Multi-Input Strictly Local Functions for Tonal Phonology

Jonathan Rawski and Hossep Dolatian
Department of Linguistics
Institute for Advanced Computational Science
Stony Brook University
{jonathan.rawski,hossep.dolatian}@stonybrook.edu

Abstract

This paper presents an automata-theoretic characterization of the typology of attested tonal patterns using enriched data structures. We generalize the Input Strictly Local class of functions to consider multiple inputs of tonal and segmental strings, and find that the associated strictly local multi-tape transducers successfully capture tonal typology. Links between automata-theoretic and logical characterizations of phonological expressivity showcase tradeoffs in data structure and locality in the expressivity of phonological computation.

1 Introduction

Recent work in mathematical phonology connects phonological mappings to subclasses of the regular functions (McNaughton and Papert, 1971; Rogers and Pullum, 2011; Rogers et al., 2013; Heinz and Lai, 2013; Chandlee, 2014). One of the simplest subclasses is the class of Input Strictly Local (ISL) functions which take as input a single string *and* generate an output based on local information. Despite their reduced expressivity, ISL functions capture a majority of phonological and morphological maps (Chandlee, 2017; Chandlee and Heinz, 2018). In addition, ISL functions are provably easier and faster to learn than full regular functions (Chandlee et al., 2015a).

In this paper, we generalize this notion of locality from the above single-input functions to functions which take *multiple* strings as input in §2. Such functions are *Multi-Input Strictly Local* (MISL). MISL functions are effectively computed by a class of deterministic asynchronous Multi-tape Finite State Transducers (MT-FSTs). Natural language has processes which are understood in terms of enriched multi-string input structures, i.e. autosegmental structure. We focus on tone association §3.

The bulk of computational results on tonal patterns are defined over graphical structures and are *local* over *autosegmental* graphs (Jardine, 2016a,b, 2017a, 2019; Chandlee and Jardine, 2019a). In §4, we show that the bulk of tonal processes are MISL: they are local when computed as a multi-input function over strings. This provides a solution to a dichotomy in formal language results between the complexity of segmental vs tonal phonology (Jardine, 2016a) via enriching the data structure in a linguistically natural way. This also connects logically defined functions to automata-theoretic characterizations over enriched data structures.

Tonal processes is sufficiently computable using types of MT-FSTs, but we show that the full power is not necessary. Showing that the bulk of tonal phonology can be computed with *only* MISL MT-FSTs, acts as a stepping stone to determining the learnability of tone. It likewise acts as a benchmark to examine the typology of attested and unattested tonal processes. Furthermore, by using multi-input functions with MT-FSTs instead single-input functions with FSTs, we can more iconically compute the fact that 1) the tone tier is separate from the vowel tier, and that 2) this separation makes certain tonal processes be local.

We emphasize that our result is NOT an argument against the use of graphs in tone. The use of graphs iconically captures tonal processes. Any linear encoding of autosegmental structure, including ours, requires the use of special symbols for preassociation (Kornai, 1995; Wiebe, 1992; Yli-Jyrä, 2013, 2015).

Single-input functions are a special case of multi-input functions. With finite-state calculus, single-input functions correspond to rational functions when modeled by 1-way single-tape FSTs, and to regular functions when modeled by 2-

Proceedings of the Society for Computation in Linguistics (SCiL) 2020, pages 245-260.
New Orleans, Louisiana, January 2-5, 2020

way single-tape FSTs (Filiot and Reynier, 2016).[1] Multi-input functions are modeled by 1-way or 2-way MT-FSTs. Although there is work on the expressivity of multi-tape automata (Furia, 2012), little is known on multi-input *functions* and their algebra or expressivity (Frougny and Sakarovitch, 1993). We show that the MISL class characterizes a substantial chunk of tonal phonology.

2 Preliminaries

2.1 Preliminaries for single-input functions

Let $\rtimes$,$\ltimes$be the start and end boundaries respectively. Let Σ be a finite alphabet of symbols (excluding $\rtimes$,$\ltimes$). Let $\Sigma_{\rtimes} = \Sigma \cup \{\rtimes, \ltimes\}$. Let Σ^* the set of all strings over Σ. Let $|w|$ indicate the length of $w \in \Sigma^*$. For two strings w and v let wv be their concatenation, and for a set $L \subset \Sigma^*$ of strings and a string w, by wL we denote $\{wv | v \in L\}$. Let λ denote the empty string.

Given some string u and a natural number k, the *k-suffix* of u is the last k symbols of u: $\texttt{suff}(u, k) = v$ s.t. $|v| = k$ and $xv = u$ for some $x \in \Sigma^*$. For an alphabet Σ, the *k-factors* of Σ are the set of strings $w \in \Sigma*$ such that $|w| \leq k$.

Informally, a single-input function f is k-ISL if for all $u_1, u_2 \in \Sigma^*$, if $\texttt{suff}(u_1, k-1) = \texttt{suff}(u_2, k-1)$ then the two strings have the output extensions w.r.t f (Chandlee, 2014; Chandlee et al., 2015b). For any k-ISL function f over domain Σ^*, there exists a *canonical* deterministic single-tape finite-state transducer (1T-FST) M such that $|M| = f$ (meaning M computes f), and every state $q \in Q$ in M is labelled with one of the $k-1$ suffixes of Σ^*. Transitions are function tuples $\Delta : Q \times \Sigma \rightarrow Q \times \Gamma^*$. For a state $q \in Q$ and input symbol $a \in \Sigma$, $\delta(q, a) = (p, B)$ such that $B \in \Gamma^*$ and $p = \text{suff}(qa,)$.

2.2 Preliminaries for multi-input functions

We introduce notation for functions which take multiple strings as input. To do so, we use tuples demarcated by brackets. In the formalization here, we only consider functions which produce one output string, not a tuple of output strings. But extending the formalization is trivial; such a function is illustrated in §4.3.1.

[1]By single-tape FST, we mean a two-tape MTFST with one input tape and one output tape. Note that the functions computed by 1-way FSTs are called 'regular functions' in American computer science. In this paper, we follow French conventions which call this class the 'rational functions' (Filiot and Reynier, 2016).

A function f is an n-input function if it takes as input a tuple of n strings: $[w_1, \ldots, w_n]$, which we represent as $\vec{w}$, where each word w_i is made up of symbols from some alphabet Σ_i such that $w_i \in \Sigma_i^*$. Each alphabet Σ_i may be disjoint or intersecting, so two input strings w_i, w_j may be part of the same language Σ_i^*. These n alphabets form a tuple $\vec{\Sigma}$. Tuples can be concatenated: if $\vec{w} = [ab, c], \vec{x} = [d, ef]$, then $\vec{w}\vec{x} = [abd, cef]$.

To generalize the notion of suffixes into multiple strings, we define a tuple of n natural numbers as $\vec{k} = [k_1, \ldots, k_n]$. Given some tuple of n strings $\vec{w}$ and tuple of n numbers $\vec{k}$, $\vec{k}$*-suffix* of $\vec{w}$ is a tuple $\vec{v}$ of n strings v_i, made up of the last k_i symbols on w_i: $\texttt{suff}(\vec{w}, \vec{k}) = V$ s.t. $\vec{v} = [v_1, \ldots, v_n]$ and $|v_i| = k_i$ and $x_i v_i = w_i$ for $x_i \in \Sigma_i^*$. E.g. for $\vec{w}$=[abc,def] and $\vec{k} = [2, 1]$, $\texttt{suff}(\vec{w}, \vec{k}) = [bc, f]$. Given a tuple $\vec{k}$, the operation $\vec{k} - x$ subtracts x from each of k_i. E.g., for $\vec{k} = [2, 3, 6]$, $\vec{k} - 1 = [1, 2, 5]$. For a tuple of alphabets $\vec{\Sigma}$, the $\vec{k} - factors$ of $\vec{\Sigma}$ is the set of tuples $\vec{w} \in \vec{\Sigma}$ such that $|w_i| \leq k_i$.

Let f be an n-input function defined over an $n-$tuple $\vec{w}$ of input strings $\vec{w} = [w_1, \ldots, w_n]$ taken from the tuple of n alphabets $\vec{\Sigma}$. As an *informal* and intuitive abstraction from ISL functions, f is Multi-Input Strictly Local (MISL) for $k = [k_1, \ldots, k_n]$ if the function operates over a bounded window of size k_i for w_i. Formally,

Definition 1: A function f is $\vec{k}$-MISL iff there exists a deterministic asynchronous Multi-tape FST such that i) $|M| = f$, and ii) the MT-FST is canonically $\vec{k}$-MISL

We explain $\vec{k}$-MISL Multi-tape FSTs in the next section.

Note that Definition 1 is an automata-theoretic definition, meaning the expressivity is necessarily dependent on the machine. A language-theoretic definition of MISL functions, and connections to this class of multi-tape transducers, is in progress. While ISL FSTs and MISL MT-FSTs similarly encode the k-suffix information and the notion of common output in the state of the transducer, the use of common output extensions used in the ISL functions is not easily extendable to multi-input functions. In particular, there are non-subsequential n-input functions which are computable with MISL MT-FSTs.

For an ISL function, it does not matter if the input string is read left-to-right or right-to-left. But for an MISL function, it does. A function may be

left-to-right MISL but not right-to-left MISL. We leave out a proof but an illustration is given in §4.1.

2.3 Multi-tape finite-state transducers

Multi-input functions can be modeled by multi-tape FSTs (MT-FST). An MT-FST is conceptually the same as single-tape FSTs, but over *multiple* input tapes (Rabin and Scott, 1959; Elgot and Mezei, 1965; Fischer, 1965; Fischer and Rosenberg, 1968; Furia, 2012). MT-FSAs and MT-FSTs are equivalent, and single-tape FSTs correspond to an MT-FSA with two tapes.

Informally, a MT-FST reads n multiple input strings as n input tapes, and it writes on a single output tape. Each of the n input strings is drawn from its own alphabet Σ_i. The output string is taken from the output alphabet Γ. For an input tuple of n strings $\vec{w} = [w_1, \ldots, w_n] = [\sigma_{1,1} \ldots \sigma_{1,|w_1|}, \ldots, \sigma_{n,1} \ldots \sigma_{n,|w_n|}]$, the initial configuration is that the MT-FST is in the initial state q_0, the read head. The FST begins at the first position of each of the n input tapes $\sigma_{i,1}$, and the writing head of the FST is positioned at the beginning of an empty output tape. After the FST reads the symbol under the read head, three things occur: 1) the state changes; 2) the FST writes some string; 3) the read head may advance to the right (+1) or stay put (0) on different tapes: either move on all tapes, no tapes, or some subset of the tapes.

This process repeats until the read head "falls off" the end of each input tape. If for some input $\vec{w}$, the MT-FST falls off the right edge of the n input tapes when the FST is in an accepting state after writing u on the output tape, we say the MT-FST transduces, transforms, or maps, $\vec{w}$ to u or $f_T\vec{w} = u$.[2] Otherwise, the MT-FST is undefined at $\vec{w}$. We illustrate MT-FSTs in §4.

A $n-$MT-FST is a 6-tuple $(Q, \vec{\Sigma_{\rtimes}}, \Gamma, q_0, F, \Delta)$ where:

- $n \in \mathbb{N}$ is the number of input tapes
- Q is the set of states
- $\vec{\Sigma_{\rtimes}} = [\Sigma_{1\rtimes}, \ldots, \Sigma_{n\rtimes}]$ is a tuple of n input alphabets Σ_i which include the end boundaries $\Sigma_{i\rtimes}$
- Γ is the output alphabet
- $q_0 \in Q$ is the initial state
- $F \subset Q$ is the set of final states
- $\delta : Q \times \vec{\Sigma_{\rtimes}} \to Q \times \vec{D} \times \Gamma^*$ is the transition function where
  - $D = \{0, +1\}$ is the set of possible directions,[3]
  - $\vec{D} = [D^n]$ is an n-tuple of possible directions to take on each tape

The above definition can be generalized for MT-FSTs which use multiple output tapes. As parameters, an MT-FST can be deterministic or non-deterministic, synchronous or asynchronous. We only use *deterministic* MT-FSTs which are weaker than non-deterministic MT-FSTs. An MT-FST is synchronous if all the input tapes are advanced at the same time, otherwise it is asynchronous. We use asynchronous MT-FSTs which are more powerful than synchronous MT-FSTs. Synchronous MT-FSTs are equivalent to multi-track FSAs which are equivalent to single-tape FSAs, making them no more expressive than regular languages. For a survey of the properties of MT-FSAs and MT-FSTs, see Furia (2012).

A configuration c of a $n-$MT-FST M is an element of $(\vec{\Sigma_{\rtimes}}^* Q \vec{\Sigma_{\rtimes}}^* \times \Gamma^*)$, short for $([\Sigma_{1\rtimes}^* q \Sigma_{1\rtimes}^*, \ldots, \Sigma_{n\rtimes}^* q \Sigma_{n\rtimes}^*] \times \Gamma^*)$. The meaning of the configuration $c = ([w_1 q x_1, \ldots, w_n q x_n], u)$ is the following. The input to M is the tuple $\vec{w}\vec{x} = [w_1 x_1, \ldots, w_n x_n]$. The machine is currently in state q. The read head is on each of the n-input tapes on the first symbol of x_i (or has fallen off the right edge of the input tape if $x_i = \lambda$). u is currently written on the output tape.

Let the current configuration be $([w_1 q a_1 x_1, \ldots, w_n q a_n x_n], u)$ and let the current transition arc be $\delta(q, [a_1, \ldots, a_n]) = (r, \vec{D}, v)$. If $\vec{D} = [0^n]$, then the next configuration is $([w_1 r a_1 x_1, \ldots, w_n r a_n x_n], uv)$ in which case we write $([w_1 q a_1 x_1, \ldots, w_n q a_n x_n], u) \to ([w_1 r a_1 x_1, \ldots, w_n r a_n x_n], uv)$ (= none of the tapes are advanced) . If $\vec{D} = [+1^n]$, then the next configuration is $([w_1 a_1 r x_1, \ldots, w_n a_n r x_n], uv)$ in which case we write $([w_1 q a_1 x_1, \ldots, w_n q a_n x_n], u) \to ([w_1 a_1 r x_1, \ldots, w_n a_n r x_n], uv)$ (= all the tapes are advanced). Otherwise, the next configuration is $([w_i C_1 x_1 \ldots, w_n C_n x_n, \ldots], uv)$ where $C_i = r a_i$ if $D_i = 0$ and $C_i = a_i r$ if $D_i = +1$ in which case we write $([w_1 q a_1 x_1, \ldots, w_n q a_n x_n], u) \to ([w_i C_1 x_1 \ldots, w_n C x_n, \ldots], uv)$ (= a subset of the tapes are advanced).[4]

[2] If the MT-FST generates tuples instead of single strings, then the MT-ST maps $\vec{w}$ to $\vec{u}$.

[3] If the MT-FST reads from right to left, then it uses the -1 direction parameter

[4] Note that the interpretation of the third type of configuration subsumes the first two. We explicitly show the first two

The transitive closure of $\rightarrow$ is denoted with $\rightarrow^+$. Thus, if $c \rightarrow^+ c'$ then there exists a finite sequence of configurations $c_1, c_2 \ldots, c_n$ with $n > 1$ such that $c = c_1 \rightarrow c_2 \rightarrow \ldots \rightarrow c_n = c'$.

As for the function that a MT-FST M computes, for each $n-$tuple $\vec{w} \in \vec{\Sigma}^*$ where $\vec{w} = [w_1, \ldots, w_n]$, $f_M(\vec{w}) = u \in \Gamma^*$ (where $f_M = |M|$) provided there exists $q_f \in F$ such that $([q_0 \rtimes w_1 \ltimes, \ldots, q_0 \rtimes w_n \ltimes], \lambda) \rightarrow^+ ([\rtimes w_1 \ltimes q_f, \ldots, \rtimes w_n \ltimes q_f], u)$. Otherwise, if the configuration is $([\rtimes w_1 \ltimes q, \ldots, \rtimes w_n \ltimes q], u)$ and $q \notin F$ then the transducer crashes and the transduction f_T is undefined on input $\vec{w}$. Note that if a MT-FST is deterministic, it follows that if $f_T(\vec{w})$ is defined then u is unique.

As explained in §2.2, we define a function as $\vec{k}$-MISL iff there exists a corresponding deterministic asynchronous $\vec{k}$-MISL Multi-tape FST.

Definition 2: A deterministic asynchronous MT-FST M with alphabet $\vec{\Sigma}$ is a canonical MT-FST for an $\vec{k}$-MISL function f if the states of M are labelled with the $\vec{k} - 1$ suffixes of $\vec{\Sigma}$.

In Definition 2, the restriction on state labels does not apply to the unique initial state and unique final state. In other words, except for the initial and final states q_0 and q_f, every state corresponds to a possible $\vec{k} - 1$ *factor* of f

.

3 Computational phonology of tone

Segmental phonological processes are generally computed as single-input functions and they are ISL (Chandlee, 2014; Chandlee and Heinz, 2018). But when treated as a single-input function, tonal processes are significantly more complex than ISL (Jardine, 2016a). Single strings also fail to capture the suprasegmental nature of tone. Instead, tonal processes are generally modeled with *autosegmental representations* (ASR). As graphs, ASRs are a richer data structure that showcase the nonlinear nature of tone by breaking up a linear string into parallel strings or tiers (tone and vowel/mora).

As a review, consider the nonce words in Table 1. On the surface, the vowels each surface with some tone feature: high V́ vs. low V̀. A common analysis is that underlyingly the tones are on a separate tier from the vowels. A mapping function creates association arcs between the tones and vowels. In the input in Table 1a, then the tones and vowels are not underlying preassociated. Some tonal processes are analyzed with underlying preassociated tones (Table 1b). That is, the input contains an association arc between the some of the tones and some of the vowels.

Most mathematical results on tonal phonology are also defined over graphs or graph-like structures (Bird and Klein, 1990; Bird, 1995; Coleman and Local, 1991; Coleman, 1998). Jardine (2016a,b, 2017a) showed that computing well-formedness for tonal structures is Strictly Local over ASRs. For transformations, Chandlee and Jardine (2019a) define a class of logical functions over ASRs called Autosegmental Input-Strictly Local functions (A-ISL), which can model many but not all tonal mappings that have preassociation. Informally, a function is A-ISL if it consists of two ISL functions operating over two tiers or two separate strings.[5] Koser et al. (2019) showed that mapping ASRs without preassociation to ASRs with associations is likewise a local process, specifically with Quantifier-Free Least Fixed Point logic (QFLFP) (Chandlee and Jardine, 2019b). However, most of these results are defined logically (Jardine, 2017b, 2019), and do not clearly correspond to other algebraic or automata-theoretic notions.

Computationally, tonal processes have been modeled with single-tape FSTs (Bird and Ellison, 1994; Kornai, 1995; Yli-Jyrä, 2013, 2015), synchronous MT-FSTs (Kiraz, 2001), and non-deterministic asynchronous MT-FSTs (Kay, 1987; Wiebe, 1992). To our knowledge, the above mathematical properties of tone as a *graph* have not been linked with finite-state calculus. As a link, we treat tonal processes as a *multi-input* function that takes as input a tuple of two strings. With this definition, the bulk of tonal processes are MISL.

4 Multi-Input Locality in Tone

Table 2 illustrates all the tonal functions which we formalize. Items a-e are taken from Koser et al. (2019), and items f-l from Chandlee and Jardine (2019a). Throughout this section, we reference only this table; see the original references for more language information.

Items a-e are not ISL but are A-ISL.[6] In §4.1, we show they are also MISL. Items f-l have preassociated tone-vowel pairs in the input. In §4.2, we

for illustrative reasons.

[5] There are much more nuances to the definition of A-ISL; readers are referred to Chandlee and Jardine (2019a).

[6] Koser et al. (2019) formalize tonal functions without preassocation with Quantifier-Free Least Fixed Point logic.

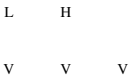
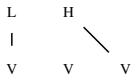
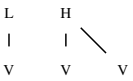
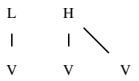
	a. Without underlying preassociation	b. With underlying preassocation
Input as string	*LH + patuki*	*patúki*
Input as graph		
Output as string	*pàtúkí*	*pàtúkí*
Output as graph		

Table 1: Review of tonal phonology.

show that with a specific linear encoding for preassociation, all the relatively simple ISL or A-ISL patterns are also MISL. More complex cases are handled in §4.3.

4.1 Tone without preassocation

4.1.1 General illustration: Mende spreading

We first illustrate with Mende (2a) which has a process of left-to-right tonal spread. Tones and vowels match 1-1 up until the last tone: *nìkílì* 'groundnut'. If there are more vowels than tones, then the final tone spreads: *félàmà* 'junction'.

As a function f, Mende left-to-right spreading is a 2-input function that takes as input a tuple of two strings: $\vec{w} = [w_1, w_2]$. The input string w_1 is a string of tones **T** taken from the input alphabet $\Sigma_1 = \Sigma_T = \{H, L\}$. The input string w_2 is a string of vowels **V** taken from the input alphabet $\Sigma_2 = \Sigma_V = \{V\}$. The input language is thus a tuple of two regular languages $[\Sigma_T^*, \Sigma_V^*]$. Each alphabet can include the start and end boundaries $\rtimes, \ltimes$: $\Sigma_{i\rtimes} = \Sigma_i \cup \{\rtimes, \ltimes\}$. The function generates a single output string of tonal vowels: $\Gamma = \{\acute{V}, \grave{V}\}$.

This 2-input function is MISL for $\vec{k} = [2, 1]$. It needs a locality window of size 2 on the **T**-string in order to know if some tone is final or not (i.e., if we see $H\ltimes$ or $L\ltimes$), and a locality window of size 1 on the **V**-string because the function only needs to know the current vowel.

This function is computed by the deterministic asynchronous MT-FST in Figure (1). It uses two input tapes: a tone tape **T** and a vowel tape **V**. The MT-FST has a dedicated initial and final state q_0 and q_f. All other states are labelled with the $\vec{k} - 1$-factors separated by commas. Transitions have the template $[\Sigma_1, \Sigma_2, \ldots, \Sigma_n]$: $[D^n] : \Gamma^*$ where Σ_i

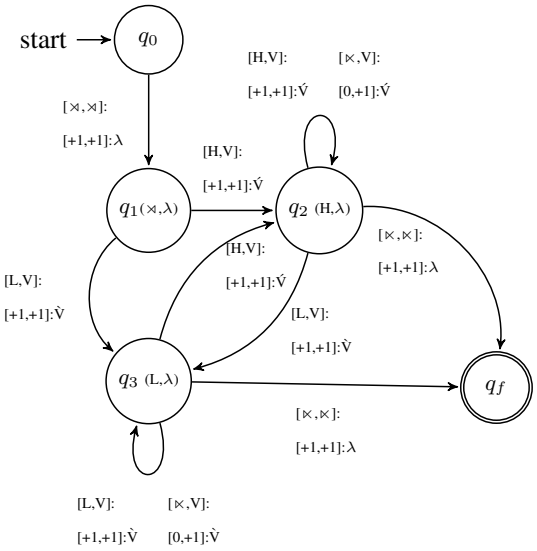


Figure 1: MT-FST for Mende

marks the read input symbols on the input string w_i, and where D is a possible direction parameter from {0,+1}. Given a parameter D_i, the transition arc dictates whether the MT-FST will advance (+1) or stay put (0) on the input tape w_i.

A sample derivation for /HL + felama/ is in Table 3. Each row keeps track of the: i) current state, ii) location of the read head on the input tapes, iii) transition arc used on each input tape, iv) outputted symbol, v) current output string. At step 5, upon reading ⋉on the **T**-tape, asynchrony allows the read-head to advance on the **V**-tape but not on the **V**-tape, capturing the spreading effect.

4.1.2 Other processes without preassociation

Data in this section is illustrated in Table 2b-e and collected from Koser et al. (2019) who showed that they are are local in that they are QFLFP. We

Table 2: Sample of tonal processes, example input-output structures, and computational complexity.
Legend: * Function was proved to be QFLFP by Koser et al. (2019), ** Function is MISL if the output is 2-tuple

	Language	Process	Pre-ass?	ISL	A-ISL	MISL	$\vec{k}$-value
a	Mende	Iterative left-right spread /LH + VVV/ → [V̀V́V́] 	✗		✓*	✓	[2,1]
b	Kikuyu	Initial spread to two + final spread /LHLH + VVVVVVV/ → [V̀V́V́V́V́V́V́] 	✗		✓*	✓	[2,3]
c	Hausa	Iterative right-left spread /LH + VVV/ → [V̀V̀V́] 	✗		✓*	✓	[2,1]
d	Northern Shona	Edge-in + initial spread + medial spread /HLH + VVVVVV/ → [V́V́V́V̀V̀V́] 	✗		✓*	✓	[4,6]
e	Kukuya	Quantity sensitive spreading /H + VVVV/ → [V́V́V́V́] 	✗		✓*	✓	[4,2]
f	Rimi	Bounded tone shift /VV́VV/ → [VVV́V] /⟨H̲⟩ + V⟨V̲⟩VV/ 	✓	✓	✓	✓	[1,2]
g	Zigula	Unbounded tone shift /VVV́VVV/ → [VVVVV́V] /⟨H̲⟩ + V⟨V̲⟩VV/ 	✓	✗	✓	✓	[1,3]
h	Bemba	Bounded tone spread /VV́VV/ → [VV́V́V] /⟨H̲⟩ + V⟨V̲⟩VV/ 	✓	✓	✓	✓	[1,2]
i	Arusa	Unbounded deletion /V́ VV́V/ → [V́VVVV] /⟨H̲⟩ ⟨H̲⟩ + ⟨V̲⟩ V(V̲V̲)V/ 	✓	✗	✓	✓	[3,1]
j	Luganda	Bounded Meussen's rule /V́V́V/ → [V́V̀VV] /⟨H̲⟩ ⟨H̲⟩ + ⟨V̲⟩(V̲V̲)V/ 	✓	✓	✗	✓	[2,2]**
k	Shona	Alternating Meussen's rule /V́-V́-V́/ → [V́-V̀-V́] /⟨H̲⟩-⟨H̲⟩-⟨H̲⟩ + ⟨V̲⟩-⟨V̲⟩-⟨V̲⟩/ 	✓	✗	✗	✗	
l	Ndebele	Unbounded spreading to ante-penultimate /V́VVVV/ → [V́V́VVV] /⟨H̲⟩ + ⟨V̲⟩VVVV/ 	✓	✗	✗	✓	[1,3]

	Current state	Tone tape		Vowel tape		Output symbol	Output string
1.	q_0	$\underline{\rtimes}$HL$\ltimes$		$\underline{\rtimes}$eaa$\ltimes$			
2.	q_1	$\rtimes\underline{\text{H}}L\ltimes$	$\rtimes$:+1	$\rtimes\underline{\text{e}}aa\ltimes$	$\rtimes$:+1	λ	
3.	q_2	$\rtimes$H$\underline{\text{L}}\ltimes$	H:+1	$\rtimes$e$\underline{\text{a}}$a$\ltimes$	e:+1	*é*	*é*
4.	q_3	$\rtimes$HL$\underline{\ltimes}$	L:+1	$\rtimes$ea$\underline{\text{a}}\ltimes$	a:+1	*à*	*éà*
5.	q_3	$\rtimes$HL$\underline{\ltimes}$	$\ltimes$:0	$\rtimes$eaa$\underline{\ltimes}$	a:+1	*à*	*éàà*
6.	q_f	$\rtimes$HL$\ltimes$	$\ltimes$:+1	$\rtimes$eaa$\ltimes$	$\ltimes$:+1	λ	*éàà*

Table 3: Derivation of *HL* + *felama* over its tone-vowel tiers *HL* + *eaa* with the MT-FST in Figure 1

show that they are all MISL. Example MT-FSTs and derivations for cases b,c are in the appendix.

Kikuyu has a process of spreading an initial tone up to first two vowels (2b). The remaining tones and vowels are associated 1-to-1. If there are more vowels than tones, the final tone is spread: /LHLH + VVVVVVV/ → [V̀V̀ V́ V̀ V́V́V́]. Initial spreading up to two vowels is [2,3]-MISL because the function requires the context [⋊L,⋊VV] in order to spread L to the first two vowels. Final spread is [2,1]-MISL as in Mende (§4.1.1). Together, Kikuya is [2,3]-MISL.

Hausa (2c) behaves analogously to Mende but tones are associated *right-to-left* with *initial*-spreading: /LH + VVV/ → [V̀V̀ V́]. This is [2,1]-MISL *when* the input string is read right-to-left.

North Karanga Shona is more complex (2d). The initial and final tones are associated to the first and last vowels respectively. The first tone can spread up until the first 3 vowels *but not* to the penultimate vowel. The medial tone can spread up until the penultimate vowel: /HLH + VVVVVV/ → [V́V́V́ V̀V̀ V́]. The process is MISL but for a very large locality window of [4,6]. The window may be larger or smaller depending on various complications discussed in Koser et al. (2019).

Lastly, Kukuya (Table 2e) allows a H tone to spread if it is the only tone: /H + VVV/ → [V́V́V́]. Otherwise, if the input is HL, the L tone spreads: /HL + VVV/ → [V́ V̀V̀]. If LH, the L spreads up until the penultimate vowel: /LH + VVV/ → [V̀V̀ V́]. This is at most [4,2]-MISL: 4 over the **T**-tape in order to check if it's H, HL, or LH; 2 over the **V**-tape to prevent an L from spreading to the final vowel if the input tone is LH.[7]

[7]If the input tone is LHL, (Koser et al., 2019) do not state if either L can ever show spreading in words of four or more vowels. If they can, this is also MISL.

4.1.3 Contour tones

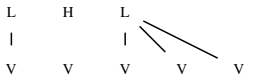
In §4.1, we assumed that the input had at least as many vowels as tones. If the input has more tones than vowels, final contour tones can be made: /HL + V/ → [V̂]. Assume that the number of possible contour tones is finite and modeled with a finite number of characters: rising V̌, falling V̂. To generate contour tones, one *compositional* approach is to first generate 1-to-1 or 1-to-many tone-vowel associations without any contour symbols; if there are more tones than vowels, then the unassigned tones are outputted at the end of the output string: /HL + V/ → //V́ L//. The string is then fed to an ISL function which changes strings of tonal vowels and tones into contour tones: //V́ L// → [V̂]. A non-compositional approach is mapping unassociated tones-and-vowels to the output through a single function. We conjecture that this function would be MISL as long as there are no long-distance dependencies involved in creating a contour tone. For easier illustration, we assume a compositional approach.

4.2 Tone with preassociation

4.2.1 Encoding preassociation

Tonal processes may include inputs where a tone is *preassociated* to one or more vowels. This dependency between the two strings is a reason why graphical structures are useful representations for tone, but it is a reason why many linear encodings require some special markup system (Kornai, 1995). For our purposes, we use the following encoding in Figure 2, inspired from an encoding system used by Yli-Jyrä (2013, 2015). We do not use other proposed encoding systems (Wiebe, 1992; Kornai, 1995; Yli-Jyrä, 2013, 2015) because they are either designed for single-tape FSTs or do not maintain strict locality.

If a tone T or single vowel V is preassociated, it is underlined and demarcated with angle brackets: $\langle\underline{\text{T}}\rangle$, $\langle\underline{\text{V}}\rangle$. If a span of multiple vowels are



Graph Encoding $\langle\underline{\text{L}}\rangle\text{H}\langle\underline{\text{L}}\rangle + \langle\underline{\text{V}}\rangle\text{V}(\underline{\text{VVV}})$

Figure 2: Encoding preassociation

associated to the same tone, they are marked with parentheses instead of angle brackets: ($\underline{\text{V}}$ $\underline{\text{V}}$ ... $\underline{\text{V}}$). This encoding creates the following enriched input alphabets of multi-character units:

- Σ_T ={ H, L, $\langle\underline{\text{H}}\rangle$, $\langle\underline{\text{L}}\rangle$}
- Σ_V ={ V, $\langle\underline{\text{V}}\rangle$, ($\underline{\text{V}}$, $\underline{\text{V}}$, $\underline{\text{V}}$) }[8]

Other possible configurations, such as word-medial contour tones require a more elaborate encoding which we do not discuss. We set these aside because the preassociation data in Chandlee and Jardine (2019a) did not have such case studies.[9] We set aside the evaluation of our encoding mechanism based on Kornai (1995)'s desirada.

4.2.2 Locality of preassociated tones

With the above encoding, the tone functions in Table 2f-i with preassociation are MISL. Example MT-FSTs and derivations are in the appendix.

In Rimi (2f), a process of bounded tone shift will cause a preassociated tone to delink from its vowel and associate with the subsequent vowel: /VV́VV/→ [VVV́V]. In our encoding, the input is /$\langle\underline{\text{H}}\rangle$ + V$\langle\underline{\text{V}}\rangle$VV/. This function is ISL, MISL, *and* [1,2]-MISL. We need a locality window of size 1 over the **T**-string because we care if the current tone symbol is a preassociated $\langle\underline{\text{H}}\rangle$. If yes, then we need a locality window of size 2 over the **V**-string in order to delink the current preassociated vowel $\langle\underline{\text{V}}\rangle$ and associate the tone with the next vowel.

Unlike Rimi, Zigula displayed unbounded tone shift (2g) whereby a preassociated H is delinked from its preassociated vowel and associated with the *penultimate* vowel which can be at any distance away from the underlyingly preassociated vowel: / VVV́VVV/ or /$\langle\underline{\text{H}}\rangle$ + VV$\langle\underline{\text{V}}\rangle$VVV/ → [VVVVV́V]. This function isn't ISL but it is A-ISL *and* [1,3]-MISL. Given a preassociated $\langle\underline{\text{H}}\rangle$ as a current input tone symbol, an underlying preassociated vowel $\langle\underline{\text{V}}\rangle$ is delinked regardless of context, while current tone symbol $\langle\underline{\text{H}}\rangle$ is associated with the penultimate vowel. This requires a window of size 3 on the vowel string to check if the current vowel is the penultimate vowel.

Similar to Rimi, Bemba (2h) shows bounded tone spread whereby a preassociated tone-vowel pair is not delinked but the next vowel also becomes associated to the tone: / VV́VV/ or /$\langle\underline{\text{H}}\rangle$ + V$\langle\underline{\text{V}}\rangle$VV/ → [VV́V́V]. This is ISL, A-ISL, and [1,2]-MISL. The only difference from Rimi is that an input preassociated vowel $\langle\underline{\text{V}}\rangle$ is not delinked, i.e. it keeps its tone in the output.

In Arusa (2i), a process of unbounded deletion deletes a phrase-final H tone if it is follows another H tone. By deleting the H tone, any preassociated vowels become delinked and toneless: /V́ VV́V́V/ or /$\langle\underline{\text{H}}\rangle$ $\langle\underline{\text{H}}\rangle$ + $\langle\underline{\text{V}}\rangle$ V($\underline{\text{VV}}$)V/ → [V́ VVVV]. This process is not ISL because of the unbounded distance between the two spans of high vowels, but it is A-ISL and [3,1]-MISL.[10] A locality window of size 3 is needed on the **T**-string in order to check if the current input tone symbol is a phrase-final $\langle\underline{\text{H}}\rangle$ and succeeds another high tone. If yes, then any currently read input vowels are delinked.

4.3 Distinct functions across locality classes

The distinctions between ISL, A-ISL, and MISL are visible in more complex patterns in Table 2j-l. So far, all the A-ISL and ISL functions we described were also MISL. But some ISL yet non-A-ISL functions are *variably* MISL depending on how the function is defined. They are MISL *only* if the function generates as output two output strings of associated tones vs. associated vowels instead of only one output string (§4.3.1). Furthermore, some patterns are neither ISL, A-ISL, or MISL (§4.3.2). And finally, some patterns are MISL but neither ISL nor A-ISL (§4.3.3).

4.3.1 ISL but not A-ISL; variably MISL

Luganda (2j) has a process of bounded Meussen's rule which is ISL but not A-ISL. Here, if a preas-

[8] Note that ($\underline{\text{V}}$, $\underline{\text{V}}$, and $\underline{\text{V}}$) are three separate input alphabet symbols.

[9] One possible system, inspired from Yli-Jyrä (2015), is to use the symbols / and \on the vowel-string. Given a tuple of [$\langle\underline{\text{H}}\rangle\langle\underline{\text{L}}\rangle$, ($\underline{\text{V}}$/ $\underline{\text{V}}$)] where space marks the separation of multicharacter symbols, the slash / means that the first tone is associated to the first vowel while the second tone to the two vowels. Similarly for [$\langle\underline{\text{H}}\rangle\langle\underline{\text{L}}\rangle$, ($\underline{\text{V}}$ \$\underline{\text{V}}$)], the first tone is associated with the two vowels while the second tone with the second vowel.

[10] The FST in the appendix is [3,1]-MISL but it cannot ensure that the number of preassociated tones in the input match the number of spans of preassociated vowels. Doing so requires that we either increase the locality window on the vowel tape to 2, *or* we output a string tuple such that the function changes the substring $\langle\underline{\text{H}}\rangle\langle\underline{\text{H}}\rangle\ltimes$to $\langle\underline{\text{H}}\rangle\langle\underline{\text{L}}\rangle\ltimes$, similarly to the Luganda case in §4.3.1.

sociated H tone precedes another preassociated H tone *and* the two tones are associated to a contiguous sequence of vowels, then the second H tone becomes low: /V́V́V́V/ or /⟨H⟩⟨H⟩ + ⟨V⟩(VV)V/ → [V́VV̀V]. The function is not A-ISL because it needs to reference contiguity on both the tone and vowel strings, see Chandlee and Jardine (2019a) on why this matters.

Similarly, if the function is defined as a multi-input function which generates only *one* output string, then the function is not MISL. Assume the **T**-string is ⟨H⟩⟨H⟩, and the **V**-string contains two vowels preassociated to the two different tones which we *represent* with butting brackets: /⟨H⟩⟨H⟩ + ⟨V⟩(VV...V)/. The second vowel (V will map to a surface low toned vowel V̀ because the two tones are contiguous. The second vowel (V starts a span of preassociated vowels. But for the other vowels like the final V), an MISL function cannot keep track if this vowel was part of a preassociated vowel span which succeeded another span, i.e. it can't know if V) is preceded by the substring ⟨V⟩ (V or not.

But if the function generates as output *two* output strings as an output tuple of tones and vowels, then the function is [2,2]-MISL. The input /⟨H⟩⟨H⟩ + ⟨V⟩(V V V)/ is mapped to [⟨H⟩⟨L⟩ + ⟨V⟩(V V V)] with the only change being on the **T**-string. The function is [2,2]-MISL because it checks if i) the current tone symbol is a preassociated ⟨H⟩ and immediately succeeds another tone symbol ⟨H⟩ *and* if ii) the current vowel symbol is preassociated ⟨V⟩ or starts a span of preassociated vowels (V, and follows a span of preassociated vowels ⟨V⟩ or V). All this information is local with a window of 2 on the two strings.

4.3.2 Neither ISL, A-ISL, nor MISL

Shona (2k) has a process of Alternating Meussen's rule where hetero-morphemic and contiguous spans of preassociated high-toned vowels alternate to form high and low sequences: /V́-V́-V́/ → [V́-V̀-V́]. This is not ISL, A-ISL, or MISL because iterative alternation is local over output information, not input information. This is explained further in Chandlee and Jardine (2019a).

4.3.3 MISL but neither ISL nor A-ISL

Finally, Ndebele (2l) has unbounded spreading of a preassociated H tone up until the ante-penultimate vowel: /V́VVVV/ or /⟨H⟩ + ⟨V⟩VVVV/ → [V́V́VVV]. This process is neither ISL nor A-ISL but it is [1,3]-MISL. Reading from right-to-left, the last two vowels surface as toneless. But if the current tone symbol is a preassociated ⟨H⟩, then any vowel which is not the penultimate or ultimate surfaces as high V́. This requires a window of size 3 on the **V**-tape, but only 1 on the tone tape.

5 Conclusion

This paper examined the computational expressivity of autosegmental phonology, in particular tonal processes. Generalizing Input Strictly Local (ISL) functions to handle multiple inputs, we showed that the class of Multi-Input Strictly Local (MISL) functions can compute almost all attested tonal processes. These MISL functions are computed by restricted deterministic asynchronous multi-tape finite-state transducers. Using a careful linear encoding mechanism, this computational result applies equally well to tonal processes with or without preassociation. The result also narrows the gap in mathematical results between segmental and autosegmental phonology.

References

Steven Bird. 1995. Computational phonology: a constraint-based approach. Studies in Natural Language Processing. Cambridge University Press, Cambridge.

Steven Bird and T Mark Ellison. 1994. One-level phonology: Autosegmental representations and rules as finite automata. Computational Linguistics, 20(1):55–90.

Steven Bird and Ewan Klein. 1990. Phonological events. Journal of linguistics, 26(1):33–56.

Jane Chandlee. 2014. Strictly Local Phonological Processes. Ph.D. thesis, University of Delaware, Newark, DE.

Jane Chandlee. 2017. Computational locality in morphological maps. Morphology, pages 1–43.

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

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

Jane Chandlee and Jeffrey Heinz. 2018. Strict locality and phonological maps. Linguistic Inquiry, 49(1):23–60.

Jane Chandlee and Adam Jardine. 2019a. Autosegmental input strictly local functions. Transactions of the Association for Computational Linguistics, 7:157–168.

Jane Chandlee and Adam Jardine. 2019b. Quantifier-free least fixed point functions for phonology. In Proceedings of the 16th Meeting on the Mathematics of Language (MoL 16), Toronto, Canada. Association for Computational Linguistics.

John Coleman. 1998. Phonological representations: their names, forms and powers. Cambridge University Press.

John Coleman and John Local. 1991. The no crossing constraint in autosegmental phonology. Linguistics and Philosophy, 14(3):295–338.

C. C. Elgot and J. E. Mezei. 1965. On relations defined by generalized finite automata. IBM Journal of Research and Development, 9(1):47–68.

Emmanuel Filiot and Pierre-Alain Reynier. 2016. Transducers, logic and algebra for functions of finite words. ACM SIGLOG News, 3(3):4–19.

Patrick C Fischer. 1965. Multi-tape and infinite-state automataa survey. Communications of the ACM, 8(12):799–805.

Patrick C Fischer and Arnold L Rosenberg. 1968. Multitape one-way nonwriting automata. Journal of Computer and System Sciences, 2(1):88–101.

Christiane Frougny and Jacques Sakarovitch. 1993. Synchronized rational relations of finite and infinite words. Theoretical Computer Science, 108(1):45–82.

Carlo A. Furia. 2012. A survey of multi-tape automata. http://arxiv.org/abs/1205.0178. Latest revision: November 2013.

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.

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

Adam Jardine. 2016b. Locality and non-linear representations in tonal phonology. Ph.D. thesis, University of Delaware, Newark, DE.

Adam Jardine. 2017a. The local nature of tone-association patterns. Phonology, 34(2):363–384.

Adam Jardine. 2017b. On the logical complexity of autosegmental representations. In Proceedings of the 15th Meeting on the Mathematics of Language, pages 22–35.

Adam Jardine. 2019. The expressivity of autosegmental grammars. Journal of Logic, Language and Information, 28(1):9–54.

Martin Kay. 1987. Nonconcatenative finite-state morphology. In Third Conference of the European Chapter of the Association for Computational Linguistics.

George Anton Kiraz. 2001. Computational nonlinear morphology: with emphasis on Semitic languages. Cambridge University Press.

Andras Kornai. 1995. Formal phonology. Garland Publishing Inc.

Nathan Koser, Christopher Oakden, and Adam Jardine. 2019. Tone association and output locality in nonlinear structures. In Supplemental proceedings of AMP 2019.

Robert McNaughton and Seymour A Papert. 1971. Counter-Free Automata (MIT research monograph no. 65). The MIT Press.

Michael O Rabin and Dana Scott. 1959. Finite automata and their decision problems. IBM journal of research and development, 3(2):114–125.

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

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

Bruce Wiebe. 1992. Modelling autosegmental phonology with multi-tape finite state transducers. Master's thesis, Simon Fraser University.

Anssi Yli-Jyrä. 2013. On finite-state tonology with autosegmental representations. In Proceedings of the 11th international conference on finite state methods and natural language processing. Association for Computational Linguistics.

Anssi Yli-Jyrä. 2015. Three equivalent codes for autosegmental representations. In Proceedings of the 12th International Conference on Finite-State Methods and Natural Language Processing 2015 (FSMNLP 2015 Düsseldorf).

A Appendix

A sample MT-FST and derivation are given for some of the tone processes.

A.1 Tonal processes without preassociation

These patterns take as input a pair of strings without preassociation.

A.1.1 Kikuyu spreading

In Kikuyu (Table 2b), the first tone associates with the first two vowels. 1-to-1 association follows. A final tone may undergo final spreading, e.g. f([LHLH, VVVVVVV])=*ÈÈÉÉÉÉ*. A [2,3]-MISL MT-FST is provided in Figure 3, with a sample derivation in Table 4.

A.1.2 Hausa right-to-left spreading

In Hausa (Table 2b), tones are associated right-to-left with initial spread, e.g. f([LH, VVV])=*ÈÈÉ*. This function is modeled by the [2,1]-MISL MT-FST in Figure 4, with a sample derivation in Table 4. The FST processes the input string-tuple from right to left using the -1 direction parameter.

A.2 Tonal processes with preassociation

These functions take as input a preassociated pair of tones and vowels.

A.2.1 Rimi bounded tone shift

In Rimi (Table 2f), a preassociated tone will shift one vowel to the right, e.g. f([$\langle\underline{\text{H}}\rangle$, V$\langle\underline{\text{V}}\rangle$VV]=*VVÉV*. This function is modeled by the [1,2]-MISL MT-FST in Figure 5, with a sample derivation in Table 6. We assume that the only possible underlying tone string is a preassociated H.

Final preassociated vowels do not undergo tone shift: f([$\langle\underline{\text{H}}\rangle$, VVV$\langle\underline{\text{V}}\rangle$]=*VVVÉ*. We factor this out for illustrative reasons. Otherwise, the function is [2,2]-MISL and needs a MT-FST with more states.

A.2.2 Zigulu unbounded tone shift

In Zigulu (Table 2g), unbounded tone shift causes a preassociated H tone to shift to the penultimate vowel, e.g. f([$\langle\underline{\text{H}}\rangle$, VV$\langle\underline{\text{V}}\rangle$VVV])=*VVVVÉV*. This function is modeled by the [1,3]-MISL MT-FST in Figure 6, with a sample derivation in 7. For easier illustration, the MT-FST processes the input right-to-left using the -1 direction parameter. We assume that the tone string can either be an empty string $\rtimes\lambda\ltimes$ or a single preassociated H tone $\rtimes\langle\underline{H}\rangle\ltimes$.

A.2.3 Bemba unbounded tone spread

In Bemba (Table 2h), bounded tone spread causes a preassociated H tone to surface on its preassociated vowel and on the subsequent vowel, e.g. f([$\langle\underline{\text{H}}\rangle$,V$\langle\underline{\text{V}}\rangle$VV])=*VÉÉV*. This function is modeled by the [1,2]-MISL MT-FST in Figure 7, with a sample derivation in Table 8. We assume that the input tone string contains either an empty string $\rtimes\lambda\ltimes$ or a single preassociated H tone $\rtimes\langle\underline{H}\rangle\ltimes$.

A.2.4 Arusa unbounded deletion

In Aursa (Table 2i), unbounded deletion causes a phrase-final preassociated H to delete if it follows another H tone, e.g. f([$\langle\underline{\text{H}}\rangle\langle\underline{\text{H}}\rangle$,$\langle\underline{\text{V}}$ V(V$\underline{\text{VV}}$V]=*É VVVV*. This function is computed by the [3,1]-MISL MT-FST in Figure 8, with a sample derivation in 9. The FST reads the input from right-to-left using the -1 direction parameter. We assume the input tone string contains zero or more preassociated H tones: **T**=$\rtimes\langle\underline{\text{H}}\rangle^*\ltimes$.

As a caveat, the function in Figure () cannot ensure that the number of preassociated tones matches the number of spans of preassociated vowels. That more faithful function is [3,2]-MISL. We do not draw it here because of size.

For clarity, in Table 9, preassociated vowels are given a subscript $_1$ instead of underlining.

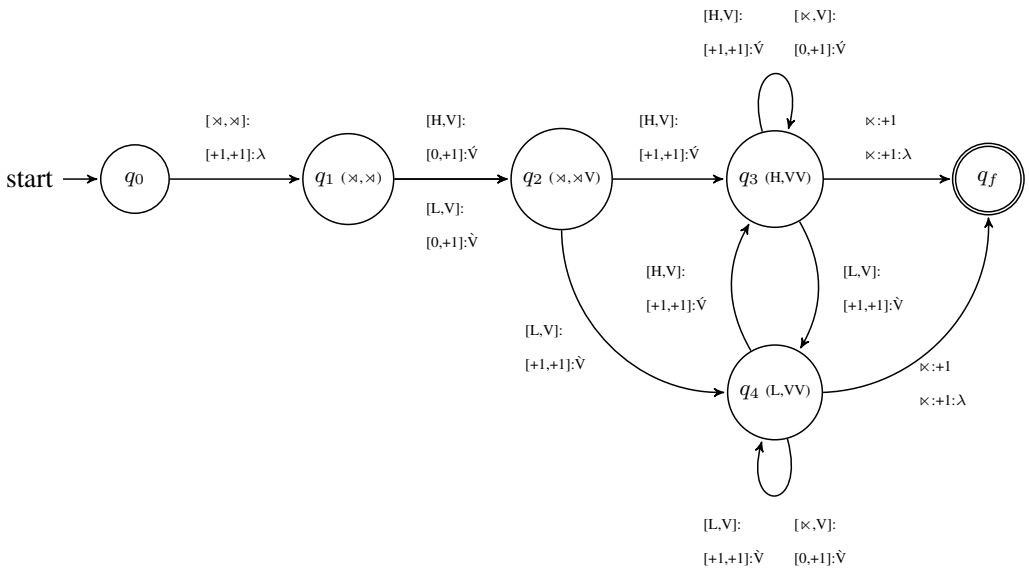


Figure 3: MT-FST for Kikuyu initial spread

	Current state	Tone tape		Vowel tape		Output symbol	Output string
1.	q_0	⋊LHLH⋉		⋊VVVVVVV⋉			
2.	q_1	⋊LHLH⋉	⋊:+1	⋊VVVVVVV⋉	⋊:+1	λ	
3.	q_2	⋊LHLH⋉	L:0	⋊VVVVVVV⋉	V:+1	V̀	V̀
4.	q_4	⋊LHLH⋉	L:+1	⋊VVVVVVV⋉	V:+1	V̀	V̀V̀
5.	q_3	⋊LHLH⋉	H:+1	⋊VVVVVVV⋉	V:+1	V́	V̀V̀V́
6.	q_4	⋊LHLH⋉	L:+1	⋊VVVVVVV⋉	V:+1	V̀	V̀V̀V́V̀
7.	q_3	⋊LHLH⋉	H:+1	⋊VVVVVVV⋉	V:+1	V́	V̀V̀V́V̀V́
8.	q_3	⋊LHLH⋉	⋉:0	⋊VVVVVVV⋉	V:+1	V́	V̀V̀V́V̀V́V́
9.	q_3	⋊LHLH⋉	⋉:0	⋊VVVVVVV⋉	V:+1	V́	V̀V̀V́V̀V́V́V́
10.	q_f	⋊LHLH⋉	⋉:+1	⋊VVVVVVV⋉	⋉:+1	λ	V̀V̀V́V̀V́V́V́

Table 4: Derivation of f([LHLH, VVVVVVV])=*V̀V̀V́V̀V́V́V́* in Kikuyu with the MT-FST in Figure 3

	Current state	Tone tape		Vowel tape		Output symbol	Output string
1.	q_0	⋊LH⋉		⋊VVV⋉			
2.	q_1	⋊LH⋉	⋉:-1	⋊VVV⋉	⋉:-1	λ	
3.	q_2	⋊LH⋉	H:-1	⋊VVV⋉	V:-1	V́	V́
4.	q_3	⋊LH⋉	L:-1	⋊VVV⋉	V:-1	V̀	V̀V́
5.	q_3	⋊LH⋉	⋊:0	⋊VVV⋉	V:-1	V̀	V̀V̀V́
6.	q_f	⋊LH⋉	⋊:-1	⋊VVV⋉	⋊:-1	λ	V̀V̀V́

Table 5: Derivation of f([LH, VVV])=*V̀V̀V́* in Hausa with the MT-FST in Figure 4

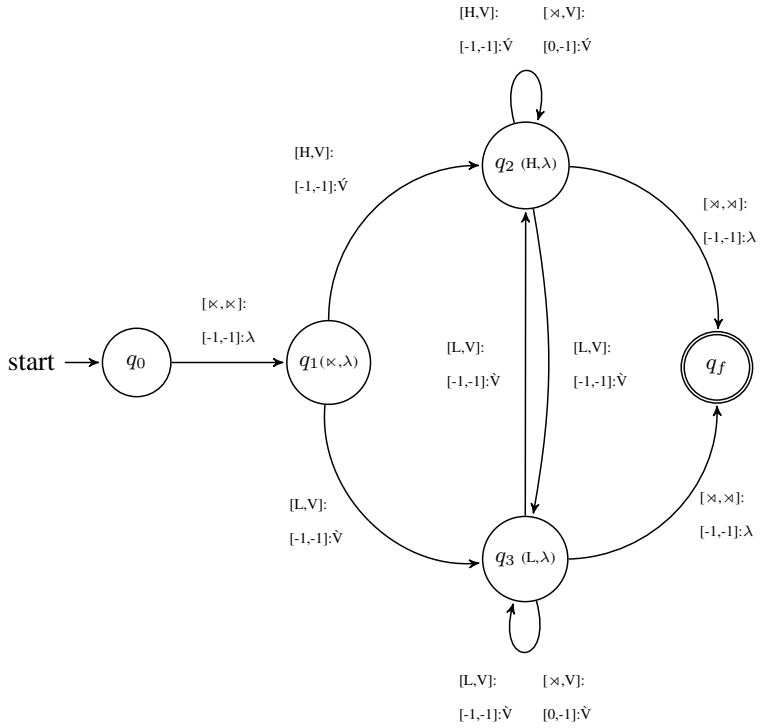


Figure 4: MT-FST for Hausa

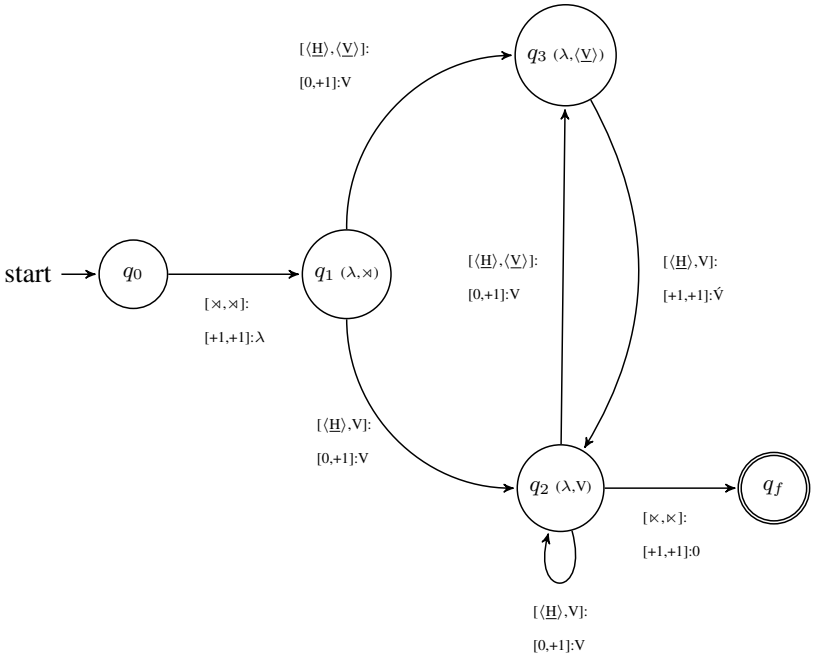


Figure 5: MT-FST for Rimi

	Current state	Tone tape		Vowel tape		Output symbol	Output string
1.	q_0	$\underline{\rtimes}\langle \underline{\text{H}} \rangle \ltimes$		$\underline{\rtimes}\text{V}\langle \underline{\text{V}} \rangle \text{VV} \ltimes$			
2.	q_1	$\rtimes \langle \underline{\text{H}} \rangle \ltimes$	$\rtimes$:+1	$\rtimes \underline{\text{V}} \langle \underline{\text{V}} \rangle \text{VV} \ltimes$	$\rtimes$:+1	λ	
3.	q_2	$\rtimes \langle \underline{\text{H}} \rangle \ltimes$	$\langle H \rangle$:0	$\rtimes \text{V} \langle \underline{\text{V}} \rangle \text{VV} \ltimes$	V:+1	V	V
4.	q_3	$\rtimes \langle \underline{\text{H}} \rangle \ltimes$	$\langle H \rangle$:0	$\rtimes \text{V} \langle \text{V} \rangle \underline{\text{V}} \text{V} \ltimes$	$\langle V \rangle$:+1	V	VV
5.	q_2	$\rtimes \langle \text{H} \rangle \underline{\ltimes}$	$\langle H \rangle$:+1	$\rtimes \text{V} \langle \text{V} \rangle \text{V} \underline{\text{V}} \ltimes$	V:+1	V́	VV́
6.	q_2	$\rtimes \langle \text{H} \rangle \underline{\ltimes}$	$\ltimes$:0	$\rtimes \text{V} \langle \text{V} \rangle \text{VV} \underline{\ltimes}$	V:+1	V	VV́V
7.	q_f	$\rtimes \langle \text{H} \rangle \ltimes$	$\ltimes$:+1	$\rtimes \text{V} \langle \text{V} \rangle \text{VV} \ltimes$	$\ltimes$:+1	λ	VV́V

Table 6: Derivation of $f([\langle \underline{\text{H}} \rangle, \text{V}\langle \underline{\text{V}} \rangle \text{VV}]$=*VV́V* in Rimi with the MT-FST in Figure 5

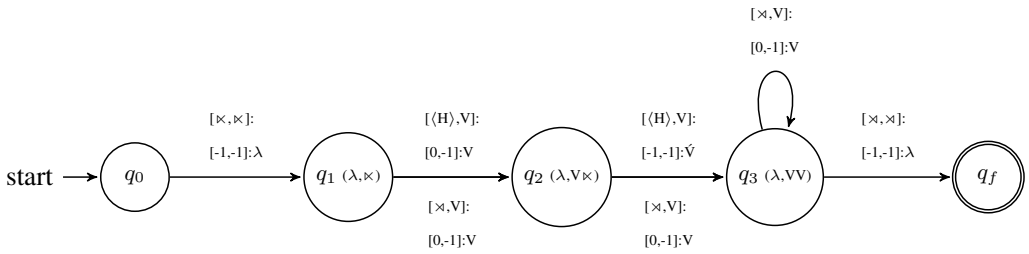


Figure 6: MT-FST for Zigulu

	Current state	Tone tape		Vowel tape		Output symbol	Output string
1.	q_0	$\rtimes \langle \underline{\text{H}} \rangle \underline{\ltimes}$		$\rtimes \text{VVVVVV} \underline{\ltimes}$			
2.	q_1	$\rtimes \langle \underline{\text{H}} \rangle \ltimes$	$\ltimes$:-1	$\rtimes \text{VVVVV} \underline{\text{V}} \ltimes$	$\ltimes$:-1	λ	
3.	q_2	$\rtimes \langle \underline{\text{H}} \rangle \ltimes$	$\langle H \rangle$:0	$\rtimes \text{VVVV} \underline{\text{V}} \text{V} \ltimes$	V:-1	V	V
4.	q_3	$\underline{\rtimes} \langle \text{H} \rangle \ltimes$	$\langle H \rangle$:-1	$\rtimes \text{VVV} \underline{\text{V}} \text{VV} \ltimes$	V:-1	V́	V́V
5.	q_3	$\underline{\rtimes} \langle \text{H} \rangle \ltimes$	$\rtimes$:0	$\rtimes \text{VV} \underline{\text{V}} \text{VVV} \ltimes$	V:-1	V	VV́V
6.	q_3	$\underline{\rtimes} \langle \text{H} \rangle \ltimes$	$\rtimes$:0	$\rtimes \text{V} \underline{\text{V}} \text{VVVV} \ltimes$	V:-1	V	VVV́V
7.	q_3	$\underline{\rtimes} \langle \text{H} \rangle \ltimes$	$\rtimes$:0	$\rtimes \underline{\text{V}} \text{VVVVV} \ltimes$	V:-1	V	VVVV́V
8.	q_3	$\underline{\rtimes} \langle \text{H} \rangle \ltimes$	$\rtimes$:0	$\underline{\rtimes} \text{VVVVVV} \ltimes$	V:-1	V	VVVVV́V
9.	q_f	$\rtimes \langle \text{H} \rangle \ltimes$	$\rtimes$:-1	$\rtimes \text{VVVVVV} \ltimes$	$\rtimes$:-1	λ	VVVVV́V

Table 7: Derivation of $f([\langle \underline{\text{H}} \rangle, \text{VV}\langle \underline{\text{V}} \rangle \text{VVV}]$=*VVVVV́V* in Zigulu with the MT-FST in Figure 6

	Current state	Tone tape		Vowel tape		Output symbol	Output string
1.	q_0	$\underline{\rtimes}\langle \underline{\text{H}} \rangle \ltimes$		$\underline{\rtimes}\text{V}\langle \underline{\text{V}} \rangle \text{VV} \ltimes$			
2.	q_1	$\rtimes \langle \underline{\text{H}} \rangle \ltimes$	$\rtimes$:+1	$\rtimes \underline{\text{V}} \langle \underline{\text{V}} \rangle \text{VV} \ltimes$	$\rtimes$:+1	λ	
3.	q_2	$\rtimes \langle \underline{\text{H}} \rangle \ltimes$	$\langle H \rangle$:0	$\rtimes \text{V} \langle \underline{\text{V}} \rangle \text{VV} \ltimes$	V:+1	V	V
4.	q_3	$\rtimes \langle \underline{\text{H}} \rangle \ltimes$	$\langle H \rangle$:0	$\rtimes \text{V} \langle \text{V} \rangle \underline{\text{V}} \text{V} \ltimes$	$\langle V \rangle$:+1	V	VV́
5.	q_2	$\rtimes \langle \text{H} \rangle \underline{\ltimes}$	$\langle H \rangle$:+1	$\rtimes \text{V} \langle \text{V} \rangle \text{V} \underline{\text{V}} \ltimes$	V:+1	V́	VV́V́
6.	q_2	$\rtimes \langle \text{H} \rangle \underline{\ltimes}$	$\ltimes$:0	$\rtimes \text{V} \langle \text{V} \rangle \text{VV} \underline{\ltimes}$	V:+1	V	VV́V́V
7.	q_f	$\rtimes \langle \text{H} \rangle \ltimes$	$\ltimes$:+1	$\rtimes \text{V} \langle \text{V} \rangle \text{VV} \ltimes$	$\ltimes$:+1	λ	VV́V́V

Table 8: Derivation of $f([\langle \underline{\text{H}} \rangle, \text{V}\langle \underline{\text{V}} \rangle \text{VV}]$=*VV́V́V* in Bemba with the MT-FST in Figure 7

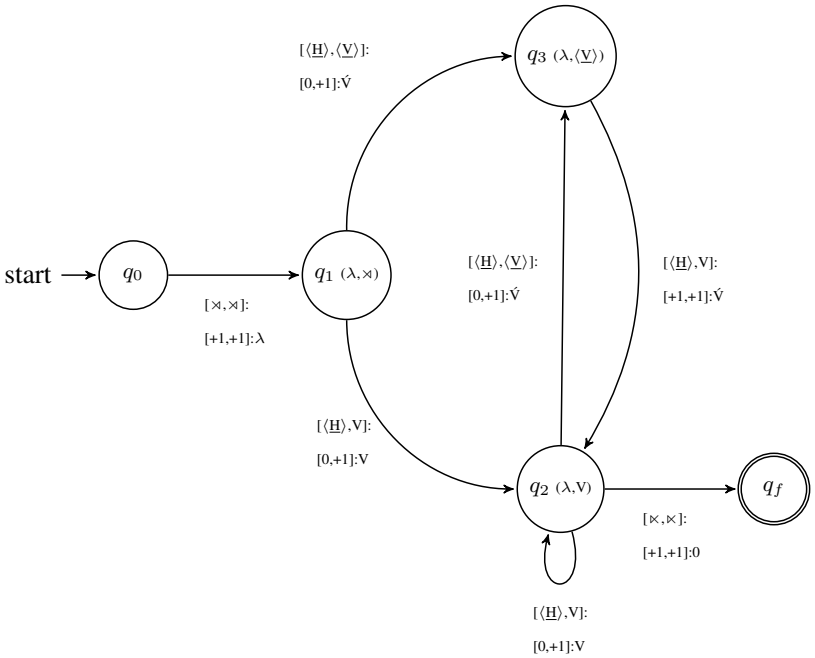


Figure 7: MT-FST for Bemba

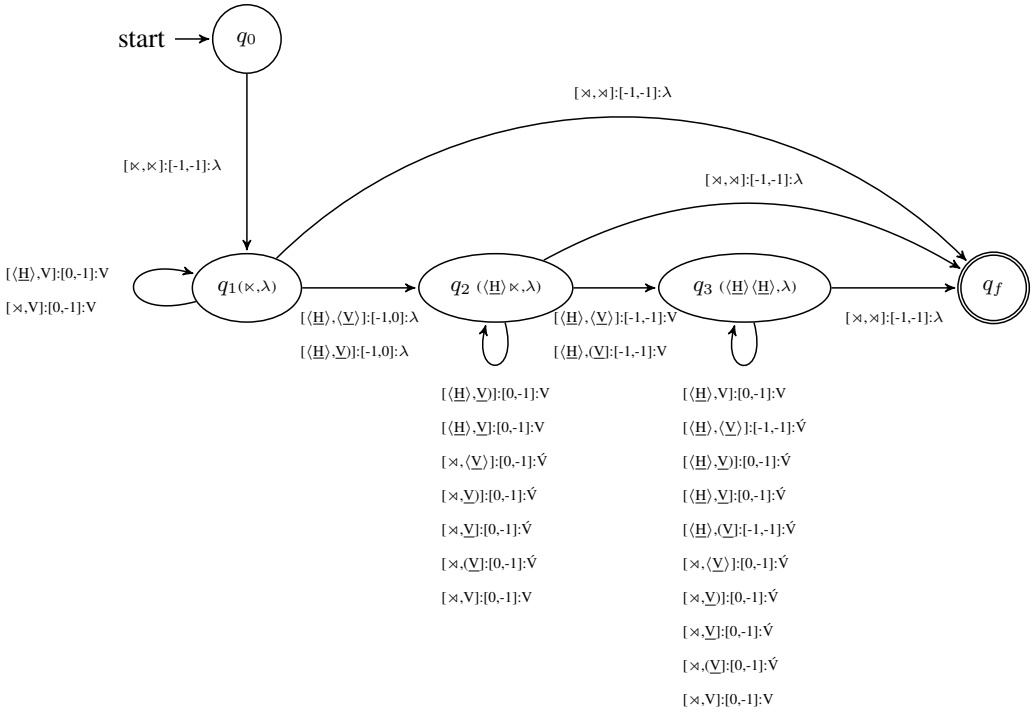


Figure 8: MT-FST for Arusa

	Current state	Tone tape		Vowel tape		Output symbol	Output string
1.	q_0	$\rtimes\langle \mathrm{H}\rangle\langle \mathrm{H}\rangle\underline{\ltimes}$		$\rtimes\langle \mathrm{V}_1\rangle \mathrm{V}(\mathrm{V}_1\mathrm{V}_1)\mathrm{V}\underline{\ltimes}$			
2.	q_1	$\rtimes\langle \mathrm{H}\rangle\underline{\langle \mathrm{H}\rangle}\ltimes$	$\ltimes$:-1	$\rtimes\langle \mathrm{V}_1\rangle \mathrm{V}(\mathrm{V}_1\mathrm{V}_1)\underline{\mathrm{V}}\ltimes$	$\ltimes$:-1	λ	
3.	q_1	$\rtimes\langle \mathrm{H}\rangle\underline{\langle \mathrm{H}\rangle}\ltimes$	$\langle H\rangle$:0	$\rtimes\langle \mathrm{V}_1\rangle \mathrm{V}(\mathrm{V}_1\underline{\mathrm{V}_1})\mathrm{V}\ltimes$	V:-1	V	V
4.	q_2	$\rtimes\underline{\langle \mathrm{H}\rangle}\langle \mathrm{H}\rangle\ltimes$	$\langle H\rangle$:-1	$\rtimes\langle \mathrm{V}_1\rangle \mathrm{V}(\mathrm{V}_1\underline{\mathrm{V}_1)}\mathrm{V}\ltimes$	$V_1)$:-1	λ	V
5.	q_2	$\rtimes\underline{\langle \mathrm{H}\rangle}\langle \mathrm{H}\rangle\ltimes$	$\langle H\rangle$:0	$\rtimes\langle \mathrm{V}_1\rangle \mathrm{V}\underline{(\mathrm{V}_1}\mathrm{V}_1)\mathrm{V}\ltimes$	$V_1)$:-1	V	VV
6.	q_3	$\underline{\rtimes}\langle \mathrm{H}\rangle\langle \mathrm{H}\rangle\ltimes$	$\langle H\rangle$:-1	$\rtimes\langle \mathrm{V}_1\rangle\underline{\mathrm{V}}(\mathrm{V}_1\mathrm{V}_1)\mathrm{V}\ltimes$	$(V_1$:-1	V	VVV
7.	q_3	$\underline{\rtimes}\langle \mathrm{H}\rangle\langle \mathrm{H}\rangle\ltimes$	$\rtimes$:0	$\rtimes\underline{\langle \mathrm{V}_1\rangle}\mathrm{V}(\mathrm{V}_1\mathrm{V}_1)\mathrm{V}\ltimes$	$(V$:-1	V	VVVV
8.	q_3	$\underline{\rtimes}\langle \mathrm{H}\rangle\langle \mathrm{H}\rangle\ltimes$	$\rtimes$:0	$\underline{\rtimes}\langle \mathrm{V}_1\rangle \mathrm{V}(\mathrm{V}_1\mathrm{V}_1)\mathrm{V}\ltimes$	$\langle V_1\rangle$:-1	Ṿ	ṾVVVV
9.	q_f	$\rtimes\langle \mathrm{H}\rangle\langle \mathrm{H}\rangle\ltimes$	$\rtimes$:-1	$\rtimes\langle \mathrm{V}_1\rangle \mathrm{V}(\mathrm{V}_1\mathrm{V}_1)\mathrm{V}\ltimes$	$\rtimes$:-1	λ	ṾVVVV

Table 9: Derivation of f([$\underline{\langle \mathrm{H}\rangle}\langle \mathrm{H}\rangle$,$\underline{\langle \mathrm{V}}$ V($\underline{\mathrm{VV}}$V]=*V́ VVVV* in Arusa with the MT-FST in Figure 8