American Linguistics*, 16(2):87–90.

Jeffrey Heinz. 2018. The computational nature of phonological generalizations. In Larry M. Hyman and Frans Plank, editors, *Phonological Typology*, number 23 in Phonology and Phonetics, pages 126–195. De Gruyter Mouton, Berlin, Germany.

Jeffrey Heinz and Regine Lai. 2013. Vowel Harmony and Subsequentiality. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pages 52–63, Sofia, Bulgaria. Association for Computational Linguistics.

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: Human Language Technologies*, pages 58–64, Portland, OR, USA. Association for Computational Linguistics.

Adam Jardine. 2016. Computationally, tone is different. *Phonology*, 33(2):247–283.

René Kager. 1997. Rhythmic vowel deletion in Optimality Theory. In Iggy Roca, editor, *Derivations and Constraints in Phonology*, pages 463–499. Clarendon Press, Oxford, United Kingdom.

Connor Mayer and Travis Major. 2018. A Challenge for Tier-Based Strict Locality from Uyghur Backness Harmony. In *Formal Grammar 2018, 23rd International Conference, FG 2018, Sofia, Bulgaria, August 11-12, 2018, Proceedings*, volume 10950 of *Lecture Notes in Computer Science*, pages 62–83, Berlin, Germany. Springer Berlin Heidelberg.

John J. McCarthy. 2008. The serial interaction of stress and syncope. *Natural Language & Linguistic Theory*, 26(3):499–546.

Adam McCollum, Eric Baković, Anna Mai, and Eric Meinhardt. 2018. The expressivity of segmental phonology and the definition of weak determinism. *LingBuzz*, lingbuzz/004197.

Kevin McMullin and Gunnar Ólafur Hansson. 2016. Long-Distance Phonotactics as Tier-Based Strictly 2-Local Languages. In *Proceedings of the 2014 Annual Meeting on Phonology*, Proceedings of the Annual Meetings on Phonology, pages 13–24, Cambridge, MA, USA. Linguistic Society of America.

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

Robert McNaughton and Seymour A. Papert. 1971. *Counter-Free Automata*. Number 65 in Research Monograph. MIT Press, Cambridge, MA, USA.

José Oncina, Pedro Garcia, and Enrique Vidal. 1993. Learning Subsequential Transducers for Pattern Recognition Interpretation Tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):448–458.

George N. Raney. 1958. Sequential Functions. *Journal of the Association for Computing Machinery*, 5(2):177–180.

Richard A. Rhodes, Karl S. Hele, and J. Randolph Valentine. 2012. Algonquian Trade Languages Revisited. In *Papers of the Fortieth Algonquian Conference/Actes Du Congrès Des Algonquinistes*, Papers of the Algonquian Conference, pages 358–369, Albany, NY, USA. State University of New York Press.

Jacques Sakarovitch. 2009. *Elements of Automata Theory*. Cambridge University Press, Cambridge, United Kingdom.

A   Proof of Theorem 16

This appendix proves the equivalence between TSSL functions and onward TSSL SFSTs. We begin by showing how to construct an onward TSSL SFST computing any given TSSL function.

Definition 22. Let $f : \Sigma^* \to \Gamma^*$ be k-TSSL on tier τ. Define the SFST transducer $\mathcal{T}(f) = \langle Q, \Sigma, \Gamma, q_0, \to, \sigma \rangle$ as follows.

- $Q := (\{\rtimes\} \cup \mathcal{A}(f))^{k-1}$ and $q_0 := \rtimes^{k-1}$.

- For each $x \in \Sigma$, $\to(q_0, x) := \langle r, f^{\leftarrow}(x) \rangle$, where $r = \text{suff}^{k-1}(\tau(x : f^{\leftarrow}(x)))$.

- For each $q \in Q \backslash \{q_0\}$, let $x \in \Sigma^*$ be such that $\text{suff}^{k-1}(\tau(f^{\Leftarrow}(x))) = q$, and let $w : y \in \mathcal{A}(f)$ be such that $f^{\leftarrow}(xw) = f^{\leftarrow}(x)y$. We define $\to(q, w) := \langle r, y \rangle$, where $r = \text{suff}^{k-1}(\tau(q(w : y)))$.

- Fix $q \in Q$. If $q = q_0$, then $\sigma(q) := f(\lambda)$. Otherwise, we define $\sigma(q) := f_x^{\to}(\lambda)$, where $\text{suff}^{k-1}(\tau(f^{\Leftarrow}(x))) = q$.

Remark 23. $\mathcal{T}(f)$ is k-TSSL on tier τ.

Note that in the third and fourth bullet points of Definition 22, the action $w : y$ and the string $f_x^{\to}(\lambda)$ only depend on q and not on x, since f is k-TSSL on tier τ. We now need to show that $\mathcal{T}(f)$ computes f and that it is onward.

Lemma 24. Let $f : \Sigma^* \to \Gamma^*$ be k-TSSL on tier τ, and write $\mathcal{T}(f) = \langle Q, \Sigma, \Gamma, q_0, \to, \sigma \rangle$. For every $x \in \Sigma^+$, if $q_0 \xrightarrow{x:y} r$, then $y = f^{\leftarrow}(x)$.

Proof. Let us induct on $|x|$. For the base case, suppose $|x| = 1$. Then, $y = f^{\leftarrow}(x)$ by definition.

Now, fix $n > 1$, and suppose that if $0 < |u| < n$ and $q_0 \xrightarrow{u:v} r$, then $v = f^{\leftarrow}(u)$. Fix $w \in \Sigma^{n-1}$ and $x \in \Sigma$, and suppose that $q_0 \xrightarrow{w:y} s \xrightarrow{x:z} t$. By the induction hypothesis, $y = f^{\leftarrow}(w)$. The definition of $\mathcal{T}(f)$ states that z is the unique string such that $f^{\leftarrow}(wx) = f^{\leftarrow}(w)z$. Thus, $yz = f^{\leftarrow}(w)z = f^{\leftarrow}(wx)$, and the proof is complete. $\square$

Lemma 25. Let $f : \Sigma^* \to \Gamma^*$ be k-TSSL on tier τ, and write $\mathcal{T}(f) = \langle Q, \Sigma, \Gamma, q_0, \to, \sigma \rangle$. For all $x \in \Sigma^+$, if $q_0 \xrightarrow{x:y} r$, then $r = \text{suff}^{k-1}(\tau(f^{\Leftarrow}(x)))$.

Proof. Let us induct on $|x|$. For the base case, suppose $|x| = 1$. Since $f^{\Leftarrow}(x) = x : f^{\leftarrow}(x)$, by definition $r = \text{suff}^{k-1}(\tau(f^{\Leftarrow}(x)))$.

Now, fix $n > 1$, and suppose that if $|w| < n$ and $q_0 \xrightarrow{w:y} r$, then $r = \text{suff}^{k-1}(\tau(f^{\Leftarrow}(w)))$. We need to show that for all $w \in \Sigma^{n-1}$ and $x \in \Sigma$, if $q_0 \xrightarrow{w:y} r \xrightarrow{x:z} s$, then $s = \text{suff}^{k-1}(\tau(f^{\Leftarrow}(wx)))$. The induction hypothesis gives us $r = \text{suff}^{k-1}(\tau(f^{\Leftarrow}(w)))$. Since $\langle s, z \rangle = \to(r, x)$, by the definition of $\mathcal{T}(f)$,

$$s = \text{suff}^{k-1}(\tau(r(x : z)))$$

$$= \text{suff}^{k-1}(\tau(r)\tau(x : z))$$

$$= \text{suff}^{k-1}\left(\tau\left(\text{suff}^{k-1}(\tau(f^{\Leftarrow}(w)))\right)\tau(x : z)\right)$$

$$= \text{suff}^{k-1}(\tau(\tau(f^{\Leftarrow}(w)))\tau(x : z))$$

$$= \text{suff}^{k-1}(\tau(f^{\Leftarrow}(w))\tau(x : z))$$

$$= \text{suff}^{k-1}(\tau(f^{\Leftarrow}(w)(x : z)))$$

$$= \text{suff}^{k-1}(\tau(f^{\Leftarrow}(wx))), \tag{26}$$

as desired. $\square$

Proposition 27. If $f : \Sigma^* \to \Gamma^*$ is k-TSSL on tier τ, then $\mathcal{T}(f)$ computes f.

Proof. We need to show that for every $x \in \Sigma^*$, $\mathcal{T}(f)$ outputs $f(x)$ on input x. Write $\mathcal{T}(f) = \langle Q, \Sigma, \Gamma, q_0, \to, \sigma \rangle$ and $q_0 \xrightarrow{x:y} q$. By Lemma 24, $y = f^{\leftarrow}(x)$, and by Lemma 25, $q = \text{suff}^{k-1}(\tau(f^{\Leftarrow}(x)))$. Definition 22 then states that $\sigma(q) = f_x^{\to}(\lambda)$, so $y\sigma(q) = f^{\leftarrow}(x)f_x^{\to}(\lambda) = f(x)$, thus $\mathcal{T}(f)$ outputs $f(x)$ on input x. $\square$

Corollary 28. If $f : \Sigma^* \to \Gamma^*$ is k-TSSL on tier τ, then $\mathcal{T}(f)$ is onward.

We then complete the proof by showing that every onward TSSL SFST computes a TSSL function.

Lemma 29. Let $T = \langle Q, \Sigma, \Gamma, q_0, \to, \sigma \rangle$ be onward and k-TSSL on tier τ. Let f be the function computed by T. For all $x \in \Sigma^*$, if $q_0 \xrightarrow{x:y} q$, then $q = \text{suff}^{k-1}(\tau(f^{\Leftarrow}(x)))$.

Proof. Let us induct on $|x|$. For the base case, suppose $|x| = 1$. Since T is onward, $y = f^{\leftarrow}(x)$, so

$$q = \text{suff}^{k-1}\left(\tau\left(\rtimes^{k-1}(x : y)\right)\right)$$

$$= \text{suff}^{k-1}\left(\tau\left(\rtimes^{k-1}(x : f^{\leftarrow}(x))\right)\right)$$

$$= \text{suff}^{k-1}\left(\tau\left(\rtimes^{k-1}f^{\Leftarrow}(x)\right)\right)$$

$$= \text{suff}^{k-1}\left(\tau\left(\rtimes^{k-1}\right)\tau(f^{\Leftarrow}(x))\right)$$

$$= \text{suff}^{k-1}(\tau(f^{\Leftarrow}(x))).$$

Now, fix $n > 1$, and suppose that if $|w| < n$ and $q_0 \xrightarrow{w:y} q$, then $q = \text{suff}^{k-1}(\tau(f^{\Leftarrow}(w)))$. We need to show that for all $w \in \Sigma^{n-1}$ and $x \in \Sigma$, if $q_0 \xrightarrow{w:y} r \xrightarrow{x:z} s$, then $s = \text{suff}^{k-1}(\tau(f^{\Leftarrow}(wx)))$. The induction hypothesis gives us $r = \text{suff}^{k-1}(\tau(f^{\Leftarrow}(w)))$, and Definition 15 states that $s = \text{suff}^{k-1}(\tau(r(x : z)))$. A derivation similar to equation (26) then gives us $s = \text{suff}^{k-1}(\tau(f^{\Leftarrow}(wx)))$, as desired. $\square$

Proof of Theorem 16. Proposition [27] has already shown the forward direction. Let $T = \langle Q, \Sigma, \Gamma, q_0, \rightarrow, \sigma \rangle$ be an onward SFST computing f that is k-TSSL on tier τ. Suppose $x, y \in \Sigma^*$ are such that $\mathrm{suff}^{k-1}(\tau(f^{\Leftarrow}(w))) = \mathrm{suff}^{k-1}(\tau(f^{\Leftarrow}(x)))$. Write $q_0 \xrightarrow{w:y} r$ and $q_0 \xrightarrow{x:z} s$. By Lemma [29], $r = \mathrm{suff}^{k-1}(\tau(f^{\Leftarrow}(w))) = \mathrm{suff}^{k-1}(\tau(f^{\Leftarrow}(x))) = s$, so $f_w^{\rightarrow} = f_x^{\rightarrow}$, thus f is k-TSSL on tier τ. $\qquad\square$