Computational locality in nonlinear morphophonology

Hossep Dolatian and Jonathan Rawski
Dept. of Linguistics & Institute for Advanced Computational Science
Stony Brook University

November 29, 2020

Abstract

This paper presents a computational analysis of locality in nonlinear phonology and morphology. Most segmental phonology and concatenative morphology are computationally local and characterized by the class of Input Strictly Local functions. We generalize these functions to consider multiple inputs, i.e. Multi-Input Strictly Local. We implement this approach using multi-tape automata, and find that they elegantly characterize the bulk of nonlinear suprasegmental phonology and morphology. We demonstrate these results for both tonal phonology and template-filling patterns in root-and-pattern morphology. We show that locality is affected by some theoretical choices (directionality, tier-conflation), but not others (phonological content of templates). For prosodic morphology, nuanced representational choices can affect locality. The flexibility and precision given by the mathematical theory of formal languages provides precise discussion of the nuances of phonological and morphological complexity and substance in a theory-independent but rigorous way.

Keywords: tone, prosodic morphology, Semitic templates, prosodic template, root-and-pattern morphology, nonlinear representations, locality, computational locality, multi-tape finite state transducers, computational phonology, finite-state phonology

1 Introduction

The mathematical theory of formal language theory (FLT) has enjoyed a recent resurgence in phonology. The reason is clear: formal languages provide necessary and sufficient conditions on the classes of functions that define the structural well-formedness of phonological phenomena.

The phenomena may concern well-formed structures (phonotactics) or phonological transformations (processes). For a given phenomenon, such conditions are invariant across all possible implementations. These are properties which any computational or cognitive mechanism needs to be sensitive to in order to correctly classify and process the phenomenon. The transparent guarantees given by formal languages define explicit classes of phonological grammars, whether for getting the right generalizations or for learning (Heinz 2018).

However, there has been methodological angst against using FLT. Segmental phonology can be adequately modeled with string-based representations, and it has a very limited generative capacity (Chandlee 2014). Most segmental phonology is computationally local and definable with subregular **Input Strictly Local** (ISL) functions. In contrast, suprasegmental processes like tone are computationally more expressive (Jardine 2016a), requiring the **Regular** functions. This dichotomy has provoked many responses. On the one end, some segmental processes like vowel harmony can display high level of expressivity (McCollum et al. 2020). On the other end, some argue against the use of FLT analyses in favor of constraint-interaction grammars like Optimality Theory (Pater 2018, 2019; Jardine 2019a).[1]

A representational assumption underlying much of this computational work treats the input and output of these functions as a single string of segments. Strings are a simple data structure that accentuate the sequential aspects of phonology. However, a crucial insight of generative phonology was that certain suprasegmental phenomena satisfy conditions of temporal independence, intuitively captured via autosegmental tiers (Leben 1973; Williams 1976; Goldsmith 1976). This representation was first proposed for tone but then extended to non-concatenative morphology (McCarthy 1979, 1981).

For tone, recent work in mathematical phonology has incorporated this autosegmental insight by changing the data structure or representation of the input from strings to graphs. Over these *autosegmental graphs*, tonal phonology has weaker expressivity than over string-based representations (Jardine 2017a; Chandlee and Jardine 2019a). String-based representations required the full power of the non-local *Regular* functions to characterize tone, since many tonal alternations behave like long-distant processes across the segmental tier. Over autosegmental graphs, tonal computation is *local*, because any apparent non-locality over the segmental tier is often local over the tonal tier. This highlights a trade-off between representation and computation: more elaborate representations yield a (formally) simpler computation.[2]

The present paper provides a different perspective on nonlinear computation (Dolatian and Rawski 2020a,b; Rawski and Dolatian 2020).Instead of using autosegmental graphs, we combine string-

[1] Constraint-interaction grammars are computationally nebulous. They are Turing-complete in general (Smolensky 1993; Hale and Smolensky 2006), can both over- and under-generate by creating bizarre processes (Hao 2019; Lamont prep), are prone to hard-to-discover errors (Karttunen 2006a,b), have issues with tractability (Idsardi 2006; Heinz et al. 2009), and have limited success at being implementable with finite-state restrictions (Eisner 1997; Karttunen 1998; Frank and Satta 1998; Riggle 2004; Gerdemann and Hulden 2012).

[2] It is also possible that a change in representation doesn't affect the computation, such as for the use of Q-theory (Shih and Inkelas 2018; Jardine et al. 2020), tonal feature geometry (Oakden 2020), and syllable representations (Strother-Garcia 2019).

based and tier-based representations to exploit the simple computation of strings and the relativized locality of tiers. We treat nonlinear processes as n-ary functions, or functions which can take multiple strings as input or output. In contrast, string-to-string and graph-to-graph transductions are unary functions that take only one input representation and produce one output. Whereas string-to-string functions are implemented with single-tape finite-state transducers (FSTs), n-ary functions are implemented with multi-tape FSTs.

Our main result is to extend the formal notion of locality to consider multiple inputs, which we implement as restrictions on multi-tape automata. We call this class of automata Multi-Input Strictly Local (MISL). We show that MISL automata elegantly characterize a majority of nonlinear phonology. Additionally, local segmental phonology is ISL, and ISL is just a special case of MISL functions. We demonstrate our results for tonal phonology and for template-filling processes in root-and-pattern morphology. This characterization shows the salience of locality over different representations. The computational rigor underlying the automata-theoretic perspective is a distinct advantage over previous approaches. Our approach in factoring representation and computation allows us to discuss the nuances of phonological and morphological substance in a theory-independent but rigorous way.

This paper is organized in the following way. Section 2 discusses the computational locality over phonological representations. Section 3 relativizes locality over multiple inputs using multi-tape automata. Section 4 analyzes the typology of tonal processes, showing that the bulk of them are computationally local. Section 5 discusses formal nuances to directionality and tier conflation. Sections 6 & 7 extend our approach to non-concatenative morphological processes, using Semitic template-filling as a case study, which is also largely local. Section §8 discusses various issues in this treatment of non-concatenativity dealing with infinity, the phonological substance of templates, and phonological emergence. Conclusions are in §9.

2 Locality in phonological representations

A common topic in phonological theory is the role of adjacency (Odden 1994). There is a substantial body of work which shows that segmental processes tend to be local. These intuitive notions of locality are formalizable with subclasses of FSTs and their corresponding functions. There is comparatively little work on analyzing locality in autosegmental processes. In this section, we discuss locality and break it down into a set of formal parameters couched within FLT.

2.1 Locality of segmental phonology over single-strings

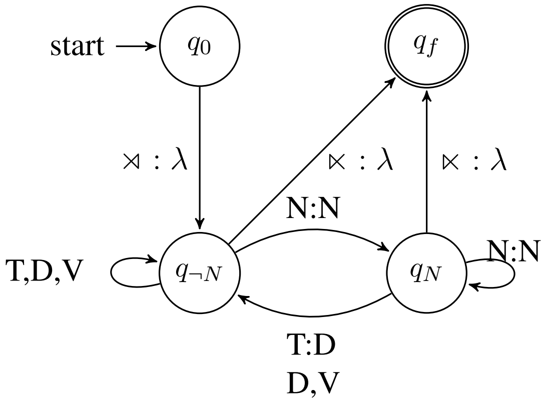
Consider the common segmental process of post-nasal voicing. The target of the rule is an underlying stop, and the trigger is an *immediately* preceding nasal.

(1) a. T → D / N _

b.	/ampa/	amba	/amta/	amda	/amka/	amga
	/anpa/	anba	/anta/	anda	/anka/	anga

Post-nasal voicing is controlled by strict adjacency. Computationally, this process is a Regular function, for which there are many different implementations. The most common implementation is an FST (Kaplan and Kay 1994; Roche and Schabes 1997). Figure 1 depicts an FST for post-nasal voicing. It consists of four states. Two of these states $q_N, q_{\neg N}$ do the brunt of the work. They encode the most-recently seen input symbol: a nasal vs. a non-nasal. The states q_0 and q_f are unique initial and final states. The former reads the start-boundary $\rtimes$, the latter is accessed once we read the end-boundary $\ltimes$.

Figure 1: *Single-tape FST for post-nasal voicing*



A sample derivation for /anta/ is in Figure 2. The input is flanked by the boundary symbols $\rtimes, \ltimes$. Each row keeps track of: i) the current state, ii) what transition arc was taken to reach this state, iii) the next input symbol which will be read, iv) the outputted symbol, and v) the current output string. The current read-head r is situated before the underlined symbol.

Figure 2: *Derivation of simple post-nasal voicing in* /anta/

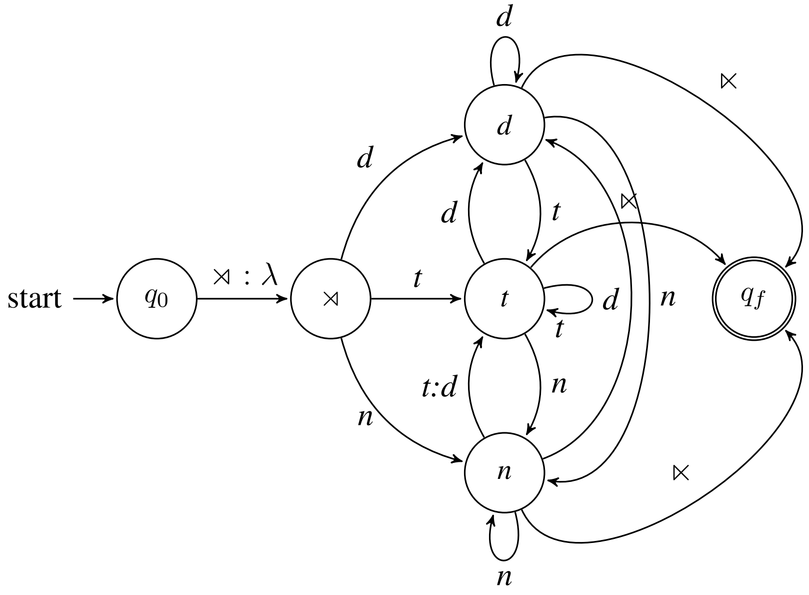
	Current state	Used Arc	Input tape	Output symbol	Output string
1.	q_0		$_r\underline{\rtimes}$anta$\ltimes$		
2.	$q_{\neg N}$	$\rtimes$:λ	$\rtimes_r\underline{\text{a}}nta\ltimes$	λ	
3.	$q_{\neg N}$	a:a	$\rtimes$a$_r\underline{\text{n}}$ta$\ltimes$	a	a
4.	q_N	n:n	$\rtimes$an$_r\underline{\text{t}}$a$\ltimes$	n	an
5.	$q_{\neg N}$	t:t	$\rtimes$ant$_r\underline{\text{a}}\ltimes$	d	and
6.	$q_{\neg N}$	a:a	$\rtimes$anta$_r\underline{\ltimes}$	a	anda
7.	q_f	$\ltimes$:λ	$\rtimes$anta$\ltimes$	λ	anda

FSTs can compute regular relations, but we don't need the full power of FSTs in order to compute post-nasal voicing. Just as adjacency or locality is a restrictive property of the voicing rule (1a), we can restrict the power of FSTs to only compute *local* functions.

Mathematically, the process of post-nasal voicing computes a *input strictly local* function (ISL) (Chandlee 2014; Chandlee and Heinz 2018; Chandlee et al. 2018). Let x be a string made up of m elements: $x_1x_2...x_{j-1}x_jx_{j+1}...x_m$. A process is ISL if the output correspondent of an input symbol x_j depends only on what symbols exist within a finite window *k* from x_j. That is, we only need information on the symbol x_j, the previous symbol x_{j-1}, the subsequent symbol x_{j+1}, and all other symbols within this finite window *k*. For every ISL function, there exists a canonical ISL FST. The FST's states are interpreted as keeping track of the last $k-1$ input symbols $(x_{j-k}...x_{j-1})$.[3] There is a different state for every possible substring of size $k-1$. These FSTs can get larger than the minimal FST.

Tying this formal concept back to post-nasal voicing, post-nasal voicing is a 2-ISL process that uses a window of size $k = 2$ over the input: the current input symbol and the previous input symbol. This process is computable with the canonical 2-ISL FST in Figure 3 with a limited input alphabet of {n,t,d}.

Figure 3: *Canonical single-tape FST for 2-ISL function of post-nasal voicing*



With a small alphabet, the above canonical ISL FST has 6 states: q_0, q_f are the initial and final state, $q_⋊$ reads the start boundary, and states q_t, q_d, q_n do the brunt of the work and read the input segments. Each state encodes the last $k-1$ input symbols $(= 2-1 = 1$ symbols). The states encode whether the previous segment was a nasal, voiceless stop, or voiced stop. In contrast, the minimal FST from Figure 1 has fewer states. It merges states $q_⋊, q_t, q_d$ into a single state $q_{\neg N}$. This state encodes whether the previous input symbol was a non-nasal. However, even though the ISL

[3] The explanation gets complicated when we look at what symbol follow x_j. An FST would need to read the entire substring $x_j...x_{j+k-1}$ before outputting anything (cf. Chandlee 2014:ch:3.1).

FST has more states, it is not more expressive than the minimal FST.

2.2 Non-locality of tone over single strings

The previous section showed how a segmental process can be computationally local. When evaluating the computational properties of a phonological process, it matters how we represent the input and output. In this section, we show that by using single-string inputs, tonal processes are not local in the *same* sense as segmental processes.

Representationally, post-nasal voicing takes as input a *single* string of symbols. Both Regular and ISL FSTs work over this single input string and locally map it to an output string. Formally, the FST is a single-tape FST because the input is represented as a single tape of symbols, and the FST uses a single read-head.[4]

The importance of representation is clearer for tonal process like final-tone spreading. In Mende, words use a restricted set of possible tone sequences (Figure 4). For each word below, we also show a separate string of tones: *kɔ́ H*. We break up contour tones to their component tones in parentheses: *mbû* (HL). The generalization is that tones are associated left-to-right. If there are more tones than vowels, then the final tone forms a contour tone. If there are more vowels than tones, then the final tone is spread across multiple vowels at the right-edge of the string.

Figure 4: *Distribution of tones in Mende* (Jardine 2016b:17)

	1-vowel words			2-vowel words			3-vowel words		
/H/	kɔ́	H	'war'	pɛ́lɛ́	HH	'house'	háwámá	HHH	'waist'
/L/	kpà	L	'debt'	bɛ̀lɛ̀	LL	'pants'	kpàkàlì	LLL	'three-legged chair'
/HL/	mbû	(HL)	'owl'	ngílà	HL	'dog'	félàmà	HLL	'junction'
/LH/	mbǎ	(LH)	'rice'	nìká	LH	'cow'	ndàvúlá	LHH	'sling'
/LHL/	mbã	(LHL)	'companion'	nyàhâ	L(HL)	'woman'	nìkílì	LHL	'groundnut'

Tonal processes like left-to-right spreading are intuitively local, but this intuition depends on our representations. An early trend in generative phonology was to represent tones as part of a vowel, and thus part of a single input string: /háwama/ or /V́VV/. Over this single-string representation, tonal processes can be computed by a single-tape FST (Gibbon 1987, 2001). However, the process is not an ISL function. It is non-local because there can be an unbounded distance between the trigger and target of tone spreading. For example, in *háwám***á** or [V́V́**V́**], the final **vowel** receives its high tone feature the initial vowel.

The Mende example demonstrates a recurrent problem in the computation of tone.[5] Over single-string representations, the computation of certain tonal processes is non-local, even though the

[4] Technically, this FST is called a two-tape FST, where one tape is for the input string and the other tape is for the output string. But for easier explanation, we call this FST a singe-tape FST, meaning it only has one tape for the input.

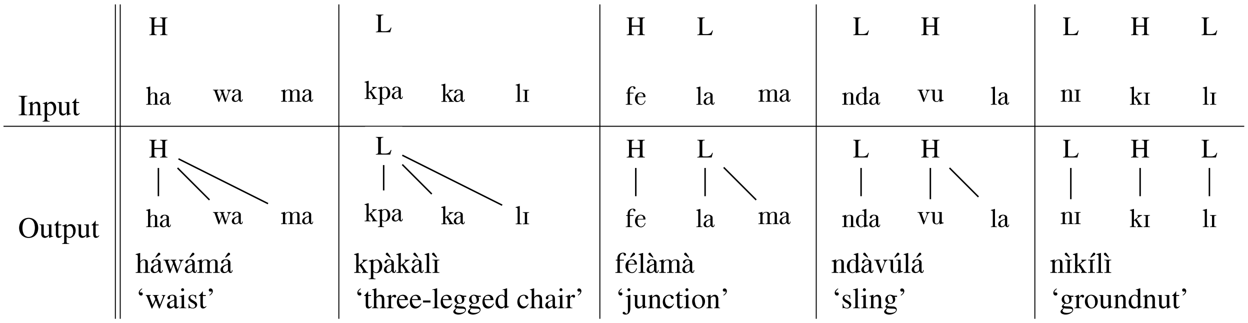
[5] In contrast, a recurrent exception is tone sandhi in East Asian languages, which tends to be local over segmental strings (Chandlee 2019; Oakden 2020).

autosegmental notation implies that the computation *should* be local (Jardine 2016b,a, 2017a; Chandlee and Jardine 2019a). Because of these problems, early work in computational phonology tried to notationally incorporate autosegmental structure into single-string representations, in order to make tonal computation easier to model and design (Bird and Ellison 1994; Kornai 1995; Yli-Jyrä 2013, 2015, 2019). The motivation for this strategy is mostly practical. Single-tape FSTs are efficient and useful tools in NLP. However, these notational strategies don't affect the generative capacity of tone. Over a single-string representation, Mende left-to-right spread is still non-local.

2.3 Multi-input representations for tone

Since the advent of autosegmental phonology (Goldsmith 1976), the representation of tone has broken away from single-string representations. Instead, tones are represented on a separate tier or string of elements. For Mende, the representation of words is now a set of multiple strings which are associated together.

Figure 5: *Tone-spreading over single-string inputs*



By breaking apart tone from vowels, the process of left-to-right spreading becomes a multi-input function. This process is local over these multiple inputs. The target is an unassociated vowel on the segment string, and the trigger is the rightmost tone on the tone string. Because the two strings are disconnected, the locality of the process depends on how we *time* the reading of both input strings, i.e., by iteratively applying left-to-right association.

The most faithful interpretation of the input-output transformation in Figure 5 would involve the use of graphs. Most existing mathematical results on tonal phonology are defined over graphs or graph-like structures (Bird and Klein 1990; Bird 1995; Coleman and Local 1991; Coleman 1998). Within mathematical phonology, the specific types of graphs used for tone have been formalized into autosegmental grammars (Jardine 2016a,b, 2017a, 2019b) and more recently melody-local grammars (Jardine 2020). The benefit of using these autosegmental grammars is their transparent interpretation, which makes it possible to analyze their expressivity in terms of formal logic (Jardine 2017b; Chandlee and Jardine 2019a; Zhu 2020; Mamadou and Jardine 2020). However, graphical structures are generally more difficult to process than string-based representations. It is difficult to develop clear implementations for these grammars, whether in terms algebraic or automata-theoretic tools.

In the next section, we introduce an alternative formalism for computing autosegmental structures. These are multi-tape FSTs which simply take as input two independent strings: a tone string and vowel string.

3 Multi-input functions

In this paper, we exploit the fact that tonal phonology operates *as if* it takes two separate input strings. Such **multi-input functions** can be computed by **multi-tape FSTs** (MT-FST). We develop a subclass of these grammars based on locality. We call them **multi-input strictly local** (MISL) functions which are computed by MISL MT-FSTs. We explain how MT-FSTs operate (§3.1), illustrate them for tone (§3.2), and then introduce locality (§3.3).

3.1 Multi-tape finite-state transducers

Multi-tape FSTs are FSTs where the input is a tuple of strings, and the computation occurs on multiple separate tapes (Rabin and Scott 1959; Elgot and Mezei 1965; Fischer 1965; Fischer and Rosenberg 1968; Furia 2012). The output can be a one or more tapes. We focus on MT-FSTs that have multiple input tapes, but a single output tape. Every input tape has its own unique read-head.

We are not the first to use MT-FSTs for tone or computational phonology. In computational linguistics, Kay (1987) developed a sketch of an MT-FST for Semitic root-and-pattern morphology. Larger-scale formalizations were later developed for tone (Wiebe 1992) and Arabic (Kiraz 2000, 2001; Habash and Rambow 2006). The benefit of our MT-FST formalization is that it connects the computation of tone with pre-existing results on the use of MT-FSTs, whether for computation or implementation. Autosegmental grammars are conceptually attractive but novel. They don't have many pre-existing computational results.[6]

In this paper, we use the informal definition of an MT-FST as essentially just an FST which processes multiple input tapes. For a formal definition of MT-FSTs, see Dolatian and Rawski (2020b). When using MT-FSTs, one dichotomy is whether the machine's multiple read-heads are moving simultaneously for every transition. If the read-heads are *always* moving simultaneously, then the MT-FST is a synchronous MT-FST.; otherwise the machine is an asynchronous MT-FST.

In terms of generative capacity, synchronous MT-FSTs are equivalent to single-tape FSTs. The multiple tapes can be compressed into a single tape. Because of this equivalence, synchronous

[6] An open question is to use MT-FSTs as an implementation for autosegmental melody-local grammars (Mamadou and Jardine 2020). We conjecture that melody-local grammars are notationally equivalent to MISL functions. To show equivalence, we need to enrich the alphabet to include both preassociated and non-associated symbols, as endorelations are not restricted by set union.

MT-FSTs have been used to model intermediate representations in computational morphology and phonology (Hulden 2017a,b). In contrast, asynchronous MT-FSTs are more expressive. When interpreted as acceptors, they are capable of generating all regular languages and even some context-free or context-sensitive languages (Wiebe 1992:ch7).

Within computational phonology, some have used synchronous MT-FST (Kiraz 2001), while others have used asynchronous MT-FSTs (Wiebe 1992). Both have likewise been used for the computation of feature tiers in speech processing (Carson-Berndsen 1998; Shu 2006). We discuss synchronous MT-FSTs more in §7.1 for Semitic. In this paper, we use the more expressive class of asynchronous MT-FSTs. We focus on what sufficient information is needed to compute autosegmental processes. We define a locally-based subclass for MT-FSTs, and show that autosegmental phonology is computationally local.

3.2 Tone as a multi-input function

Tonal phonology can easily be represented as a multi-input function. Importantly, multi-input functions are defined over sets of *strings*, not graphs. In Figure 6, we show the input and output structures of a multi-input interpretation of Mende left-to-right spreading. The input is a pair of strings: a string **T** of unassociated tones, and a string **V** of unassociated vowels. They are called the **T**-string and **V**-string respectively. For illustration, we omit consonants from the **V**-string and we don't specify vowel quality. The output is a single string over a new alphabet of toned-vowels. The strings are flanked by the boundaries ⋊,⋉.

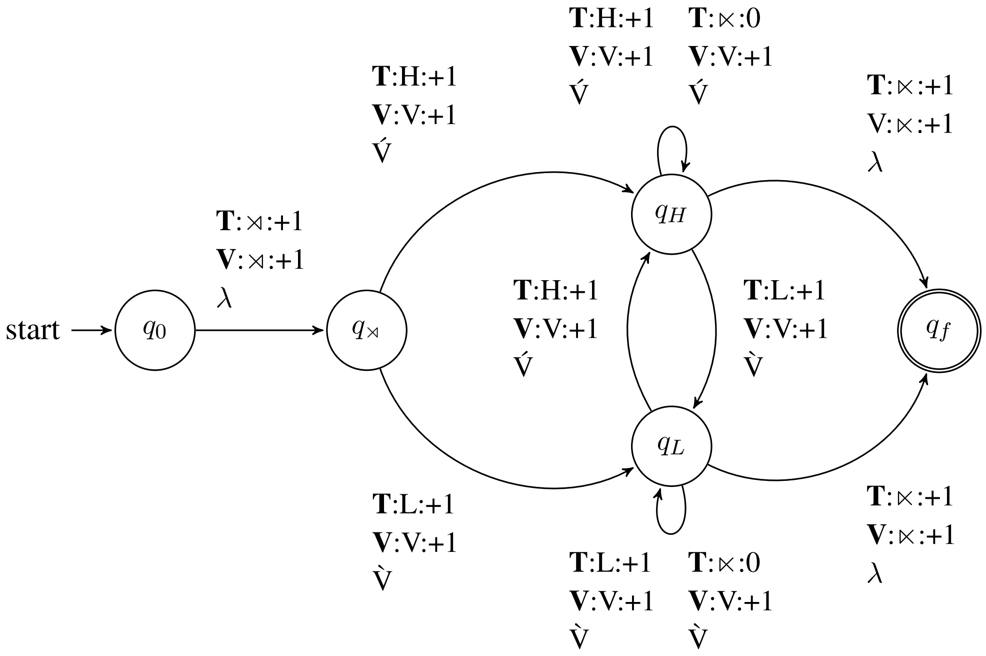
Figure 6: *Processing left-to-right spreading as a multi-input function*

Input:	**T**-string:	⋊H⋉	⋊L⋉	⋊H L ⋉	⋊L H ⋉	⋊L H L ⋉
	V-string:	⋊V V V ⋉	⋊V V V ⋉	⋊V V V ⋉	⋊V V V ⋉	⋊V V V ⋉
Output:	Single string	⋊V́ V́ V́ ⋉	⋊V̀ V̀ V̀ ⋉	⋊V́ V̀ V̀ ⋉	⋊V̀ V́ V́ ⋉	⋊V̀ V́ V̀ ⋉
		háwámá	kpàkàlì	félàmà	ndàvúlá	nìkílì

The above multi-input function can be implemented with the MT-FST in Figure 7. The two input strings form two tapes: **T**-tape and **V**-tape. The machine has two read-heads r_T, r_V for the two tapes. In each transition arc, either one or both of the read-heads advance on the two tapes. The MT-FST computes left-to-right spreading. The MT-FST will associate every pair of tones and vowels until it has run out of tones. In this case, the final tone is associated with every remaining syllable.[7]

[7] This particular machine does not handle cases where there are more tones than vowels, i.e., contour tones. We show how contour tones are formalized in Rawski and Dolatian (2020).

Figure 7: *MT-FST for Mende left-to-right spreading*



The MT-FST has 5 states. The states q_0 and q_f are unique initial and final states. The state $q_{\rtimes}$ reads the start-boundary $\rtimes$. We enter state q_L if the next tone on the **T**-string is a low tone; we enter the state q_H if the next tone is a high tone. These two states will go through the **T**-string and **V**-string and will generate a toned-vowel. The transition arcs are interpreted as follows.[8] Each arc has three lines. For the arc from $q_{\rtimes}$ to q_H, the arc is:

T:H:+1
V:V:+1
V́

Given a H tone on the **T**-tape and vowel V on the **V**-tape, this transition arc will output a high-toned vowel V́. The first two lines show how the MT-FST moves across the two input tapes; they consist of the tape's name, what input symbol to read, and the direction to take upon reading. The line '**T**:H:+1' means that upon reading a H tone on the **T**-tape, the FST progresses or advances (+1) on the **T**-tape. Similarly for '**V**:V:+1', upon reading any vowel V on the **V**-tape, the machine advances. The third line shows the output string for this transition arc as 'V́'. The numbers 0 and +1 are direction instructions which tell the machine to either stay put (0) or advance (+1) on some tape. Allowing one tape to halt while another tape advances allows the machine to move asynchronously through the input strings.

[8] For illustration, we use a simpler notation in this paper than in Dolatian and Rawski (2020b). There, the transition arcs are represented in terms of vectors and tuples.

Figure 8 shows a derivation for *félàmà* using just the vowels. The columns keep track of i) the current state, ii) what transition arc we took on the **T**-tape, iii) the next input symbol to read on the **T**-tape, iv-v) the used arc and next input symbol for the **V**-tape, vi) the outputted symbol, vii) and the current output string. For readability, we don't show the two read-heads r_T and r_V, but they are situated before the underlined symbol, e.g., $\rtimes_{r_T}\underline{\mathrm{H}}\mathrm{L}\ltimes$and $\rtimes_{r_V}\underline{\mathrm{e}}\mathrm{aa}\ltimes$.

Figure 8: *Derivation of left-to-right spreading for* félàmà *with the MT-FST from Figure 7*

	Current	**T**-tape	**T**-tape	**V**-tape	**V**-tape	Output	Output
	State	Arc	Next symbol	Arc	Next symbol	symbol	String
1.	q_0		$\underline{\rtimes}\mathrm{HL}\ltimes$		$\underline{\rtimes}\mathrm{eaa}\ltimes$		
2.	$q_{\rtimes}$	$\rtimes$:+1	$\rtimes\underline{\mathrm{H}}\mathrm{L}\ltimes$	$\rtimes$:+1	$\rtimes\underline{\mathrm{e}}\mathrm{aa}\ltimes$	λ	
3.	q_H	H:+1	$\rtimes\mathrm{H}\underline{\mathrm{L}}\ltimes$	e:+1	$\rtimes\mathrm{e}\underline{\mathrm{a}}\mathrm{a}\ltimes$	é	é
4.	q_L	L:+1	$\rtimes\mathrm{HL}\underline{\ltimes}$	a:+1	$\rtimes\mathrm{ea}\underline{\mathrm{a}}\ltimes$	à	éà
5.	q_L	$\ltimes$:0	$\rtimes\mathrm{HL}\underline{\ltimes}$	a:+1	$\rtimes\mathrm{eaa}\underline{\ltimes}$	à	éàà
6.	q_f	$\ltimes$:+1	$\rtimes\mathrm{HL}\ltimes$	$\ltimes$:+1	$\rtimes\mathrm{eaa}\ltimes$	λ	*éàà*

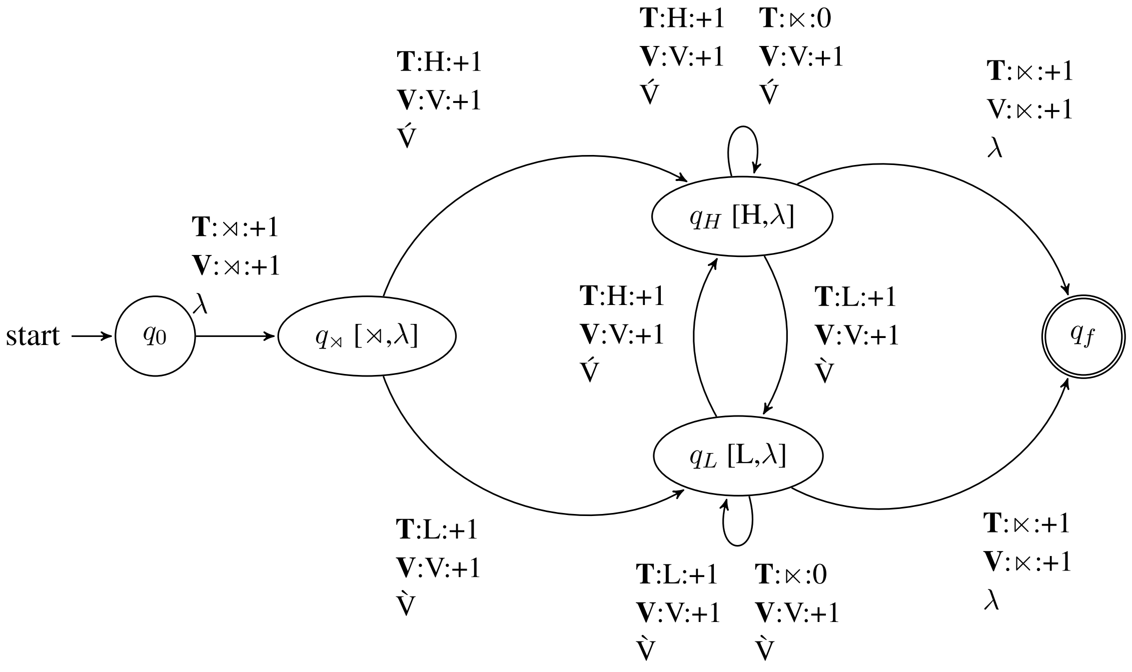
The MT-FST above is *asynchronous* because the two read-heads don't simultaneously advance at every step or transition. At point 5 for *felama*, the **T**-tape and **V**-tape have the configurations $\rtimes\mathrm{HL}\underline{\ltimes}$ and $\rtimes\mathrm{ea}\underline{\mathrm{a}}\ltimes$. The MT-FST has already read the final tone but not the final vowel. The read-head r_T on the **T**-tape has recently read an L tone, and the next symbol on the **T**-tape is the end-boundary $\ltimes$. In contrast on the **V**-tape, the read-tape r_V still has to read a new vowel V (=*a*). With this configuration, the MT-FST will not advance on the **T**-tape (= **T**:$\ltimes$:0), but advance on the **V**-tape (= **V**:*a*:+1) and produce a low-toned vowel *à*. This is how the machine generates final spreading.

3.3 Locality over multi-input representations

The previous section showed that MT-FSTs capture the fact that tonal phonology operates over two separate tiers, formalized as separate strings. The finite-state nature of MT-FSTs restricts the space of possible tonal processes. We now show that as a multi-input function, Mende left-to-right spreading is computationally local. We do so by defining a subclass of MT-FSTs. We then project a class of functions from these machines.

In the case of ISL functions over single-string inputs, each state encodes the previously seen $k-1$ input symbols. For post-nasal voicing, *k* is of size 2; thus each state kept track of the last *single* input symbol. For the MT-FST for Mende, we have two input tapes: the **T**-tape and **V**-tape. We derive the locality of Mende by restricting the possible states of the MT-FST. We repeat in Figure 9 the MT-FST for Mende. But now, the states are notationally augmented with the last seen $k-1$ input symbols on the two tapes.

Figure 9: *MISL MT-FST for Mende left-to-right spreading*



Intuitively, left-to-right spreading looks at the current vowel, the current tone, and checks if the current tone is final or not. It uses a window k_V of size 1 over the **V**-tape, and a window k_T of size 2 over the **T**-tape. The MT-FST is the canonical MT-FST for a multi-input strictly local (MISL) function where the locality windows are $\vec{k} = [k_T, k_V] = [2, 1]$. The state q_L has the local memory $[L, \lambda]$. It memorizes that the fact it saw a L tone on the **T**-tape; it didn't memorize the previously seen vowel. The machine captures the fact that left-to-right spreading is a computationally local process over multiple inputs.

4 Multi-input strict locality across tone

The previous section established two goals. First, it provided a finite-state characterization for tonal processes. It did so by encoding the multi-input nature of tonal associations into MT-FSTs. Second, we established a restrictive subclass of MT-FSTs that are computationally local. Their states are defined in terms of windows of the last *k* seen input symbols on the different tapes. These restricted MT-FSTs are the canonical transducers for MISL functions.

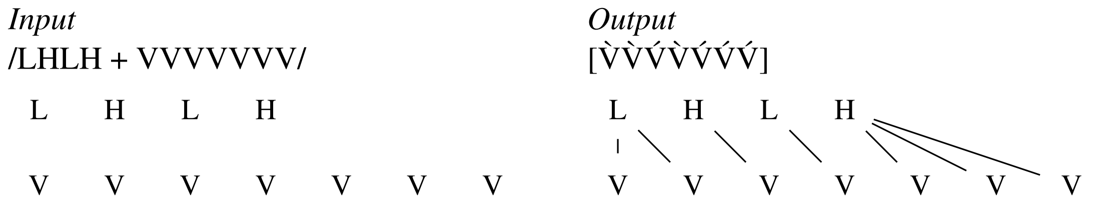
In this section, we go through a typology of tonal phenomena. They are all computable by MT-FSTs, and most are local. There are many typological or theoretical classifications of tonal phonology (Yip 2002; Hyman 2011). But, there are few computational ones. We evaluate MT-FSTs against Koser et al. (2019) and Chandlee and Jardine (2019a); The former is concerned with tonal processes where the input tones are all unassociated (§4.1, while the latter includes processes where tones are underlyingly preassociated (§4.3). For information on the relevant langauges, see the original sources in the two surveys.

4.1 Locality of tone without preassociation

Preassociation is when the input tones are underlying associated with some vowel or tone-bearing unit. For Mende left-to-right spreading, the tones were underlyingly unassociated. In this section, we go over other examples of tonal processes that don't involve preassociation. All of them are definable with MT-FSTs and are local over multiple inputs.

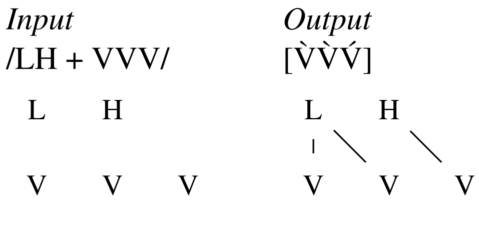
Kikuyu has a process of limited spreading. In Figure 10, we show an example input and output, both in terms of multiple strings and in terms of autosegmental graphs. For the input /LHLH + VVVVVVV/, the first tone associates with the first two vowels. The remaining tones and vowels are associated 1-to-1. If there are more vowels than tones, the final tone is spread: [V̀V̀V́V̀V́V́V́]. Initial spreading requires the context [⋊L, ⋊VV], while final spread is [2,1]-MISL Together, Kikuya is [2,3]-MISL. A sample MT-FST and derivation are in the appendix (Figure 36).

Figure 10: *Limited spreading in Kikuyu*



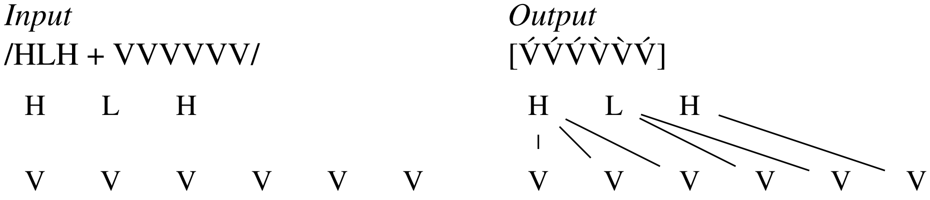
Hausa behaves analogously to Mende but tones are associated *right-to-left* with *initial*-spreading: /LH + VVV/ → [V̀V̀V́]. This is [2,1]-MISL *when* the input string is read right-to-left. A sample MT-FST and derivation are in the appendix (Figure 37). We discuss the role of directionality in §5.1.

Figure 11: *Right-to-left spreading in Hausa*



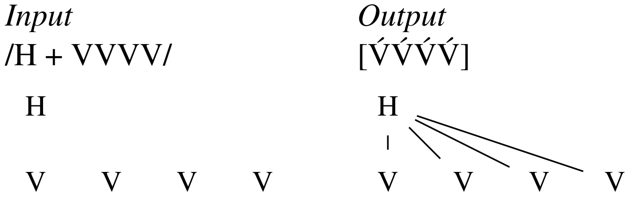
North Karanga Shona is more complex. The initial and final tones are associated to the first and last vowels respectively. The first tone can spread up until the first 3 vowels *but not* to the penultimate vowel. The medial tone can spread up until the penultimate vowel: /HLH + VVVVVV/ → [V́V́V́V̀V̀V́]. The process is MISL but for a very large locality window of [4,6]. We essentially keep track of the substrings ⋊T, T⋉, ⋊VVV, and VV⋉. The locality window may be larger depending on complications discussed in Koser et al. (2019). We don't draw an MISL MT-FST because of its size.

Figure 12: *Edge-in spread in Northern Karanga Shona*



Lastly, Kukuya allows a H tone to spread if it is the only tone: /H + VVV/ → [V́V́V́]. Otherwise, only L tone spreads: /HL + VVV/ → [V́V́V̀], /LH + VVV/ → [V̀V̀V́]. 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 V⋉ if the input tone is LH. We don't draw an MISL MT-FST because of its size.

Figure 13: *Quality-sensitive spreading in Kukuya*



To summarize, just like Mende, the above tonal processes are MISL.

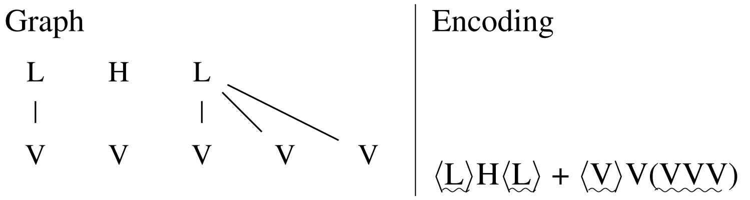
4.2 Linear encoding of tonal preassociation

In contrast to the previous section, tonal processes may involve tone that are underlyingly *preassociated* to vowels. This dependency between the two strings is a reason why graphical structures are useful representations for tone. This is also the reason why linear encodings of tone require special markup systems (Kornai 1995). Before we formalize such tonal processes, we first develop a linear encoding for preassocation.

For our purposes, we use the following encoding in Figure 14, inspired from an encoding system used by Yli-Jyrä (2013). We don't use other proposed encoding systems (Wiebe 1992; Kornai

1995; Yli-Jyrä 2013, 2015, 2019) because they either are designed for single-tape FSTs or don't maintain strict locality.

Figure 14: Encoding preassociation



$\langle\utilde{L}\rangle H\langle\utilde{L}\rangle + \langle\utilde{V}\rangle V(\utilde{VVV})$

If a tone T or single vowel V is preassociated, it is has wavy underlining and has angle brackets: $\langle\utilde{T}\rangle$, $\langle\utilde{V}\rangle$. If a span of multiple vowels are associated to the same tone, they are marked with parentheses instead of angle brackets: $(\utilde{V}\ \utilde{V}\ \utilde{V} \ldots \utilde{V}\ \utilde{V})$. Note that this sequence of preassociated vowels uses only three unique multi-character symbols: '$(\utilde{V}$' and '$\utilde{V}$' and '$\utilde{V})$'. This encoding creates the following enriched input alphabets of multi-character units:

- Σ_T ={ H, L, $\langle\utilde{H}\rangle$, $\langle\utilde{L}\rangle$}
- Σ_V ={ V, $\langle\utilde{V}\rangle$, $(\utilde{V}$, $\utilde{V}$, $\utilde{V})$ }

Other possible configurations, such as word-medial contour tones require a more elaborate encoding which we don't discuss here; see Rawski and Dolatian (2020). We set aside evaluating our encoding mechanism based on Kornai (1995)'s desirada. However, because the alphabet uses multiple preassocation symbols for vowels, the alphabet can create MT-FSTs with a large number of states and arcs.

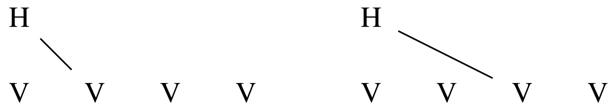
4.3 Locality of preassociated tones

With the above encoding, we show how various preassociated-based tonal processes are computationally local. Examples are taken from Chandlee and Jardine (2019a). Illustrative MT-FSTs and derivations are in the appendix.

Rimi has a process of bounded tone shift. A preassociated tone delinks from its vowel and reassociates with the subsequent vowel: /$V\acute{V}VV$/$\rightarrow$ $VV\acute{V}V$]. In our encoding, the input is /$\langle\utilde{H}\rangle$ + $V\langle\utilde{V}\rangle VV$/. This function is [1,2]-MISL. We need a window of size 1 over the **T**-string because to check if the current tone is a preassociated $\langle\utilde{H}\rangle$. If yes, then we use a window of size 2 over the **V**-string to delink the current preassociated vowel $\langle\utilde{V}\rangle$ and then associate the tone with the next vowel. A sample MT-FST and derivation are in the appendix (Figure 38).

Figure 15: *Bounded tone shift in Rimi* (Chandlee and Jardine 2019a

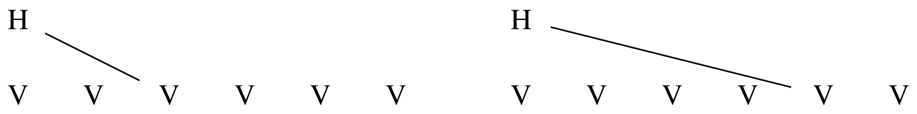
Input *Output*
/VV́VV/ [VVV́V]
/⟨$\underset{\sim}{\text{H}}$⟩ + V⟨$\underset{\sim}{\text{V}}$⟩VV/



Unlike Rimi, Zigula displays unbounded tone shift whereby a preassociated H is delinked from its preassociated vowel. The tone is re-associated with the *penultimate* vowel which can be at any distance away from the underlyingly preassociated vowel: / VVV́VVV/ or /⟨$\underset{\sim}{\text{H}}$⟩ + VV⟨$\underset{\sim}{\text{V}}$⟩VVV/ → [VVVVV́V]. This function is [1,3]-MISL. Given a preassociated ⟨$\underset{\sim}{\text{H}}$⟩ as the current input tone, an underlying preassociated vowel ⟨$\underset{\sim}{\text{V}}$⟩ is delinked regardless of context, and the tone symbol ⟨$\underset{\sim}{\text{H}}$⟩ is re-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 VV⋉. A sample MT-FST and derivation are in the appendix (Figure 39).

Figure 16: *Unbounded tone shift in Zigula*

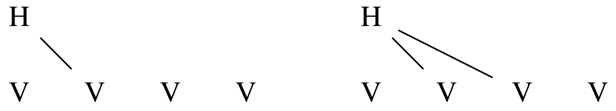
Input Output
/VVV́VVV/ [VVVVV́V]
/⟨$\underset{\sim}{\text{H}}$⟩ + V⟨$\underset{\sim}{\text{V}}$⟩VV/



Similar to Rimi, Copperbelt Bemba shows bounded tone spread. But instead of delinking a preassociated tone-vowel pair, the tone spreads to the subsequent vowel: / VV́VV/ or /⟨$\underset{\sim}{\text{H}}$⟩ + V⟨$\underset{\sim}{\text{V}}$⟩VV/ → [VV́V́V]. This is [1,2]-MISL. The only difference from Rimi is that an input preassociated vowel ⟨$\underset{\sim}{\text{V}}$⟩ keeps its tone in the output. A sample MT-FST and derivation are in the appendix (Figure 40).

Figure 17: *Bounded tone spread in Bemba*

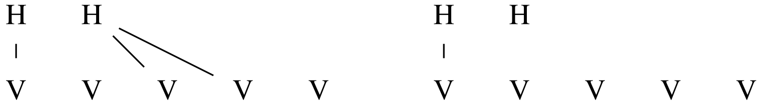
Input *Output*
/VV́VV/ [VV́V́V]
/⟨$\underset{\sim}{\text{H}}$⟩ + V⟨$\underset{\sim}{\text{V}}$⟩VV/



In Arusa, a process of unbounded deletion deletes a phrase-final H tone if it follows another H tone. By deleting the H tone, any preassociated vowels become toneless: /V́ VV́V́V/ or /$\langle\utilde{\text{H}}\rangle$ $\langle\utilde{\text{H}}\rangle$ + $\langle\utilde{\text{V}}\rangle$ V($\utilde{\text{VV}}$)V/ → [V́ VVVV]. When computed over segments, this process is non-local (not ISL) because of the unbounded distance between the two spans of vowels. But given the multi-input representation, this function is [3,1]-MISL. A locality window of size 3 is needed on the **T**-string to check if the input tone symbol is a phrase-final and succeeds another high tone. If yes, then any currently read input vowels are delinked. We don't provide an MT-FST. It is rather large because of our preassociation alphabet, but it is still local.

Figure 18: *Unbounded deletion in Arusa*

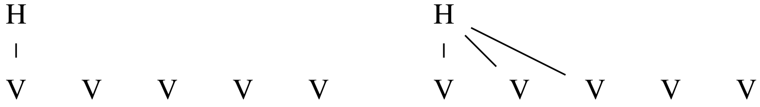
Input	*Output*
/V́ VV́V́V/	[V́VVVV]
/$\langle\utilde{\text{H}}\rangle$ $\langle\utilde{\text{H}}\rangle$ + $\langle\utilde{\text{V}}\rangle$ V($\utilde{\text{VV}}$)V/	



Finally, Ndebele has unbounded spreading of a preassociated H tone up until the ante-penultimate vowel: /V́VVVV/ or /$\langle\utilde{\text{H}}\rangle$ + $\langle\utilde{\text{V}}\rangle$VVVV/ → [V́V́V́VV]. This process is [1,3]-MISL. Reading from right-to-left, the last two vowels surface as toneless. If the current tone symbol is a preassociated $\langle\utilde{\text{H}}\rangle$, then any vowel between $\langle\utilde{\text{V}}\rangle$ and the penult surfaces as V́. This requires a window of size 3 on the **V**-tape, but only 1 on the tone tape. A sample MT-FST and derivation are in the appendix (Figure 41).[9]

Figure 19: *Unbounded spreading in Ndebele*

Input	Output
/V́VVVV/	[V́V́V́VV]
/$\langle\utilde{\text{H}}\rangle$ + $\langle\utilde{\text{V}}\rangle$VVVV/	



5 Nuances in locality and non-locality

Having shown how various tonal processes are computationally local, we now go over some complications which can cause non-locality. We show that directionality of tone affects locality (§5.1),

[9] If we assume that only the final tone can spread; then the locality window is [2,3].

that tier-conflation can create non-locality (§5.2), and that there are tonal patterns that are not local over the input (§5.3).

5.1 Computation of directionality

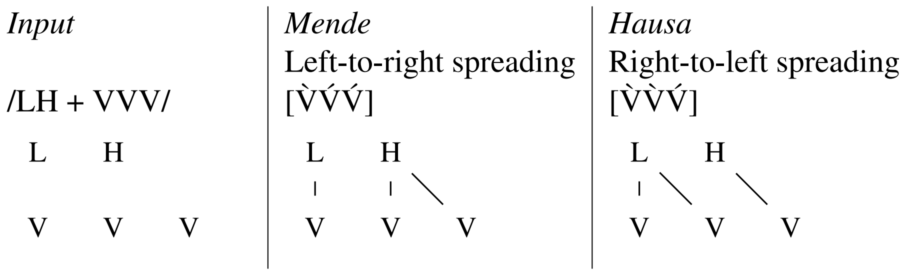
It it is a common fact that phonological processes show directionality. Computationally, directionality matters. For tone, some tonal processes are MISL only under a specific direction, i.e., whether the input is read left-to-right or right-to-left.

For directionality in segmental phonology, left-to-right vs. right-to-left rules require different classes of formal functions. A simple example comes from the computation of vowel harmony (Heinz and Lai 2013). Progressive (left-to-right) harmony requires left-subsequential functions which read the input left-to-right. Analogously, regressive (right-to-left) harmony requires right-subsequential functions. Outside of vowel harmony, directionality is manifested in iterative segmental rules (Howard 1972) and syllabification (Itô 1989). For these processes, the computation requires different directionality parameters (Chandlee et al. 2015; Chandlee and Jardine 2019b; Dolatian et al. prep).

However, for local non-iterative segmental phonology, left-to-right ISL functions are sufficient. Over a single string, there is no difference in expressivity between left-to-right vs.r̃ight-to-left ISL functions (Chandlee 2014). This is because ISL functions act like the application of simultaneous SPE rules.

For tone however, directionality matters. Recall that Mende has left-to-right spreading which is [2,1]-MISL. Similarly, Hausa has the mirror process of right-to-left spreading. Hausa is likewise [2,1]-MISL.

Figure 20: *Left-to-right vs. right-to-left spreading*



Although both left-to-right spreading and right-to-left spreading are MISL, the two functions require different directions. Hausa must be read right-to-left. If the input were read left-to-right, then the MT-FST would not be MISL, and would also require non-determinism. Informally, the machine has to guess if the first tone L has to spread to the first vowel or not. It would spread if the input were /LH + VVV/ or /LHL + VVVV/, i.e., the input had more vowels than tones. But the initial L would not spread if the input were /LH + VV/, i.e., the input had an equal number of tones and vowels.

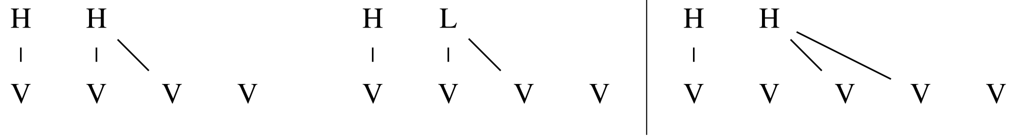
5.2 Tier-conflation and output structure

So far, the MT-FSTs for tone generated a single output string where the tones and vowels are combined. In a sense, our formalization applies tier-conflation (McCarthy 1986). In contrast in an alternative formalization, the MT-FST would generate two output strings: one for tones and one for vowels, with some special markup to show their associations. This alternative doesn't apply tier-conflation.

The mechanism of tier-conflation has computational effects in multiple formalisms (cf. see discussion in Oakden 2020). It likewise matters for multi-input functions. To illustrate, Luganda has a process of bounded Meussen's rule. If a preassociated 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 $/\langle\utilde{H}\rangle\langle\utilde{H}\rangle + \langle\utilde{V}\rangle(\utilde{VV})V/$ maps [V́V̀V̀V]. If the vowel spans aren't contiguous, then the rule does not apply: /V́VV́V́V/ or $/\langle\utilde{H}\rangle\langle\utilde{H}\rangle + \langle\utilde{V}\rangle V(\utilde{VV})V/$ is faithfully mapped to [V́VV́V́V].

Figure 21: *Bounded Meussen's rule in Luganda*

Application		*Non-application*
Input	*Output*	*Input = Output*
/V́V́V́V/	[V́V̀V̀V]	/V́VV́V́V/ → [V́VV́V́V]
$/\langle\utilde{H}\rangle\langle\utilde{H}\rangle + \langle\utilde{V}\rangle(\utilde{VV})V/$		$/\langle\utilde{H}\rangle\langle\utilde{H}\rangle + \langle\utilde{V}\rangle V(\utilde{VV})V/$



Because of the dependence on contiguity over the both the tone-string and vowel-string, this tonal process is MISL only if we don't apply tier-conflation. To illustrate, assume that the function generates only *one* output string. For /V́V́V́V/, the **T**-string is $\langle\utilde{H}\rangle\langle\utilde{H}\rangle$, and the **V**-string contains two vowels preassociated to the two different tones which we *represent* with butting brackets: $/\langle\utilde{H}\rangle\langle\utilde{H}\rangle + \langle\utilde{V}\rangle(\utilde{VV}\ldots\utilde{V})/$. The first vowel ⟨V⟩ maps to a high V́. The second vowel $(\utilde{V}$ will map to a surface low toned vowel V́.

The second vowel $(\utilde{V}$ starts a span of preassociated vowels. But for the other vowels like the span-final $\utilde{V})$, 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 $\utilde{V})$ is preceded by the substring $\langle\utilde{V}\rangle$ $(\utilde{V}$ or not. Over the **V**-string, this type of information is long-distant. In contrast, consider the case of non-application in /V́VV́V́V/ or $/\langle\utilde{H}\rangle\langle\utilde{H}\rangle + \langle\utilde{V}\rangle V(\utilde{VV}\ldots\utilde{V})/$. The span-final vowel $\utilde{V})$ should not undergo the rule because its span $(\utilde{VV}\ldots\utilde{V})$ did not immediately follow the the preassociated vowel $\langle\utilde{H}\rangle$.

Thus, bounded Meussen's rule is not MISL if we apply tier-conflation. But if we don't apply tier-conflation, then the rule is [2,2]-MISL. The input /⟨H̰⟩⟨H̰⟩ + ⟨V̰⟩(V̰ V̰ V̰)/ is mapped to [⟨H̰⟩⟨L̰⟩ + ⟨V̰⟩(V̰ V̰ V̰)] 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 ⟨H̰⟩ and immediately succeeds another tone symbol ⟨H̰⟩ *and* if ii) the current vowel symbol is preassociated ⟨V̰⟩ or starts a span of preassociated vowels (V̰, and follows a span of preassociated vowels ⟨V̰⟩ or V̰). All this information is local with a window of 2 on the two strings.

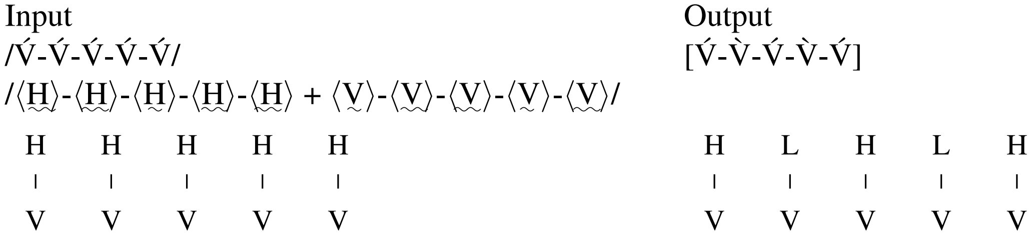
To summarize, the representation of the output *does* affect the computation of tone. Tier-conflation has subtle empirical effects (McCarthy 1986; Bat-El 1988). We showed that tier-conflation also has computational effects which impact the generative capacity or locality of tone.

5.3 Genuine non-locality in tonal inputs

So far, all the case studies were computationally local over multiple-inputs. The only problematic case was bounded Meussen's rule, but even that is local if we don't apply tier-conflation. In this section, we quickly go over one case study where the tonal phenomenon is computationally non-local regardless of how we represent the input and output.

Shona has a process of alternating Meussen's rule whereby hetero-morphemic and contiguous spans of preassociated high-toned vowels alternate to form high and low sequences: /V́-V́-V́-V́-V́/ → [V́-V̀-V́-V̀-V́].

Figure 22: *Alternating Meussen's rule in Shona* (**?**



This process is not MISL because iterative alternations are local over output information, not input information. This is the same reason why ISL is an insufficient characterization for iterative segmental rules (Chandlee et al. 2015). We conjecture that if we generalize Output Strictly Local (OSL) functions over n-ary functions, then this hypothetical class of multi-OSL functions would capture this pattern. Essentially, the function checks for the recently seen *outputted* symbols, instead of the recently seen input symbols. We leave this idea for future work.

6 Templatic phonology and prosodic morphology

The previous sections showed that tonal possesses are computationally local when computed over MT-FSTs. In the following 3 sections, we extend the analysis to Semitic root-and-pattern morphology. We first go over computational issues in the representation and derivation of templatic phonology, i.e. of prosodic morphology.

6.1 Representations in templatic phonology and morphology

Soon after the introduction of autosegmental structure for tone, McCarthy (1979, 1981) utilized these non-linear representations to model the non-concatenative phonology of root-and-pattern morphology (RPM) in Semitic. Since then, these structures have been used for processes that reference prosodic templates, such as truncation and reduplication (McCarthy and Prince 1986).

For most cases of templatic morphology, the interaction between segments and the template can be analyzed in one of three ways: a multi-input approach, concatenated approach, and derivational approach. We illustrate with the simple case of English nickname formation whereby words are truncated to the first CVC: *Jeffrey* → *Jeff*.

Figure 23: Input representations for templatic morphology

	Multi-input	Concatenated	Derivational
Input:	*Jeffrey*, CVC	CVC-*Jeffrey*	*Jeffrey*
Output:	*Jeff*	*Jeff*	*Jeff*

In the multi-input approach, the truncation function is a multi-input function over two elements: a base *Jeffrey* and a prosodic template CVC. The function outputs only the segments which are licensed by the template. In the concatenated approach, the two elements are concatenated to form a single string and submitted to a single-string function. In a sense, the template is affixed to the base. Finally in the derivational approach, the truncation function is a single-input function over the string *Jeffrey*. The function is defined to only generate truncated CVC forms. Conceptually, the first two approaches underlie item-and-arrangement theories, while the third approach is item-and-process (Hockett 1942)

Within theoretical linguistics, the multi-input approach was first proposed for Semitic RPM (McCarthy 1979), with some extensions to other templatic languages (Archangeli 1984, 1991). Since then, the multi-input and concatenated approaches have became blurred into a single concatenated approach that utilizes affixed templates. Early work in prosodic morphology argued that the template was composed of CV slots (McCarthy 1979, 1981; Marantz 1982). Later worked argued that the template was instead composed of prosodic units such as syllables and feet (McCarthy and Prince 1986). The field later shifted towards a derivational approach, whereby these templates are discarded and argued to be emergent during the phonological derivation (Ito and Mester 1992;

McCarthy and Prince 1993, 1999; Benua 1997; Downing 2006; Inkelas and Zoll 2005; Trommer 2012). Currently, attention has swung back to the concatenated approach, oftentimes with the addition of sub-segmental units like moras (Bye and Svenonius 2012; Bermúdez-Otero 2012; Saba Kirchner 2013; Trommer and Zimmermann 2014; Guekguezian 2017; Paschen 2018).

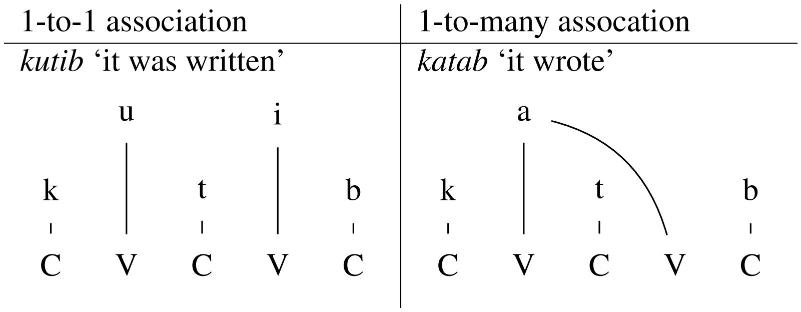
Computationally, the three approaches are distinct in terms of their input structure and history. The multi-input approach has been restricted to early work in Semitic RPM (Kay 1987; Kiraz 2001), the concatenated approach to Beesley and Karttunen (2003)'s system, and the derivational approach to Roark and Sproat (2007)'s system. However, for the latter two single-input systems, they both display computational locality for truncation. For example, Chandlee (2017) adopted a derivational approach and shows that truncation is local. But, a concatenated approach would also be computationally local as long as the template is concatenated on the same side of its prosodic target: CVC-*Jeffrey*, not *Jeffrey*-CVC.

Although truncation can be easily represented in either of the 3 formalisms, the status of Semitic root-and-pattern morphology is less clear. We turn to this topic in the next section.

6.2 Prosodic templates in root-and-pattern morphology

As is well-known, Semitic morphology uses root-and-pattern morphology (RPM) or templates. In a traditional analysis (McCarthy 1981), an Arabic word like *kutib* consists of three morphological items: a consonantal root *ktb*, vocalic inflection *ui*, and a prosodic template *CVCVC*. Contrast this with the active form of the verb *katab* where the vocalic inflection is a lone vowel *a* which undergoes one-to-many associations.[10]

Figure 24: *Associations in Semitic root-and-pattern morphology*



For Semitic RPM, there is work on designing large-scale computational resources for industrial use (Soudi et al. 2007; Farghaly and Shaalan 2009; Attia et al. 2011). Semitic RPM has been a frequent target of formal grammars, and work ranges between single-tape FSTs (Bird and Ellison 1994; Bird and Klein 1994; Walther 1998; Beesley and Karttunen 2000, 2003; Cohen-Sygal

[10] In Hebrew, some roots consists of consonants *and* vowels (Kastner 2016). This difference is computationally trivial as long the template still treats Cs and Vs differently.

and Wintner 2006; Roark and Sproat 2007; Gasser 2009), synchronous MT-FSAs (Kiraz 2000, 2001; Hulden 2009), non-deterministic asynchronous MT-FSTs (Kay 1987; Wiebe 1992), and even mildly context-sensitive acceptors (Botha and Blunsom 2013). For a review, see Kiraz (2000:92), Kiraz (2001:Ch4), and Wintner (2014:47).

But in terms understanding the formal properties of templatic morphology, there has been less work (cf. Chandlee 2017). On the one hand, the template-filling process that underlies Semitic RPM has been modeled with both the concatenated approach (Beesley and Karttunen 2003) and the derivational approach (Roark and Sproat 2007). Thus in principle, RPM can be modeled using a single-input function and computed over a single-tape FST. But on the other hand, we argue that Semitic RPM is not local over single-input representations. These linear representations cause problems in terms of state complexity or memory because they utilize a significant amount of listing, whether listing all finitely possible roots, vocalized templates, or words. We discuss this problem further in §8.3.

With this background, we argue that the locality of Semitic RPM is lost in single-input representations. In the next section, we show that the majority of RPM processes are however MISL over multi-input representations.

7 Multi-input locality in templatic phonology

Having shown the diversity of models for templatic phonology, this section focuses on the computation of Semitic root-and-pattern morphology. We show that not only is it easily amenable to multi-input representations, but it is also computationally local in most cases. We focus on Standard Arabic because of its rich array of templatic processes.

7.1 Locality of simple templatic association

The traditional autosegmental analysis of Semitic RPM lends itself to an MISL analysis with MT-FSTs. We flesh out this analysis and show that its computationally local, i.e., MISL. To generate the word *kutib*, we use 3 input strings for consonants **C**-string, vowels **V**-string, and the prosodic template **P**-string. The MT-FST in Figure 25 computes this words and uses 3 input tapes: a **C**-tape, **V**-tape, and **P**-tape.[11]

[11] We depart from Dolatian and Rawski (2020b) by calling the template string as the **P**-string instead of the **T**-string, to disambiguate from the tone **T-string**.

Figure 25: *MISL MT-FST for one-to-one template association in* kutib

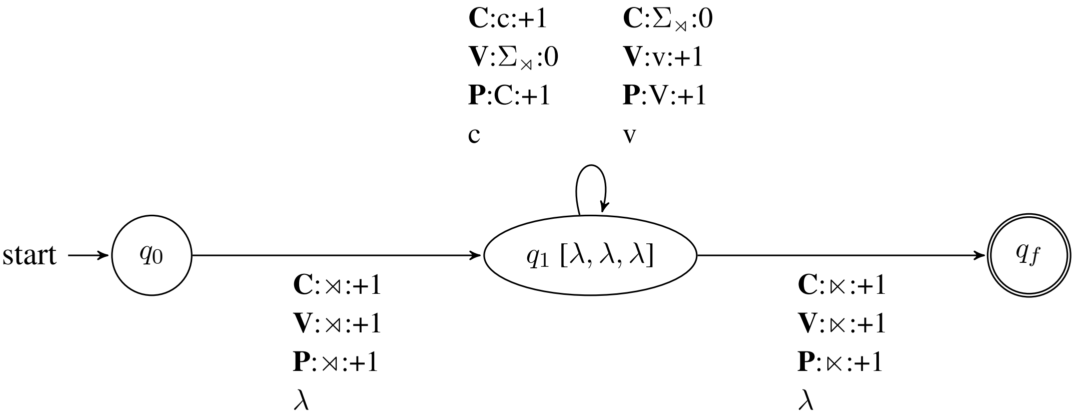


Figure 26: MT-FST for 1-to-1 slot-filling.

The word *kutib* shows one-to-one association between the vowels in the vocalism *ui* and the vowel slots in the template *CVCVC*, and between the consonants in the root *ktb* and the consonant slots in the template *CVCVC*. For the **C**-tape, its input alphabet is the set of consonants. For the **V**-tape, its alphabet is the set of vowels. And for the **P**-tape, its input alphabet is the set of CV slots. In the transition arcs, the symbol $\Sigma_{\rtimes}$ marks any input alphabet symbol including the edge boundaries. The symbols *c* and *v* denote variables over the set of possible consonants and vowels.

The brunt of the work is done by the state q_1. When it sees a consonant *c* on the **C**-tape, and consonantal slot *C* on the **P**-tape, the state simply outputs the consonant *c*. It behaves similarly for vowels. The machine does not keep track of any previously seen input symbols. Thus, this MT-FST computes a [1,1,1]-MISL function where $\vec{k} = [k_C, k_V, k_P] = [1, 1, 1]$ because the machine which doesn't memorize anything about its input. We show a derivation below for *kutib*.

Figure 27: *Derivation of* kutib *using the MISL MT-FST in Figure 26.*

	Current State	**C**-tape Arc	**C**-tape Next	**V**-tape Arc	**V**-tape Next	**P**-tape Arc	**P**-tape Next	Output Symbol	Output String
1.	q_0		$\underline{\rtimes}\text{ktb}\ltimes$		$\underline{\rtimes}\text{ui}\ltimes$		$\underline{\rtimes}\text{CVCVC}\ltimes$		
2.	q_1	$\rtimes$:+1	$\rtimes\underline{\text{k}}\text{tb}\ltimes$	$\rtimes$:+1	$\rtimes\underline{\text{u}}\text{i}\ltimes$	$\rtimes$:+1	$\rtimes\underline{\text{C}}\text{VCVC}\ltimes$	λ	
3.	q_1	k:+1	$\rtimes\text{k}\underline{\text{t}}\text{b}\ltimes$	u:0	$\rtimes\underline{\text{u}}\text{i}\ltimes$	C:+1	$\rtimes\text{C}\underline{\text{V}}\text{CVC}\ltimes$	k	k
4.	q_1	t:0	$\rtimes\text{k}\underline{\text{t}}\text{b}\ltimes$	u:+1	$\rtimes\text{u}\underline{\text{i}}\ltimes$	V:+1	$\rtimes\text{CV}\underline{\text{C}}\text{VC}\ltimes$	u	ku
5.	q_1	t:+1	$\rtimes\text{kt}\underline{\text{b}}\ltimes$	i:0	$\rtimes\text{u}\underline{\text{i}}\ltimes$	C:+1	$\rtimes\text{CVC}\underline{\text{V}}\text{C}\ltimes$	t	kut
6.	q_1	b:0	$\rtimes\text{kt}\underline{\text{b}}\ltimes$	i:+1	$\rtimes\text{ui}\underline{\ltimes}$	V:+1	$\rtimes\text{CVCV}\underline{\text{C}}\ltimes$	i	kuti
7.	q_1	b:+1	$\rtimes\text{ktb}\underline{\ltimes}$	$\ltimes$:0	$\rtimes\text{ui}\underline{\ltimes}$	C:+1	$\rtimes\text{CVCVC}\underline{\ltimes}$	b	kutib
8.	q_f	$\ltimes$:+1	$\rtimes\text{ktb}\ltimes$	$\ltimes$:+1	$\rtimes\text{ui}\ltimes$	$\ltimes$:+1	$\rtimes\text{CVCVC}\ltimes$	λ	kutib

To better understand why 1-to-1 template-filling is [1,1,1]-MISL, consider the example of an absolute neutralization rule: p→b. The segment *p* is voiced regardless of context. Not needing any context makes this rule be 1-ISL over a single-tape FST. Similarly, 1-to-1 template-filling is [1,1,1]-MISL because outputting some consonant *k* depends only on the current input symbols on the **C** and **P** tapes.

Thus, MISL MT-FSTs iconically capture the locality of simple templatic patterns in Semitic RPM. In fact, the most developed system of MT-FSTs for Semitic is Kiraz (2001). He utilized *synchronous* MT-FSTs where the tapes are simultaneously advanced. Although synchronous MT-FSTs can model Arabic morphology, they sacrifice locality. Because synchronous MT-FSTs are equivalent to single-tape FSAs, they effectively enforce non-local computation in RPM. To illustrate, Figure 28 is the derivation for *kutib* using a synchronous 4-tape MT-FSA. To avoid asynchrony, the 3 'input' tapes are aligned with the corresponding symbols on the 'output' tape by using the special symbol □ as a padding symbol (Hulden 2009).

Figure 28: *Alignment of* kutib *with a synchronous MT-FSA.*

Input Tapes	**C**:	k	□	t	□	b
	V:	□	u	□	i	□
	P:	C	V	C	V	C
Output Tape		k	u	t	i	b

The use of padding symbols however can sacrifice locality. This is clearer for cases of segmental spreading which we discuss in §7.2.2.

7.2 Locality across root-and-pattern morphology

The previous section showcased how 1-to-1 templatic-filling is computationally local when computed over multiple inputs. Arabic morphology uses many other possible combinations of roots, vowels, and templates. This section goes through a substantial chunk of Arabic morphology and shows that it is also computationally local.[12]

For illustration, we distinguish between cases of 1-to-1 association or slot-filling, and 1-to-many association. Both types are local. There are few cases of non-local computation can be given local interpretations; this concerns reduplication and loanword adaptation. For space, we refer readers to Dolatian and Rawski (2020b). When readable, we provide example MT-FSTs in the appendix.

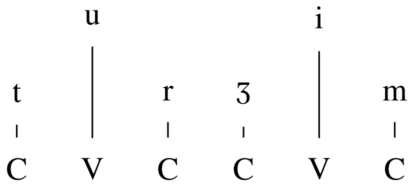
[12] We don't formalize RPM functions in broken plurals (McCarthy and Prince 1990b). Kiraz (2001:106) formalizes it as a MT-FSA which use two inputs tapes: the singular and the vocalism. The singular tape can be annotated with prosodic information. We conjecture that broken plural formation is also MISL because there are no long-distance dependencies. We leave out a full formalization for space.

7.2.1 1-to-1 slot filling

In Arabic, various patterns of 1-to-1 association are local. This includes cases of quadriliteral roots, nativization of borrowed roots, and affixed templates.

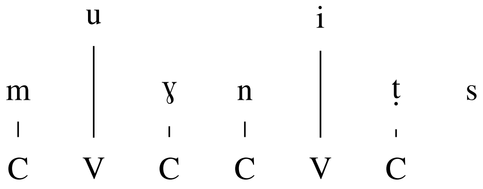
The word *kutib* represents the default case in Semitic RPM whereby a triliteral root surfaces in a CVCVC template. Some Arabic roots are however quadriliteral and have four consonants: **C**=*trʒm*. They usually take templates that have at least four consonantal slots: **P**=*CVCCVC*. When combined with the vocalism *ui*, these units generate the word *turʒim*. The generation is local and uses the same [1,1,1]-MISL MT-FST from Figure 26. A sample derivation is in the appendix (Figure 42).

Figure 29: *1-to-1 association in 4-consonant roots:* turʒim 'was translated'



Similar locality is found in words where there are more root consonants **C**=*mɣnṭs* than consonantal slots **P**=*CVCCVC* (Figure 30). The output shows deletion of the last consonant: *muɣniṭ* not **muɣniṭs*. This is also [1,1,1]-MISL. A sample MT-FST and derivation are in the appendix (Figure 43).

Figure 30: *1-to-1 association in 5-consonant roots:* maɣnaṭ 'be magnetized'



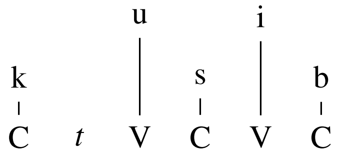
Intuitively, the machine processes the strings left-to-right. By the time we reach the final consonant *m* on the **C**-string, all the consonantal slots on the **P**-string have been associated. The machine doesn't output any consonants after reaching this point.

Lastly, just like tone, some situations involve both 1-to-1 slot-filling and pre-association. Given a root **C**=*ksb*, some outputs show an additional affix, e.g. the infix ⟨*t̰*⟩ in *k*⟨*t̰*⟩*usib*. The affix ⟨t̰⟩ is pre-associated to a slot after the first consonant. Pre-associated templates can be computed either representationally or derivationally. Both are local.[13] The representational route is to add the affix

[13] A third alternative is to treat the infix ⟨*t*⟩ as part of a separate input-string or input-tape. The template is *C*⟨***C***⟩*VCVC*

into the template: **P**=*CtVCVC*. The root and template are then combined to generate *k⟨t⟩usib*. This function is [1,1,1]-MISL. A sample MT-FST and derivation are in the appendix (Figure 44).

Figure 31: *1-to-1 association with preassociation:* k⟨t̮⟩usib 'was gained'



Informally, the machine treats the preassociated ⟨*t*⟩ as invisible to any interactions between the 3 tapes. It is ignored by the consonants on the **C**-tape, just as the **C**-tape ignores all vocalic slots on the **P**-tape. A derivational alternative is to derive *k⟨t̮⟩usib* from *kusib* by infixing ⟨*t̮*⟩. Generating *kusib* from [*ksb, ui, CVCVC*] is [1,1,1]-MISL, while infixing ⟨*t̮*⟩ is 2-ISL. The representational route is the composition of the derivational approach.

To summarize, all our cases of 1-to-1 association in Arabic RPM are computationally local over multi-input representations.

7.2.2 1-to-many slot filling

The previous cases involved 1-to-1 slot association, whereby an individual consonant or vowel became associated with only one prosodic slot. In contrast, 1-to-many association or slot-filling occurs when an input consonant or vowel gets associated with multiple prosodic slots. These processes are still computationally local but need more nuanced representations.

In some situations, Arabic RPM utilizes final spreading: *katab* (Figure 32a). The word consists of **C**=*ktb*, **V**=*a*, **P**=*CVCVC*. The vocalism **V** has only one vowel *a* because of the OCP (McCarthy 1981). The vowel is spread and associated with multiple *V* slots in the **P**-string. Computing final vowel spread is [1,2,1]-MISL with $k_V = 2$ on the **V**-string in order to find the final vowel (...v⋉). Informally, by the time we reach the second V on the **P**-string, there are no more vowels left on the **V**-string. We use our local memory to spread the previously seen vowel. A sample MT-FST and derivation are in the appendix (Figure 45).[14]

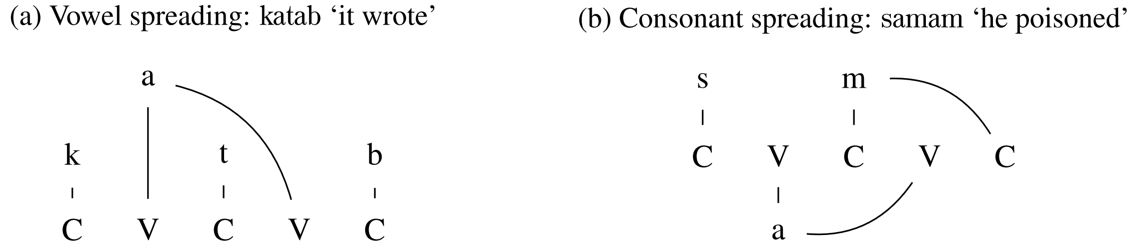
Consonants also undergo final spread in biliteral roots. The combination of **C**=*sm*, **V**=*a*, and **P**=*CVCVC* generates *samam* (Figure 32b).[15] This is [2,1,1]-MISL, analogous to the final spread of vowels except that the locality window is now larger over the **C**-string.

where ⟨**C**⟩ is pre-associated to ⟨*t*⟩. This is analogous to giving each morpheme its own autosegmental tier (McCarthy 1981). This is necessary for tonal phonology (§4.2), but there are many workarounds for templatic phonology.

[14] Interestingly, final *i* does not spread (McCarthy 1981:401). To block final *i*-spread, we need a slightly larger locality window of [1,3,1]. Alternative analyses for the final *i* have different locality effects (Dolatian and Rawski 2020b).

[15] Since McCarthy (1981), the analysis of final consonant spread has been controversial (Hudson 1986; Hoberman 1988; Yip 1988; McCarthy 1993; Gafos 1998; Bat-El 2006). Alternative analyses involving reduplication, preference

Figure 32: *1-to-many association with final spreading*


(a) Vowel spreading: katab ‘it wrote’

(b) Consonant spreading: samam ‘he poisoned’

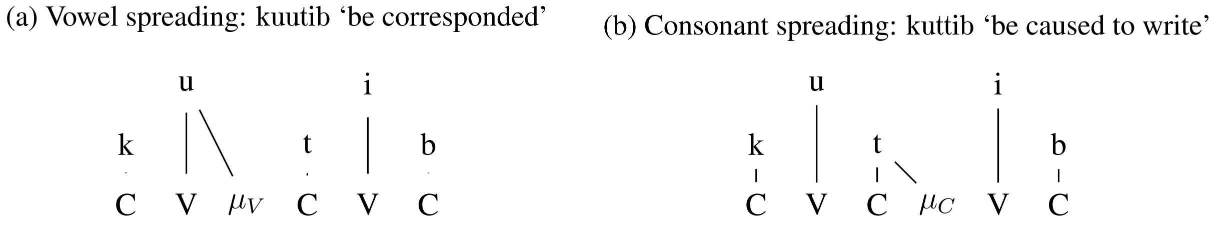
Final spreading is local (MISL) given these multiple inputs. However, these multiple inputs must be processed asynchronously in order to capture the locality. Recall that MISL functions are defined over asynchronous MT-FSTs, i.e., an MT-FST where the multiple tapes are advanced at different times. In contrast, if the inputs were processed synchronously, then the computation is non-local. The **P**-string would still CVCVC, but the input vocalism *a* would be □a□□□ (Figure 33). The zero padding □ renders this function as non-local because it separates the second V on the **P**-string from the *a* on the **V**-string.

Figure 33: *Alignment of* katab *with a synchronous MT-FSA.*

Input Tapes	**C**:	k	□	t	□	b
	V:	□	a	□	□	□
	P:	C	V	C	V	C
Output Tape		k	a	t	a	b

In contrast to final spread, medial spread involves associating a string-medial vowel or consonant to multiple slots on the **P**-string: *kuutib* with a long-vowel *u* (Figure 34a) or *kuttib* with a geminate *t* (Figure 34b). Like pre-associated affixes, medial spread can be analyzed either representationally or derivationally.

Figure 34: *1-to-many association with medial spreading*


(a) Vowel spreading: kuutib ‘be corresponded’

(b) Consonant spreading: kuttib ‘be caused to write’

For consonant spreading, the representational route involves enriching the template with a con-

for local spreading, or right-to-left association can be potentially non-local and are discussed in Dolatian and Rawski (2020b). Computationally, Beesley (1998) formalizes final spread with a special symbol X. This formalization is [2,1,1]-MISL, just like for medial spread in *kuttib*.

sonantal mora μ_C in **P**=*CVC*μ_V*VC*. Generating *kuttib* is [2,1,1]-MISL with $k_C = 2$. A sample MT-FST and derivation are in the appendix (Figure 46). Informally, whenever the machine sees μ_C on the **P**-string, it outputs the previously seen consonant on the **C**-string. Virtually the same treatment is done for vowel spreading via a vocalic mora node μ_V.

A derivational alternative is to derive *kuttib* from *kutib* by infixing a consonant mora μ_C followed by consonant spreading. Generating the base *kutib* is [1,1,1]-MISL. Infixing the mora *kut*μ_C*ib* is 4-ISL and spreading the consonant *kuttib* is 2-ISL. As with preassociation, the representational solution is a composition of the derivational solution; both are local functions.

8 Debates in Semitic root-and-pattern morphology

Analogous to tone, the computation of autosegmental structures in templatic phonology brings its own set of conceptual issues. The multi-input analysis is agnostic towards many controversies, including debates on the phonological nature of the template (§8.1), its status within the lexicon (§8.2), and its finite size (§8.3). In fact, approaches that use single-input representations rely on the finite size of templates. This makes them lose generalizations on locality.

8.1 Phonological substance of templates

For tonal phonology, the meaning of the two tiers is straightforward. This is less clear for templatic phonology. Since McCarthy (1979), there has been significant debate over the components of prosodic templates. Computationally though, these debates are tangential to locality.

In the literature, the formulation of templates is controversial (Ussishkin 2011; Bat-El 2011). One hypothesis is that the template is composed of CV slots (McCarthy 1981). Alternatives are that the template is made of prosodic units like moras, syllables, and feet (McCarthy and Prince 1990a,b), is derived from other templates via affixation (McCarthy 1993), or is a set of optimized prosodic constraints (Kastner 2016). Alternatively, words are derived words from other words via *overwriting* changing the vowels and consonants (Ussishkin 2005), e.g. *katab+ui*$\rightarrow$*kutib*. These debates on the content of prosodic templates in Semitic RPM are likewise manifested in other cases of prosodic morphology (§6.1).

In this paper, we used the first hypothesis that the template is a sequence of CV slots. But this was merely for convenience. Whether the prosodic template is up CV units, syllables, or moras is a *notational* difference (Kiraz 2001) and does not affect locality. The use of derivational affixation is analogous to function composition; it does not affect locality and is partially discussed for preassociation (§7.2.1) and medial spreading (§7.2.2). For the overwriting approach, it still requires a mechanism for placing the new segments that references slots. The function *katab+ui*$\rightarrow$*kutib* implicitly assumes that the vowels can be separated: *kVtVb+ui*. A 'template' with filled consonants *kVtVb* can be well-broken down to a root and template *ktb+CVCVC*. As for prosodic optimization, the function still needs to be well-defined over multiple inputs. This is discussed in the next section.

To summarize, by looking at templatic morphology from a mathematical perspective, the issue of the 'right' phonological components of prosodic templates does not affect locality.

8.2 Phonological emergence of templates

Similar to the previous section, a related issue is the ontological status of the template, i.e., is the template part of the lexicon (input) or does it emerge through the course of the derivation? We show that this issue has dubious computational consequences.

Our MT-FSTs took as input three morphological items, each on its own input tape. To generate *kutib*, the inputs were the root consonants **C**=*ktb*, the inflectional vowels **V**=*ui*, and a prosodic template **P**=*CVCVC*. Crucially, the template **P** was part of the input as a morphological primitive.[16] This assumption was made in earlier work on Semitic morphology (McCarthy 1981). However, recent work on Semitic argues that there is no pre-specified input template (Tucker 2010; Kastner 2016, 2019; Zukoff 2017). The same has been argued for many other cases of prosodic templates in morphology, i.e. *Generalized Template Theory* (McCarthy and Prince 1993).

In a template-less theory, the only inputs to Semitic RPM are root consonants **C**, vowels **V**, and a set of phonological constraints CON. The prosodic organization of these morphological items emerges via optimizing phonological constraints on syllable structure, autosegmental docking, and word-size requirements. To illustrate, consider a toy OT derivation for the word *kutib* in Figure 35. Constraints on syllable structure will choose the optimal candidate *kutib* for the input *ktb,ui*.

Figure 35: *OT tableau for* /ktb,ui/ *without an input template*

ktb, ui	*[CC	ONSET	CONTIGUITY
a. ☞ kutib			∗ ∗ ∗
b. ktbui	∗!		
c. uktib		∗!	

Given alternative input structure, it *seems* that MT-FSTs are now superfluous because there are no longer any templates. But this is premature. The function still takes as input two elements (a root and vocalism), and is still a multi-input function. Even though the template is emergent, there must be still be a mechanism which will interdigitate the root and vowels into this emergent template. The only complication is to define the right procedures for deterministically predicting how the segments must be ordered together. EVAL utilizes parallelist competition among multiple

[16] A possible morphosyntactic function for templates is to mark verbalization, inflectional class, or part of speech (Aronoff 1994; Kastner 2016).

candidates, which is itself a difficult process to formalize in the first place; see footnote (1).[17]

To summarize, regardless if we consider templates as morphologically primitives or phonological emergent, the templates still need to be computed in some way. Any generalizations on locality must still reference a multi-input function.

8.3 Finiteness of templates vs. infinity in the template-filling function

The third conceptual issue concerns the role of finiteness in designing grammars. In brief, there is a tug-of-war between making generalizations over finitely bounded vs. unbounded strings (Savitch 1993). It is this conflict which makes it possible to develop grammars with either of the 3 approaches reviewed in §6.1. However, these approaches differ in how much locality they can capture.

As a multi-input function, template-filling takes as input a root *ktb*, a vocalism *ui*, and an *unfilled* template *CVCVC*. Its output is the *filled* template *kutib*. However, Arabic roots are generally at most 5 segments, vocalisms at most 3 segments, and the template is at most 12 slots (McCarthy 1981). With this bound, RPM is reducible to modeling a function over a finite domain and range, i.e., a *finite* list of input-output pairs. Throughout this paper, we abstracted away from this. Our functions assume that there is no bound on the size of the root **C**, vocalism **V**, or prosodic template **P**. This allows us to treat RPM as a function over an infinitely sized domain. Doing so allows us to better capture the underlying function's generative capacity.

For example, given the *hypothetical* root consonants **C**=*ktbm*, vocalism **V**=*uaui*, and 4-syllable template **P**=*CVCVCVCV*, the same MT-FST from Figure 25 for *kutib* would output *ku.ta.bu.mi* with 1-to-1 association. But this input-output pair is *hypothetical*. All *existing* verb templates in Arabic are at most 2 syllables with additional 1 or 2 syllables for prefixation, for a total of around 10 segment slots. The MT-FSTs discussed in this paper don't enforce this bound on verb size.

However, because of the bound on the size of the **C**-,**V**-, and **P**-strings, an alternative computational implementation is a single-taped FST over a *finite* language. In a concatenated approach, the single-tape FST would take as input a single string where the 3 morphological items are concatenated: *ktb-ui-CVCVC*→*kutib*. Arabic verbs can be represented as a large finite list of inputs of the shape *root-vowels-template*. Any function with a finite domain-range is ISL over a single-taped FST. For Arabic, the single-tape FST would be ISL with a large locality window of at least size 9. This is essentially the approach taken by Beesley and Karttunen (2003), among others.

Alternatively with a derivational approach, the input language is just a list of roots. The template and vocalism are simultaneously generated in the output: *ktb*→*kutib*. If the input also includes a set of morphosyntactic features, then these features determine the choice of output: *ktb*-[PASSIVE]→*kutib*; *ktb*-[ACTIVE]→*katab*. This system works as long as there is a finite list

[17] If the input were a single concatenated string *ktb-ui*, the computation involve long-distance metathesis. If there is a bound on the original and final location of metathesized vowels, then this is local. However, the locality window would be substantially large, and at least as large as the maximum size of input roots.

of all possible vocalized templates: CaCaC, CuCiC, etc. This system would again be ISL for a potentially large locality window that matches the size of the input string. This is essentially the approach taken by Roark and Sproat (2007), among other models.

Based on the above discussion, it then appears that Semitic RPM could be adequately modeled as either a multi-input function over an MT-FST or a single-input function over a single-tape FST. The single-input function could either use a concatenated approach or a derivational approach.[18] However, using a single-input function is problematic in terms of implementation, cognition, and computation. In terms of implementation, there is a trade-off between the state explosion in single-tape FSTs vs. using richer computational structure in MT-FSTs. And in terms of cognition, the single-input approaches require that some part of Semitic morphology be finitely sized. This finite bound is the set of possible root-vocalism-template combinations for a concatenated approach, or the set of possible vocalized templates in the derivational approach. However, listing is not a cognitively insightful model. The bulk of theoretical and psycholinguistic results show that template-filling is a real process (Prunet 2006; Aronoff 2013; Kastner 2016) and it can productively and easily extend to nonce roots, loanwords, and language games (McCarthy 1981, 1986).

As for computation, the single-tape FST reduces Semitic morphology into a large but finite set of words. By being finite, any generalizations on locality are lost. In contrast, the MT-FST models an infinite function that can process an infinite set of licit combinations of consonants, vowels, and template. Of this infinite set, only a finite subset exist as real words because a filled template has at most 8 segments. This finiteness is an *independent* generalization.

Based on the above discussion, we argue that Semitic RPM shows two generalizations: one based on the potential infinity of template-filling, and the other based on the finite bound on word size. Both generalizations are part of mental grammars. To model both of these generalizations, one compromise is to use both a multi-input function and a single-input function. The multi-input function generates filled templates. It feeds its output to a single-input function which filters out all words that are larger than 10 segments. By factorizing the grammar this way, we can still maintain the generalization on locality into the multi-input part of the equation. This is similar to how haplology has been formalized in the theoretical literature (Nevins 2012). These two functions would not be composed into a single function; otherwise all generalizations are lost and the result is a finite language again.

In sum, although there are practical reasons to not use templates in Natural Language Processing, those reasons are independent of the status of templates in mental grammars. In terms of computation, abandoning template-filling requires the loss of clear locality restrictions.

[18] There is likewise another alternative. Most computational implementations avoid *directly* implementing the template-filling function by instead listing all existing *filled* templates (Buckwalter 2004). The implementation is then just a finite but large lexicon. The listing approach works in practice because roots unpredictably combine with different templates, i.e., the choice is lexicalized. But these models are unlikely to have any cognitive support.

9 Conclusion

This paper showed the salience of computational locality in nonlinear morphophonology. By extending ISL functions to consider multiple inputs, we defined the Multi-Input Strictly Local functions. The associated MISL automata elegantly characterize a majority of nonlinear suprasegmental phonology.

We demonstrated these results for tonal phonology and for root-and-pattern morphology. We provide precise discussion of the nuances of phonological and morphological substance in a theory-independent but rigorous way. We showed that locality is affected by some theoretical choices (directionality, tier-conflation), but not others (phonological content of templates). We suspect that these insights can extend to other areas in phonology (vowel harmony) and prosodic morphology.

The flexibility and precision that are given by formal language theory are not a source of methodological angst (Pater 2019). They instead provide a promising path to analyzing the nature of phonology across any domain (Heinz 2018; Jardine 2019a).

References

Archangeli, D. (1984). Underspecification in Yawelmani phonology and morphology. PhD thesis, Massachusetts Institute of Technology.

Archangeli, D. (1991). Syllabification and prosodic templates in Yawelmani. Natural Language & Linguistic Theory, 9(2):231–283.

Aronoff, M. (1994). Morphology by itself: Stems and inflectional classes. Number 22 in Linguistic Inquiry Monographs. MIT press, London/Cambridge.

Aronoff, M. (2013). The roots of language. In Cruschina, S., Maiden, M., , and Smith, J. C., editors, The boundaries of pure morphology, pages 161–180. Oxford University Press, Oxford.

Attia, M., Pecina, P., Toral, A., Tounsi, L., and van Genabith, J. (2011). An open-source finite state morphological transducer for modern standard Arabic. In Proceedings of the 9th International Workshop on Finite State Methods and Natural Language Processing, pages 125–133. Association for Computational Linguistics.

Bat-El, O. (1988). Remarks on tier conflation. Linguistic Inquiry, 19(3):477–485.

Bat-El, O. (2006). Consonant identity and consonant copy: The segmental and prosodic structure of Hebrew reduplication. Linguistic Inquiry, 37(2):179–210.

Bat-El, O. (2011). Semitic templates. In van Oostendorp et al. (2011), pages 2586–2609.

Beesley, K. (1998). Consonant spreading in Arabic stems. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on

Computational Linguistics-Volume 1, pages 117–123. Association for Computational Linguistics.

Beesley, K. and Karttunen, L. (2000). Finite-state non-concatenative morphotactics. In Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, ACL '00, pages 191–198, Hong Kong. Association for Computational Linguistics.

Beesley, K. and Karttunen, L. (2003). Finite-state morphology: Xerox tools and techniques. CSLI Publications, Stanford, CA.

Benua, L. (1997). Transderivational identity: Phonological relations between words. PhD thesis, University of Massachusetts at Amherst, Amherst, MA.

Bermúdez-Otero, R. (2012). The architecture of grammar and the division of labour in exponence. In Trommer (2012), pages 8–83.

Bird, S. (1995). Computational phonology: A constraint-based approach. Studies in Natural Language Processing. Cambridge University Press, Cambridge.

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

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

Bird, S. and Klein, E. (1994). Phonological analysis in typed feature systems. Computational linguistics, 20(3):455–491.

Botha, J. A. and Blunsom, P. (2013). Adaptor grammars for learning non-concatenative morphology. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP13), page 345–56, Stroudsburg, PA. Association for Computational Linguistics.

Buckwalter, T. (2004). Issues in Arabic orthography and morphology analysis. In Proceedings of the Workshop on Computational Approaches to Arabic Script-Based Languages, Semitic '04, page 31–34, USA. Association for Computational Linguistics.

Bye, P. and Svenonius, P. (2012). Non-concatenative morphology as epiphenomenon. In Trommer (2012), pages 427–495.

Carson-Berndsen, J. (1998). Time map phonology: Finite state models and event logics in speech recognition, volume 5 of Text, Speech and Language Technology. Kluwer Academic Publishers, Dordrecht.

Chandlee, J. (2014). Strictly Local Phonological Processes. PhD thesis, University of Delaware, Newark, DE.

Chandlee, J. (2017). Computational locality in morphological maps. Morphology, 27(4):1–43.

Chandlee, J. (2019). A computational account of tone sandhi interaction. In Proceedings of the Annual Meetings on Phonology, volume 7.

Chandlee, J., Eyraud, R., and Heinz, J. (2015). Output strictly local functions. In 14th Meeting on the Mathematics of Language, pages 112–125.

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

Chandlee, J., Heinz, J., and Jardine, A. (2018). Input strictly local opaque maps. Phonology, 35(2):171–205.

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

Chandlee, J. and Jardine, A. (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.

Cohen-Sygal, Y. and Wintner, S. (2006). Finite-state registered automata for non-concatenative morphology. Computational Linguistics, 32(1):49–82.

Coleman, J. (1998). Phonological representations: Their names, forms and powers. Cambridge University Press.

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

Dolatian, H., Koser, N., Strother-Garcia, K., and Rawski, J. (prep). Computational restrictions on iterative prosodic processes. Unpublished manuscript.

Dolatian, H. and Rawski, J. (2020a). Finite-state locality in Semitic root-and-pattern morphology. In University of Pennsylvania Working Papers in Linguistics, volume 26.

Dolatian, H. and Rawski, J. (2020b). Multi input strictly local functions for templatic morphology. In Proceedings of the Society for Computation in Linguistics, volume 3.

Downing, L. J. (2006). Canonical forms in prosodic morphology. Number 12 in Oxford studies in Theoretical Linguistics. Oxford University Press, Oxford.

Eisner, J. (1997). Efficient generation in primitive optimality theory. In 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics, pages 313–320.

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

Farghaly, A. and Shaalan, K. (2009). Arabic natural language processing: Challenges and solutions. ACM Transactions on Asian Language Information Processing (TALIP), 8(4):14.

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

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

Frank, R. and Satta, G. (1998). Optimality theory and the generative complexity of constraint violability. Computational linguistics, 24(2):307–315.

Furia, C. A. (2012). A survey of multi-tape automata. `http://arxiv.org/abs/1205.0178`. Latest revision: November 2013.

Gafos, D. (1998). Eliminating long-distance consonantal spreading. Natural Language & Linguistic Theory, 16(2):223–278.

Gasser, M. (2009). Semitic morphological analysis and generation using finite state transducers with feature structures. In Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009), pages 309–317. Association for Computational Linguistics.

Gerdemann, D. and Hulden, M. (2012). Practical finite state optimality theory. In Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing, pages 10–19.

Gibbon, D. (1987). Finite state processing of tone systems. In Third Conference of the European Chapter of the Association for Computational Linguistics, pages 291–297, Copenhagen, Denmark. Association for Computational Linguistics.

Gibbon, D. (2001). Finite state prosodic analysis of African corpus resources. In Dalsgaard, P., Lindberg, B., Benner, H., and Tan, Z., editors, EUROSPEECH 2001 Scandinavia, 7th European Conference on Speech Communication and Technology, 2nd INTERSPEECH Event, Aalborg, Denmark, September 3-7, 2001, pages 83–86. ISCA.

Goldsmith, J. (1976). Autosegmental phonology. PhD thesis, MIT.

Guekguezian, P. A. (2017). Templates as the interaction of recursive word structure and prosodic well-formedness. Phonology, 34(1):81–120.

Habash, N. and Rambow, O. (2006). Magead: A morphological analyzer and generator for the Arabic dialects. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, pages 681–688.

Hale, J. and Smolensky, P. (2006). Harmonic grammars and harmonic parsers for formal languages. In Smolensky, P. and Legendre, G., editors, The harmonic mind: From neural computation to optimality-theoretic grammar (Cognitive architecture), Vol. 1, pages 393–416. MIT Press, Cambridge, MA.

Hao, Y. (2019). Finite-state optimality theory: non-rationality of harmonic serialism. Journal of Language Modelling, 7(2):49–99.

Heinz, J. (2018). The computational nature of phonological generalizations. In Hyman, L. and Plank, F., editors, Phonological Typology, Phonetics and Phonology, chapter 5, pages 126–195. Mouton de Gruyter, Berlin.

Heinz, J., Kobele, G. M., and Riggle, J. (2009). Evaluating the complexity of optimality theory. Linguistic Inquiry, 40(2):277–288.

Heinz, J. and Lai, R. (2013). Vowel harmony and subsequentiality. In Kornai, A. and Kuhlmann, M., editors, Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13), pages 52–63, Sofia, Bulgaria. Association for Computational Linguistics.

Hoberman, R. D. (1988). Local and long-distance spreading in Semitic morphology. Natural Language & Linguistic Theory, 6(4):541–549.

Hockett, C. F. (1942). A system of descriptive phonology. Language, 18:3–21.

Howard, I. (1972). A directional theory of rule application in phonology. PhD thesis, Massachusetts Institute of Technology.

Hudson, G. (1986). Arabic root and pattern morphology without tiers. Journal of Linguistics, 22(1):85–122.

Hulden, M. (2009). Revisiting multi-tape automata for Semitic morphological analysis and generation. In Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic Languages, pages 19–26. Association for Computational Linguistics.

Hulden, M. (2017a). Formal and computational verification of phonological analyses. Phonology, 34(2):407–435.

Hulden, M. (2017b). Rewrite rule grammars with multitape automata. Journal of Language Modelling, 5(1):107–130.

Hyman, L. (2011). Tone: Is it different? In Goldsmith, J., Riggle, J., and Yu, A. C. L., editors, The Handbook of Phonological Theory, volume 4, pages 197–239. Blackwell, Oxford, 2 edition.

Idsardi, W. J. (2006). A simple proof that optimality theory is computationally intractable. Linguistic Inquiry, 37(2):271–275.

Inkelas, S. and Zoll, C. (2005). Reduplication: Doubling in Morphology. Cambridge University Press, Cambridge.

Itô, J. (1989). A prosodic theory of epenthesis. Natural Language & Linguistic Theory, 7(2):217–259.

Ito, J. and Mester, A. (1992). Weak layering and word binarity. Linguistic Research Center Technical Repoty (LRC-92-09).

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

Jardine, A. (2016b). Locality and non-linear representations in tonal phonology. PhD thesis, University of Delaware, Newark, DE.

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

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

Jardine, A. (2019a). Computation also matters: a response to pater (2018). Phonology, 36(2):341–350.

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

Jardine, A. (2020). Melody learning and long-distance phonotactics in tone. Natural Language & Linguistic Theory, 38:1145–1195.

Jardine, A., Danis, N., and Iacoponi, L. (2020). A formal investigation of Q-theory in comparison to autosegmental representations. Linguistic Inquiry.

Kaplan, R. M. and Kay, M. (1994). Regular models of phonological rule systems. Computational linguistics, 20(3):331–378.

Karttunen, L. (1998). The proper treatment of optimality in computational phonology. Proceedings of the International Workshop on Finite State Methods in Natural Language Processing, pages 1–12.

Karttunen, L. (2006a). A finite-state approximation of optimality theory: The case of Finnish prosody. In Advances in Natural Language Processing, pages 4–15. Springer.

Karttunen, L. (2006b). The insufficiency of paper-and-pencil linguistics: The case of Finnish prosody. In Butt, M., Dalrymple, M., , and King, T. H., editors, Intelligent linguistic architectures: Variations on themes by Ronald M. Kaplan, number 179 in CSLI Lecture Notes, pages 287–300. CSLI, Stanford, CA.

Kastner, I. (2016). Form and meaning in the Hebrew verb. PhD thesis, New York University.

Kastner, I. (2019). Templatic morphology as an emergent property. Natural Language & Linguistic Theory, 37(2):571–619.

Kay, M. (1987). Nonconcatenative finite-state morphology. In Third Conference of the European Chapter of the Association for Computational Linguistics, Copenhagen, Denmark. Association for Computational Linguistics.

Kiraz, G. A. (2000). Multitiered nonlinear morphology using multitape finite automata: A case study on Syriac and Arabic. Computational Linguistics, 26(1):77–105.

Kiraz, G. A. (2001). Computational nonlinear morphology: With emphasis on Semitic languages. Cambridge University Press.

Kornai, A. (1995). Formal phonology. Garland Publishing, New York.

Koser, N., Oakden, C., and Jardine, A. (2019). Tone association and output locality in non-linear structures. In Supplemental proceedings of AMP 2019.

Lamont, A. (prep). Optimizing over subsequences generates context-sensitive languages. Unpublished manuscript.

Leben, W. R. (1973). Suprasegmental phonology. PhD thesis, Massachusetts Institute of Technology.

Mamadou, T. and Jardine, A. (2020). Representation and the computation of long distance tone processes. In Proceedings of the Annual Meetings on Phonology, volume 8.

Marantz, A. (1982). Re reduplication. Linguistic Inquiry, 13(3):435–482.

McCarthy, J. and Prince, A. (1990a). Prosodic morphology and templatic morphology. In Perspectives on Arabic linguistics II: Papers from the second annual symposium on Arabic linguistics, pages 1–54. John Benjamins Amsterdam.

McCarthy, J. J. (1979). Formal problems in Semitic phonology and morphology. PhD thesis, Massachusetts Institute of Technology.

McCarthy, J. J. (1981). A prosodic theory of nonconcatenative morphology. Linguistic Inquiry, 12(3):373–418.

McCarthy, J. J. (1986). OCP effects: Gemination and antigemination. Linguistic inquiry, 17(2):207–263.

McCarthy, J. J. (1993). Template form in prosodic morphology. In Proceedings of the Formal Linguistics Society of Mid-America, volume 3, pages 187–218.

McCarthy, J. J. and Prince, A. (1986). Prosodic morphology. Unpublished manuscript.

McCarthy, J. J. and Prince, A. (1993). Prosodic morphology I: Constraint interaction and satisfaction.

McCarthy, J. J. and Prince, A. (1999). Faithfulness and identity in prosodic morphology. In Kager, R., van der Hulst, H., and Zonneveld, W., editors, The prosody-morphology interface, pages 218–309. Cambridge University Press, Cambridge.

McCarthy, J. J. and Prince, A. S. (1990b). Foot and word in prosodic morphology: The Arabic broken plural. Natural Language & Linguistic Theory, 8(2):209–283.

McCollum, A. G., Baković, E., Mai, A., and Meinhardt, E. (2020). Unbounded circumambient patterns in segmental phonology. Phonology, 37(2):215–255.

Nevins, A. (2012). Haplological dissimilation at distinct stages of exponence. In Trommer (2012), pages 84–116.

Oakden, C. (2020). Notational equivalence in tonal geometry. Phonology, 37:257–296.

Odden, D. (1994). Adjacency parameters in phonology. Language, 70(2):289–330.

Paschen, L. (2018). The interaction of reduplication and segmental mutation: A phonological account. PhD thesis, Universität Leipzig.

Pater, J. (2018). Substance matters: a reply to Jardine (2016). Phonology, 35(1):151–156.

Pater, J. (2019). Phonological typology in optimality theory and formal language theory: goals and future directions. Phonology, 36(2):351–353.

Prunet, J.-F. (2006). External evidence and the Semitic root. Morphology, 16(1):41.

Rabin, M. O. and Scott, D. (1959). Finite automata and their decision problems. IBM Journal of Research and Development, 3(2):114–125.

Rawski, J. and Dolatian, H. (2020). Multi input strictly local functions for tonal phonology. In Proceedings of the Society for Computation in Linguistics, volume 3.

Riggle, J. A. (2004). Generation, recognition, and learning in finite state Optimality Theory. PhD thesis, University of California, Los Angeles.

Roark, B. and Sproat, R. (2007). Computational Approaches to Morphology and Syntax. Oxford University Press, Oxford.

Roche, E. and Schabes, Y., editors (1997). Finite-state language processing. MIT press.

Saba Kirchner, J. (2013). Minimal reduplication and reduplicative exponence. Morphology, 23(2):227–243.

Savitch, W. J. (1993). Why it might pay to assume that languages are infinite. Annals of Mathematics and Artificial Intelligence, 8(1-2):17–25.

Shih, S. S. and Inkelas, S. (2018). Autosegmental aims in surface-optimizing phonology. Linguistic Inquiry, 50(1):137–196.

Shu, H. (2006). Multi-tape finite-state transducer for asynchronous multi-stream pattern recognition with application to speech. PhD thesis, Massachusetts Institute of Technology.

Smolensky, P. (1993). Harmonic grammars for formal languages. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, Advances in neural information processing systems 5, pages 847–854, San Mateo. Morgan Kauffman.

Soudi, A., Neumann, G., and Van den Bosch, A., editors (2007). Arabic computational morphology: Knowledge-based and empirical methods. Springer.

Strother-Garcia, K. (2019). Using model theory in phonology: a novel characterization of syllable structure and syllabification. PhD thesis, University of Delaware.

Trommer, J. (2012). The morphology and phonology of exponence. Number 41 in Oxford Studies in Theoretical Linguistics. Oxford University Press, Oxford.

Trommer, J. and Zimmermann, E. (2014). Generalised mora affixation and quantity-manipulating morphology. Phonology, pages 463–510.

Tucker, M. A. (2010). Roots and prosody: The Iraqi Arabic derivational verb. Recherches linguistiques de Vincennes, (39):31–68.

Ussishkin, A. (2005). A fixed prosodic theory of nonconcatenative templaticmorphology. Natural Language & Linguistic Theory, 23(1):169–218.

Ussishkin, A. (2011). Tier segregation. In van Oostendorp et al. (2011), pages 2516–2535.

van Oostendorp, M., Ewen, C., Hume, E., and Rice, K., editors (2011). The Blackwell companion to phonology. Wiley-Blackwell, Malden, MA.

Walther, M. (1998). Computing declarative prosodic morphology. In SIGPHON'98 The Computation of Phonological Constraints.

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

Williams, E. S. (1976). Underlying tone in Margi and Igbo. Linguistic Inquiry, pages 463–484.

Wintner, S. (2014). Morphological processing of Semitic languages. In Zitouni, I., editor, Natural language processing of Semitic languages, pages 43–66. Springer.

Yip, M. (1988). Template morphology and the direction of association. Natural Language & Linguistic Theory, 6(4):551–577.

Yip, M. (2002). Cambridge University Press, Cambridge.

Yli-Jyrä, A. (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.

Yli-Jyrä, A. (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).

Yli-Jyrä, A. (2019). Optimal Kornai-Karttunen codes for restricted autosegmental representations. In Condoravdi, C. and King, T. H., editors, Tokens of Meaning: Papers in Honor of Lauri Karttunen. Center for the Study of Language and Information (CSLI).

Zhu, Y. (2020). Extending the autosegmental input strictly local framework: Metrical dominance and floating tones. In Proceedings of the Society for Computation in Linguistics, volume 3.

Zukoff, S. (2017). Arabic nonconcatenative morphology and the syntax-phonology interface. In NELS 47: Proceedings of the Forty-Seventh Annual Meeting of the North East Linguistic Society, volume 3, pages 295—-314, Amherst, MA. Graduate Linguistics Student Association.

A Appendix

This appendix provides sample MISL MT-FSTs and derivations for some of the processes discussed in the paper. We constructed MT-FSTs only for the processes where the locality windows were small enough to make them readable.

A.1 Initial and final states

The main paper used the unique state q_0, q_f as the initial and final state respectively. In the appendix, we do not show these two states because of the size of our MT-FST. But they are still used in the derivation tables.

Every MT-FST has a unique state q_0. For tone-based MT-FSTs, there is always the following transition arc between q_0 and q_1:

T:$\rtimes$:+1

V:$\rtimes$:+1

λ

For RPM-based MT-FSTs, there is always the following transition arc between q_0 and q_1:

C:$\rtimes$:+1

V:$\rtimes$:+1

P:$\rtimes$:+1

λ

We do not show the final state q_f. Instead, a state p is marked as final (with double circles) iff there is the following implied transition arc between p and q_f for tone-based MT-FSTs:

T:$\ltimes$:+1

V:$\ltimes$:+1

λ

Similarly, RPM-based MT-FSTs have the following transition arc between 'final states' and q_f:

C:$\ltimes$:+1

V:⋉:+1

P:⋉:+1

λ

For MT-FSTs which process the string right-to-left, the location of the ⋊ and ⋉ symbols are switched, and +1 is replaced with -1

A.2 Appendix of transducers for tone

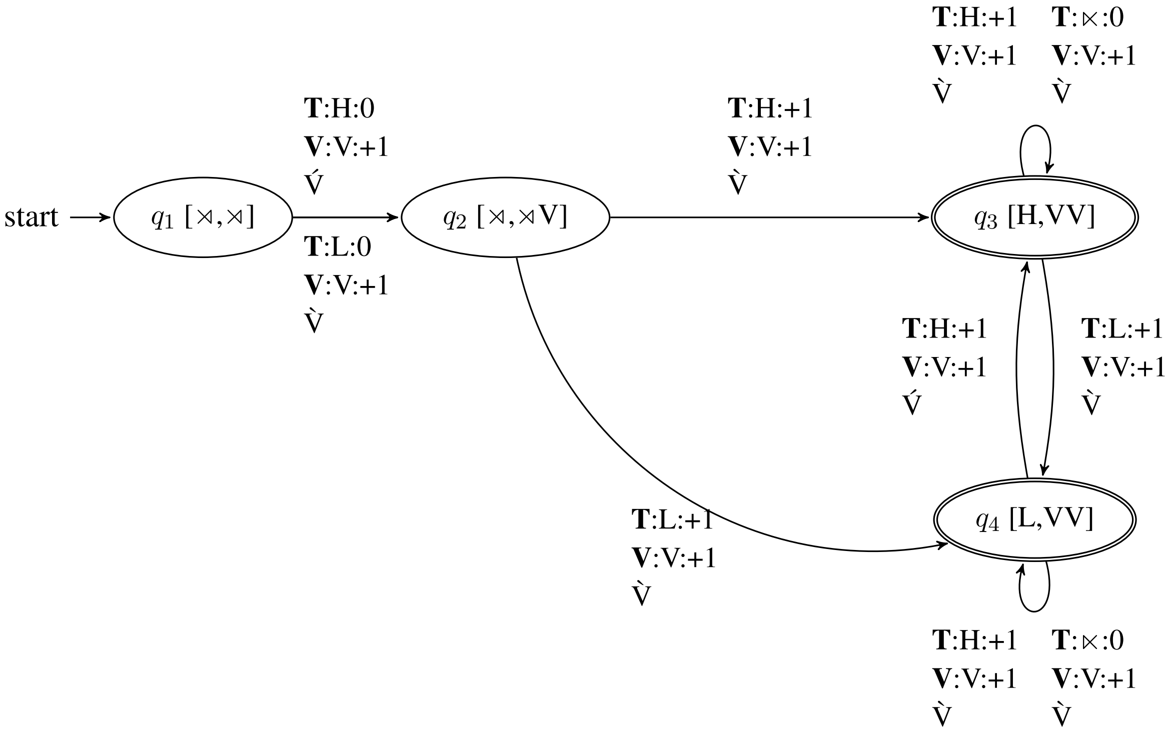
A sample MT-FST and derivation are given for some of the tone processes from §4.1 and §4.3. These include patterns with preassociation, and without preassocation.

A.2.1 Kikuyu limited spreading

In Kikuyu (Figure 10), the first tone associates with the first two vowels. Left-to-right spreading follows. A final tone may undergo final spreading, e.g. /LHLH + VVVVVVV/ is mapped to V̀V̀V́V̀V́V́V́. A [2,3]-MISL MT-FST is provided in Figure 36a, with a sample derivation in Figure 36b.

Figure 36: *Limited spreading in Kikuyu*

(a) *MT-FST for Kikuyu limited spread*



(b) *Derivation of* /LHLH + VVVVVVV/→V̀V̀V́V̀V́V́V́ *in Kikuyu*

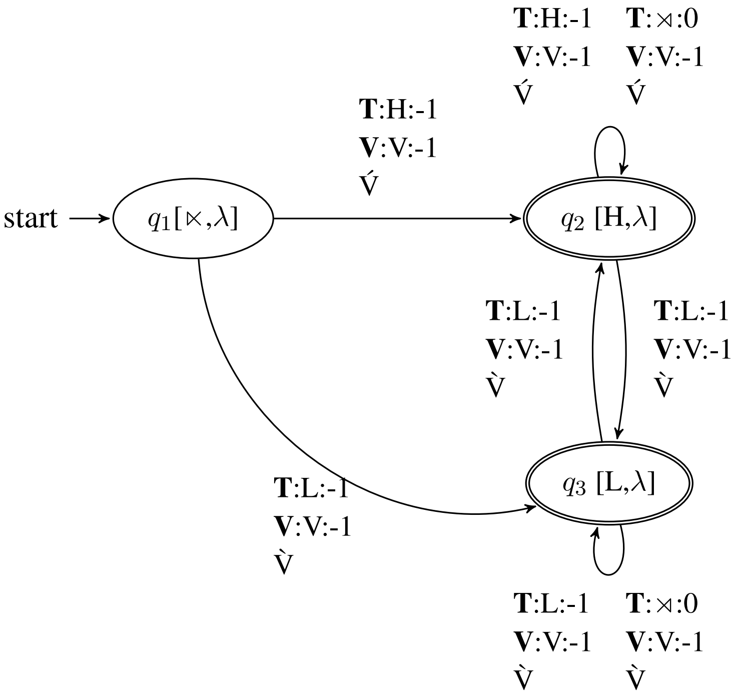
	Current State	**T**-tape Arc	**T**-tape Next	**V**-tape Arc	**V**-tape Next	Output symbol	Output String
1.	q_0		$\underline{\rtimes}\text{LHLH}\ltimes$		$\underline{\rtimes}\text{VVVVVVV}\ltimes$		
2.	q_1	$\rtimes$:+1	$\rtimes\underline{\text{L}}\text{HLH}\ltimes$	$\rtimes$:+1	$\rtimes\underline{\text{V}}\text{VVVVVV}\ltimes$	λ	
3.	q_2	L:0	$\rtimes\underline{\text{L}}\text{HLH}\ltimes$	V:+1	$\rtimes\text{V}\underline{\text{V}}\text{VVVVV}\ltimes$	V̀	V̀
4.	q_4	L:+1	$\rtimes\text{L}\underline{\text{H}}\text{LH}\ltimes$	V:+1	$\rtimes\text{VV}\underline{\text{V}}\text{VVVV}\ltimes$	V̀	V̀V̀
5.	q_3	H:+1	$\rtimes\text{LH}\underline{\text{L}}\text{H}\ltimes$	V:+1	$\rtimes\text{VV}\langle\utilde{\text{V}}\rangle\underline{\text{V}}\text{VVV}\ltimes$	V́	V̀V̀V́
6.	q_4	L:+1	$\rtimes\text{LHL}\underline{\text{H}}\ltimes$	V:+1	$\rtimes\text{VV}\langle\utilde{\text{V}}\rangle\text{V}\underline{\text{V}}\text{VV}\ltimes$	V̀	V̀V̀V́V̀
7.	q_3	H:+1	$\rtimes\text{LHLH}\underline{\ltimes}$	V:+1	$\rtimes\text{VV}\langle\utilde{\text{V}}\rangle\text{VV}\underline{\text{V}}\text{V}\ltimes$	V́	V̀V̀V́V̀V́
8.	q_3	$\ltimes$:0	$\rtimes\text{LHLH}\underline{\ltimes}$	V:+1	$\rtimes\text{VV}\langle\utilde{\text{V}}\rangle\text{VVV}\underline{\text{V}}\ltimes$	V́	V̀V̀V́V̀V́V́
9.	q_3	$\ltimes$:0	$\rtimes\text{LHLH}\underline{\ltimes}$	V:+1	$\rtimes\text{VV}\langle\utilde{\text{V}}\rangle\text{VVVV}\underline{\ltimes}$	V́	V̀V̀V́V̀V́V́V́
10.	q_f	$\ltimes$:+1	$\rtimes\text{LHLH}\ltimes$	$\ltimes$:+1	$\rtimes\text{VV}\langle\utilde{\text{V}}\rangle\text{VVVV}\ltimes$	λ	V̀V̀V́V̀V́V́V́

A.2.2 Hausa right-to-left spreading

In Hausa (Figure 11), tones are associated right-to-left with initial spread: /LH, VVV/ is mapped to V̀V̀V́. This function is modeled by the [2,1]-MISL MT-FST in Figure 37a, with a sample derivation in Figure 36b. The FST processes the input string-tuple from right to left using the -1 direction parameter.

Figure 37: *Right-to-left spreading in Hausa*

(a) *MT-FST for Hausa right-to-left spreading*



(b) *Derivation of* /LH + VVV/→V̀V̀V́ *in Hausa*

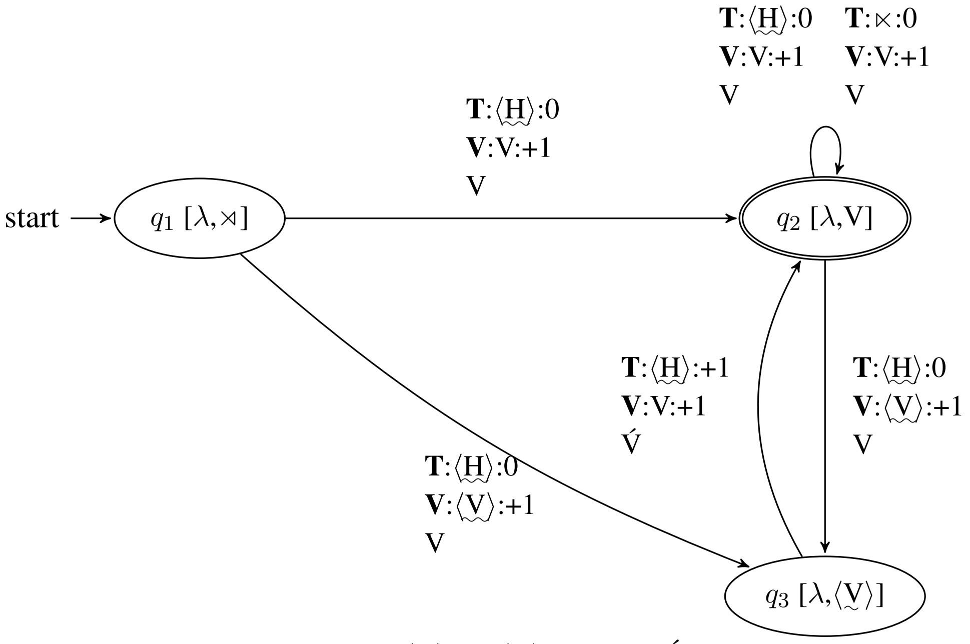
	Current State	**T**-tape Arc	**T**-tape Next	**V**-tape Arc	**V**-tape Next	Output symbol	Output String
1.	q_0		$\rtimes\mathrm{LH}\underline{\ltimes}$		$\rtimes\mathrm{VVV}\underline{\ltimes}$		
2.	q_1	$\ltimes$:-1	$\rtimes\mathrm{L}\underline{\mathrm{H}}\ltimes$	$\ltimes$:-1	$\rtimes\mathrm{VV}\underline{\mathrm{V}}\ltimes$	λ	
3.	q_2	H:-1	$\rtimes\underline{\mathrm{L}}\mathrm{H}\ltimes$	V:-1	$\rtimes\mathrm{V}\underline{\mathrm{V}}\mathrm{V}\ltimes$	V́	V́
4.	q_3	L:-1	$\underline{\rtimes}\mathrm{LH}\ltimes$	V:-1	$\rtimes\underline{\mathrm{V}}\mathrm{VV}\ltimes$	V̀	V̀V́
5.	q_3	$\rtimes$:0	$\underline{\rtimes}\mathrm{LH}\ltimes$	V:-1	$\underline{\rtimes}\mathrm{VVV}\ltimes$	V̀	V̀V̀V́
6.	q_f	$\rtimes$:-1	$\rtimes\mathrm{LH}\ltimes$	$\rtimes$:-1	$\rtimes\mathrm{VVV}\ltimes$	λ	V̀V̀V́

A.2.3 Rimi bounded tone shift

In Rimi (Figure 15), a preassociated tone will shift one vowel to the right: $/\langle\utilde{\mathrm{H}}\rangle + \mathrm{V}\langle\utilde{\mathrm{V}}\rangle\mathrm{VV}/$ is mapped to VVV́V. This function is modeled by the [1,2]-MISL MT-FST in Figure 38a, with a sample derivation in Figure 38b. We assume that the only possible underlying tone string is a string of preassociated H tones.[19]

Figure 38: *Bounded tone shift in Rimi*

(a) *MT-FST for Rimi*


start
q_1 [λ,⋊]
q_2 [λ,V]
q_3 [λ,$\langle\utilde{\text{V}}\rangle$]
T:$\langle\utilde{\text{H}}\rangle$:0
V:V:+1
V
T:$\langle\utilde{\text{H}}\rangle$:0
V:V:+1
V
T:⋉:0
V:V:+1
V
T:$\langle\utilde{\text{H}}\rangle$:+1
V:V:+1
V́
T:$\langle\utilde{\text{H}}\rangle$:0
V:$\langle\utilde{\text{V}}\rangle$:+1
V
T:$\langle\utilde{\text{H}}\rangle$:0
V:$\langle\utilde{\text{V}}\rangle$:+1
V

(b) *Derivation of* $/\langle\utilde{\text{H}}\rangle + \text{V}\langle\utilde{\text{V}}\rangle\text{VV}/\rightarrow$VVV́V *in Rimi*

	Current State	**T**-tape Arc	**T**-tape Next	**V**-tape Arc	**V**-tape Next	Output symbol	Output String
1.	q_0		$\underline{\rtimes}\langle\utilde{\text{H}}\rangle\ltimes$		$\underline{\rtimes}\text{V}\langle\utilde{\text{V}}\rangle\text{VV}\ltimes$		
2.	q_1	$\rtimes$:+1	$\rtimes\underline{\langle\utilde{\text{H}}\rangle}\ltimes$	$\rtimes$:+1	$\rtimes\underline{\text{V}}\langle\utilde{\text{V}}\rangle\text{VV}\ltimes$	λ	
3.	q_2	$\langle\utilde{\text{H}}\rangle$:0	$\rtimes\underline{\langle\utilde{\text{H}}\rangle}\ltimes$	V:+1	$\rtimes\text{V}\underline{\langle\utilde{\text{V}}\rangle}\text{VV}\ltimes$	V	V
4.	q_3	$\langle\utilde{\text{H}}\rangle$:0	$\rtimes\underline{\langle\utilde{\text{H}}\rangle}\ltimes$	$\langle\utilde{\text{V}}\rangle$:+1	$\rtimes\text{V}\langle\utilde{\text{V}}\rangle\underline{\text{V}}\text{V}\ltimes$	V	VV
5.	q_2	$\langle\utilde{\text{H}}\rangle$:+1	$\rtimes\langle\utilde{\text{H}}\rangle\underline{\ltimes}$	V:+1	$\rtimes\text{V}\langle\utilde{\text{V}}\rangle\text{V}\underline{\text{V}}\ltimes$	V́	VVV́
6.	q_2	$\ltimes$:0	$\rtimes\langle\utilde{\text{H}}\rangle\underline{\ltimes}$	V:+1	$\rtimes\text{V}\langle\utilde{\text{V}}\rangle\text{VV}\underline{\ltimes}$	V	VVV́V
7.	q_f	$\ltimes$:+1	$\rtimes\langle\utilde{\text{H}}\rangle\ltimes$	$\ltimes$:+1	$\rtimes\text{V}\langle\utilde{\text{V}}\rangle\text{VV}\ltimes$	λ	VVV́V

A.2.4 Zigulu unbounded tone shift

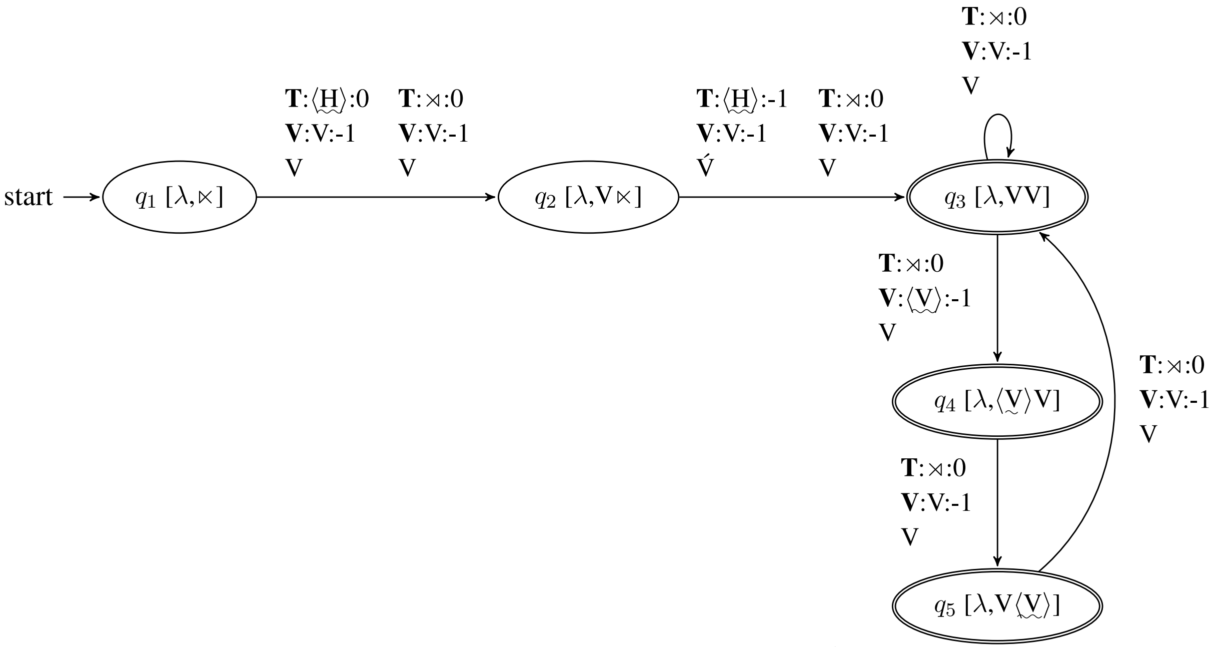
In Zigulu (Figure 16), unbounded tone shift causes a preassociated H tone to shift to the penultimate vowel: $/\langle\utilde{\text{H}}\rangle + \text{VV}\langle\utilde{\text{V}}\rangle\text{VVV}/$ maps to VVVVV́V. This function is modeled by the [1,3]-MISL MT-FST in Figure 39a, with a sample derivation in Figure 39b. For easier illustration, the MT-FST

[19] Final preassociated vowels do not undergo tone shift: $/\langle\utilde{\text{H}}\rangle + \text{VVV}\langle\utilde{\text{V}}\rangle/\rightarrow$VVVV́. We factor this out for illustrative reasons. Otherwise, the function is [2,2]-MISL and needs a MT-FST with more states.

processes the input right-to-left using the -1 direction parameter. We assume that the tone string can either be an empty string $\rtimes\ltimes$ or a single preassociated H tone $\rtimes\langle\underline{\mathrm{H}}\rangle\ltimes$.[20]

Figure 39: *Unbounded tone shift in Zigulu*

(a) *MT-FST for Zigulu*



(b) *Derivation of* /$\langle\underset{\sim}{\mathrm{H}}\rangle$ + VV$\langle\underset{\sim}{\mathrm{V}}\rangle$VVV/→VVVVV́V *in Zigulu*

	Current State	**T**-tape Arc	**T**-tape Next	**V**-tape Arc	**V**-tape Next	Output symbol	Output String
1.	q_0		$\rtimes\langle\underset{\sim}{\mathrm{H}}\rangle\underline{\ltimes}$		$\rtimes\mathrm{VV}\langle\underset{\sim}{\mathrm{V}}\rangle\mathrm{VVV}\underline{\ltimes}$		
2.	q_1	$\ltimes$:-1	$\rtimes\underline{\langle\underset{\sim}{\mathrm{H}}\rangle}\ltimes$	$\ltimes$:-1	$\rtimes\mathrm{VV}\langle\underset{\sim}{\mathrm{V}}\rangle\mathrm{VV}\underline{\mathrm{V}}\ltimes$	λ	
3.	q_2	$\langle\underset{\sim}{\mathrm{H}}\rangle$:0	$\rtimes\underline{\langle\underset{\sim}{\mathrm{H}}\rangle}\ltimes$	V:-1	$\rtimes\mathrm{VV}\langle\underset{\sim}{\mathrm{V}}\rangle\mathrm{V}\underline{\mathrm{V}}\mathrm{V}\ltimes$	V	V
4.	q_3	$\langle\underset{\sim}{\mathrm{H}}\rangle$:-1	$\underline{\rtimes}\langle\underset{\sim}{\mathrm{H}}\rangle\ltimes$	V:-1	$\rtimes\mathrm{VV}\langle\underset{\sim}{\mathrm{V}}\rangle\underline{\mathrm{V}}\mathrm{VV}\ltimes$	V́	V́V
5.	q_3	$\rtimes$:0	$\underline{\rtimes}\langle\underset{\sim}{\mathrm{H}}\rangle\ltimes$	V:-1	$\rtimes\mathrm{VV}\underline{\langle\underset{\sim}{\mathrm{V}}\rangle}\mathrm{VVV}\ltimes$	V	VV́V
6.	q_4	$\rtimes$:0	$\underline{\rtimes}\langle\underset{\sim}{\mathrm{H}}\rangle\ltimes$	$\langle\underset{\sim}{\mathrm{V}}\rangle$:-1	$\rtimes\mathrm{V}\underline{\mathrm{V}}\langle\underset{\sim}{\mathrm{V}}\rangle\mathrm{VVV}\ltimes$	V	VVV́V
7.	q_5	$\rtimes$:0	$\underline{\rtimes}\langle\underset{\sim}{\mathrm{H}}\rangle\ltimes$	V:-1	$\rtimes\underline{\mathrm{V}}\mathrm{V}\langle\underset{\sim}{\mathrm{V}}\rangle\mathrm{VVV}\ltimes$	V	VVVV́V
8.	q_3	$\rtimes$:0	$\underline{\rtimes}\langle\underset{\sim}{\mathrm{H}}\rangle\ltimes$	V:-1	$\underline{\rtimes}\mathrm{VV}\langle\underset{\sim}{\mathrm{V}}\rangle\mathrm{VVV}\ltimes$	V	VVVVV́V
9.	q_f	$\rtimes$:-1	$\rtimes\langle\underset{\sim}{\mathrm{H}}\rangle\ltimes$	$\rtimes$:-1	$\rtimes\mathrm{VV}\langle\underset{\sim}{\mathrm{V}}\rangle\mathrm{VVV}\ltimes$	λ	VVVVV́V

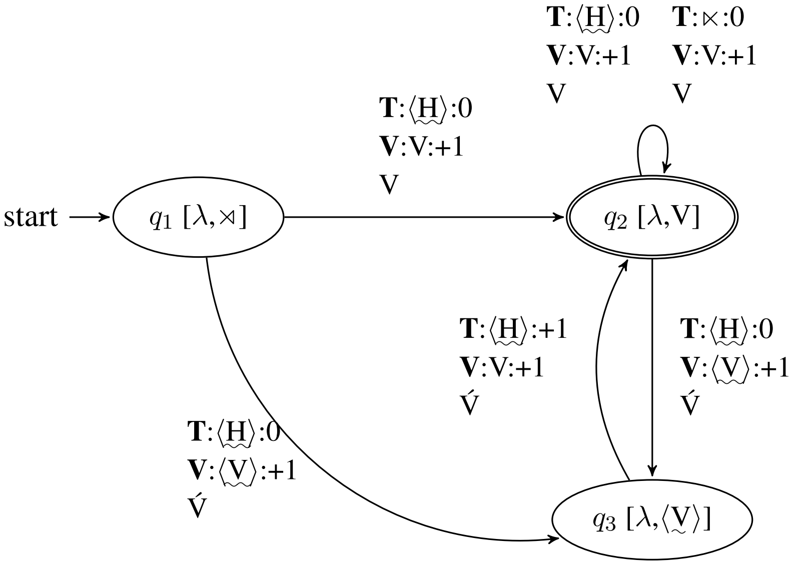
[20] Note that this MISL MT-FST cannot guarantee that there exists a preassociated vowel $\langle\underset{\sim}{\mathrm{V}}\rangle$ for every preassociated tone ⟨H⟩. It would map /$\langle\underset{\sim}{\mathrm{H}}\rangle$ + VVVV/ to VVV́V. This is because the distance between the preassociated vowel and the penultimate vowel is unbounded.

A.2.5 Bemba unbounded tone spread

In Bemba (Figure 17), bounded tone spread causes a preassociated H tone to surface on its preassociated vowel and on the subsequent vowel: /⟨H̰⟩ + V⟨V̰⟩VV/ maps to VV́V́V. This function is modeled by the [1,2]-MISL MT-FST in Figure 40a, with a sample derivation in Figure 40b. We assume that the input tone string contains either an empty string ⋊⋉ or a sequence of preassociated H tones ⋊⟨H̲⟩*⋉.

Figure 40: *Unbounded tone spread in Bemba*

(a) *MT-FST for Bemba*



(b) *Derivation of* /⟨H̰⟩ + V⟨V̰⟩VV/→VV́V́V *in Bemba*

	Current State	**T**-tape Arc	**T**-tape Next	**V**-tape Arc	**V**-tape Next	Output symbol	Output String
1.	q_0		$\underline{⋊}$⟨H̰⟩⋉		$\underline{⋊}$V⟨V̰⟩VV⋉		
2.	q_1	⋊:+1	⋊$\underline{⟨H̰⟩}$⋉	⋊:+1	⋊$\underline{V}$⟨V̰⟩VV⋉	λ	
3.	q_2	⟨H̰⟩:0	⋊$\underline{⟨H̰⟩}$⋉	V:+1	⋊V$\underline{⟨V̰⟩}$VV⋉	V	V
4.	q_3	⟨H̰⟩:0	⋊$\underline{⟨H̰⟩}$⋉	⟨V̰⟩:+1	⋊V⟨V̰⟩$\underline{V}$V⋉	V	VV́
5.	q_2	⟨H̰⟩:+1	⋊⟨H̰⟩$\underline{⋉}$	V:+1	⋊V⟨V̰⟩V$\underline{V}$⋉	V́	VV́V́
6.	q_2	⋉:0	⋊⟨H̰⟩$\underline{⋉}$	V:+1	⋊V⟨V̰⟩VV$\underline{⋉}$	V	VV́V́V
7.	q_f	⋉:+1	⋊⟨H̰⟩⋉	⋉:+1	⋊V⟨V̰⟩VV⋉	λ	VV́V́V

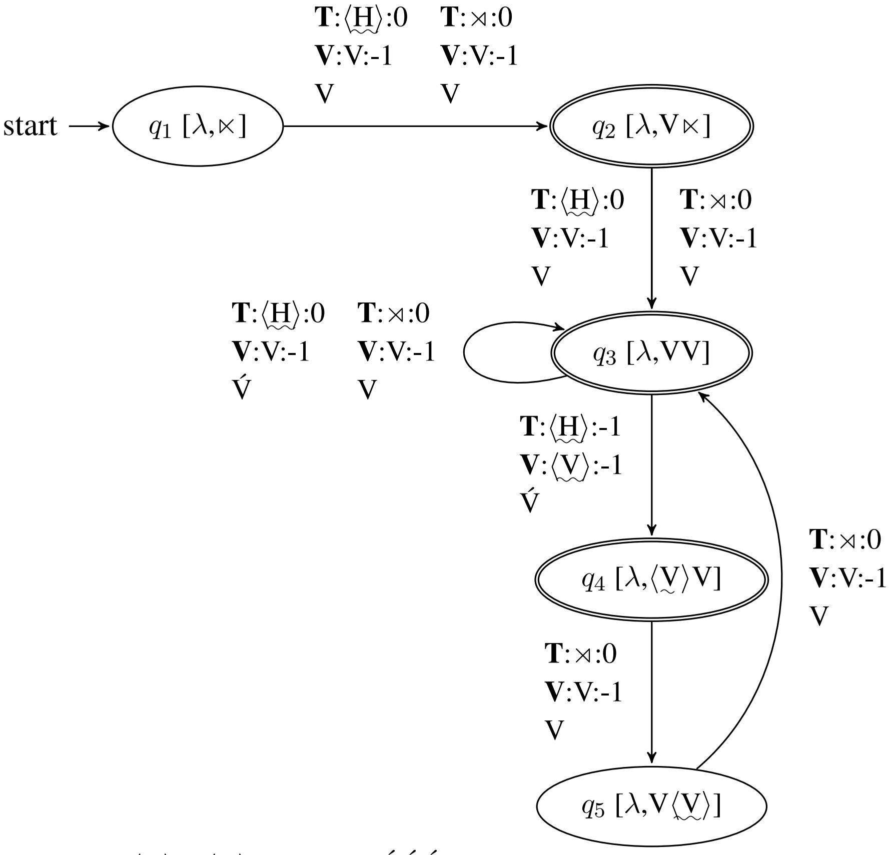
A.2.6 Ndebele unbouned spreading

In Ndebele (Figure 19), a preassociated H tone spreads up until the ante-penultimate vowel: /⟨H̰⟩ + ⟨V̰⟩VVVV/ maps to [V́V́V́VV]. This function is modeled by the [1,3]-MISL MT-FST in Figure

41a, with a sample derivation in Figure 41b. We assume that the input tone string contains either an empty string $\rtimes\ltimes$ or a single preassociated H tone $\rtimes\langle\underline{\mathrm{H}}\rangle\ltimes$. The MT-FSTs processes the strings right-to-left. The final and penultimate strings are faithfully outputted; the ante-penultimate and any preceding vowels (up until the preassociated vowel) are outputted as high V́ if there exists a preassociated H tone.

Figure 41: *Unbounded spreading in Ndebele*

(a) *MT-FST for Ndebele*



(b) *Derivation of* /$\langle\utilde{\text{H}}\rangle$ + $\langle\utilde{\text{V}}\rangle$VVVV/→[V́V́V́VV] *in Ndebele with the MT-FST in* Figure 41a

	Current State	**T**-tape Arc	**T**-tape Next	**V**-tape Arc	**V**-tape Next	Output symbol	Output String
1.	q_0		$\rtimes\langle\utilde{\text{H}}\rangle\underline{\ltimes}$		$\rtimes\langle\utilde{\text{V}}\rangle\text{VVVV}\underline{\ltimes}$		
2.	q_1	$\ltimes$:-1	$\rtimes\underline{\langle\utilde{\text{H}}\rangle}\ltimes$	$\ltimes$:-1	$\rtimes\langle\utilde{\text{V}}\rangle\text{VVV}\underline{\text{V}}\ltimes$	λ	
3.	q_2	$\langle\utilde{\text{H}}\rangle$:0	$\rtimes\underline{\langle\utilde{\text{H}}\rangle}\ltimes$	V:-1	$\rtimes\langle\utilde{\text{V}}\rangle\text{VV}\underline{\text{V}}\text{V}\ltimes$	V	V
4.	q_3	$\langle\utilde{\text{H}}\rangle$:0	$\rtimes\underline{\langle\utilde{\text{H}}\rangle}\ltimes$	V:-1	$\rtimes\langle\utilde{\text{V}}\rangle\text{V}\underline{\text{V}}\text{VV}\ltimes$	V	VV
5.	q_3	$\langle\utilde{\text{H}}\rangle$:0	$\rtimes\underline{\langle\utilde{\text{H}}\rangle}\ltimes$	V:-1	$\rtimes\langle\utilde{\text{V}}\rangle\underline{\text{V}}\text{VVV}\ltimes$	V́	V́VV
6.	q_3	$\langle\utilde{\text{H}}\rangle$:0	$\rtimes\underline{\langle\utilde{\text{H}}\rangle}\ltimes$	V:-1	$\rtimes\underline{\langle\utilde{\text{V}}\rangle}\text{VVVV}\ltimes$	V́	V́V́VV
7.	q_4	$\langle\utilde{\text{H}}\rangle$:-1	$\underline{\rtimes}\langle\utilde{\text{H}}\rangle\ltimes$	$\langle\utilde{\text{V}}\rangle$:-1	$\underline{\rtimes}\langle\utilde{\text{V}}\rangle\text{VVVV}\ltimes$	V́	V́V́V́VV
8.	q_f	$\rtimes$:-1	$\rtimes\langle\utilde{\text{H}}\rangle\ltimes$	$\rtimes$:-1	$\rtimes\langle\utilde{\text{V}}\rangle\text{VVVV}\ltimes$	λ	V́V́V́VV

A.3 Appendix of transducers for Semitic RPM

Below are MT-FSTs and derivation tables for some of the described Semitic processes from §7.2.

A.3.1 1-to-1 slot-filling with four consonants

In Figure 29, the input root **C** has 4 consonants *trʒm* and the template **P** has enough consonantal slots *CVCCVC*. The vocalism **V** is *ui*. The output is *turʒim*. A derivation table is provided in Table 42 using the [1,1,1]-MISL MT-FST from Figure 26.

Figure 42: *Derivation of* turʒim *using the MT-FST in* Figure 26

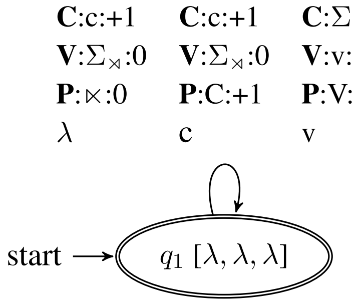
	Current	**C**-tape	**C**-tape	**V**-tape	**V**-tape	**P**-tape	**P**-tape	Output	Output
	State	Arc	Next	Arc	Next	Arc	Next	Symbol	String
1.	q_0		⋊̲trʒm⋉		⋊̲ui⋉		⋊̲CVCCVC⋉		
2.	q_1	⋊:+1	⋊t̲rʒɱ⋉	⋊:+1	⋊u̲i⋉	⋊:+1	⋊C̲VCCVC⋉	λ	
3.	q_1	t:+1	⋊tr̲ʒɱ⋉	u:0	⋊u̲i⋉	C:+1	⋊CV̲CCVC⋉	t	t
4.	q_1	r:0	⋊tr̲ʒɱ⋉	u:+1	⋊ui̲⋉	V:+1	⋊CVC̲CVC⋉	u	tu
5.	q_1	r:+1	⋊trʒ̲ɱ⋉	i:0	⋊ui̲⋉	C:+1	⋊CVCC̲VC⋉	r	tur
6.	q_1	ʒ:+1	⋊trʒm̲⋉	i:0	⋊ui̲⋉	C:+1	⋊CVCCV̲C⋉	ʒ	turʒ
7.	q_1	m:0	⋊trʒm̲⋉	i:+1	⋊ui⋉̲	V:+1	⋊CVCCVC̲⋉	i	turʒi
8.	q_1	m:+1	⋊trʒm⋉̲	⋉:0	⋊ui⋉̲	C:+1	⋊CVCCVC⋉̲	m	turʒim
9.	q_1	⋉:+1	⋊trʒm⋉	⋉:+1	⋊ui⋉	⋉:+1	⋊CVCCVC⋉	λ	turʒim

A.3.2 1-to-1 slot-filling with larger roots

In Figure 30, the root **C**=*mɣnṭs* contains more consonants than the template **P**=*CVCCVC*. With a vocalism **V**=*ui*, the output is *muɣniṭ* with final consonant deletion. This function is modeled by the [1,1,1]-MISL MT-FST in Figure 43a, illustrated with the derivation in Figure 43b.

Figure 43: *1-to-1 slot-filling in 5-consonant roots*

(a) *MT-FST for 1-to-1 association in 5-consonant roots*


C:c:+1 **C**:c:+1 **C**:Σ⋊:0
V:Σ⋊:0 **V**:Σ⋊:0 **V**:v:+1
P:⋉:0 **P**:C:+1 **P**:V:+1
λ c v
start
q_1 [λ, λ, λ]

(b) *Derivation of* muɣniṭ *using the MT-FST in* Figure 43a

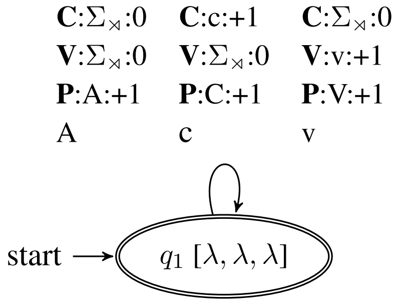
	Current State	**C**-tape Arc	**C**-tape Next	**V**-tape Arc	**V**-tape Next	**P**-tape Arc	**P**-tape Next	Output Symbol	Output String
1.	q_0		$\underline{\rtimes}$mɣnṭs⋉		$\underline{\rtimes}$ui⋉		$\underline{\rtimes}$CVCCVC⋉		
2.	q_1	⋊:+1	⋊$\underline{\text{m}}$ɣnṭs⋉	⋊:+1	⋊$\underline{\text{u}}$i⋉	⋊:+1	⋊$\underline{\text{C}}$VCCVC⋉	λ	
3.	q_1	m:+1	⋊m$\underline{\text{ɣ}}$nṭs⋉	u:0	⋊$\underline{\text{u}}$i⋉	C:+1	⋊C$\underline{\text{V}}$CCVC⋉	m	m
4.	q_1	ɣ:0	⋊m$\underline{\text{ɣ}}$nṭs⋉	u:+1	⋊u$\underline{\text{i}}$⋉	V:+1	⋊CV$\underline{\text{C}}$CVC⋉	u	mu
5.	q_1	ɣ:+1	⋊mɣ$\underline{\text{n}}$ṭs⋉	i:0	⋊u$\underline{\text{i}}$⋉	C:+1	⋊CVC$\underline{\text{C}}$VC⋉	ɣ	muɣ
6.	q_1	n:+1	⋊mɣn$\underline{\text{ṭ}}$s⋉	i:0	⋊u$\underline{\text{i}}$⋉	C:+1	⋊CVCC$\underline{\text{V}}$C⋉	n	muɣn
7.	q_1	ṭ:0	⋊mɣn$\underline{\text{ṭ}}$s⋉	i:+1	⋊ui$\underline{\ltimes}$	V:+1	⋊CVCCV$\underline{\text{C}}$⋉	i	muɣni
8.	q_1	ṭ:+1	⋊mɣnṭ$\underline{\text{s}}$⋉	⋉:0	⋊ui$\underline{\ltimes}$	C:+1	⋊CVCCVC$\underline{\ltimes}$	t	muɣnit
9.	q_1	s:+1	⋊mɣnṭs$\underline{\ltimes}$	⋉:0	⋊ui$\underline{\ltimes}$	⋉:0	⋊CVCCVC$\underline{\ltimes}$	λ	muɣnit
10.	q_1	⋉:+1	⋊mɣnṭs⋉	⋉:+1	⋊ui⋉	⋉:+1	⋊CVCCVC⋉	λ	muɣniṭ

A.3.3 1-to-1 slot-filling and pre-associated affixes

In Figure 31, the template **P**=*CtVCVC* has a preassociated affix ⟨t̲⟩. With a root **C**=*ksb* and vocalism **V**=*ui*, the output is *ktusib*. A [1,1,1]-MISL MT-FST is provided in Figure 44a along with a sample derivation in Figure 44b. The symbol *A* represents any input symbol from the input alphabet of segments {t,n,m} which are possible segmental affixes in McCarthy (1981).

Figure 44: *1-to-1 slot-filling with pre-associated affixes*

(a) *MT-FST for 1-to-1 slot-filling with pre-associated affixes*


C:$\Sigma_{\rtimes}$:0
V:$\Sigma_{\rtimes}$:0
P:A:+1
A
C:c:+1
V:$\Sigma_{\rtimes}$:0
P:C:+1
c
C:$\Sigma_{\rtimes}$:0
V:v:+1
P:V:+1
v
start
q_1 [$\lambda, \lambda, \lambda$]

(b) *Derivation of* k⟨t⟩usib

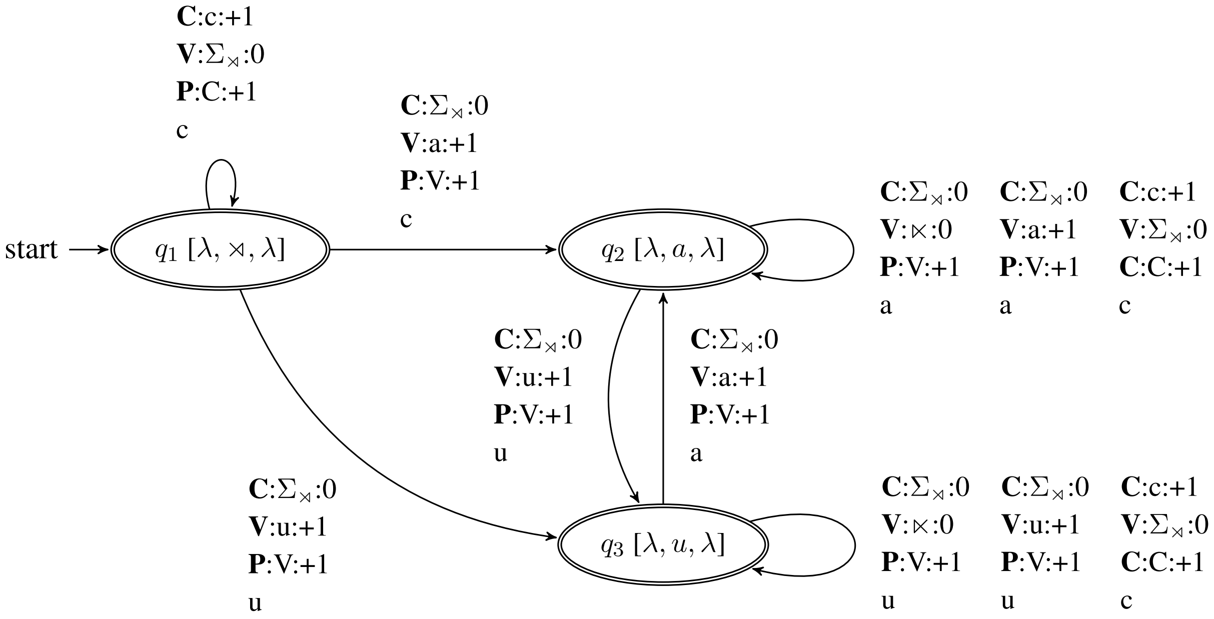
	Current State	**C**-tape Arc	**C**-tape Next	**V**-tape Arc	**V**-tape Next	**P**-tape Arc	**P**-tape Next	Output Symbol	Output String
1.	q_0		$\underline{\rtimes}\text{ksb}\ltimes$		$\underline{\rtimes}\text{ui}\ltimes$		$\underline{\rtimes}\text{CtVCVC}\ltimes$		
2.	q_1	$\rtimes$:+1	$\rtimes\underline{\text{k}}\text{sb}\ltimes$	$\rtimes$:+1	$\rtimes\underline{\text{u}}\text{i}\ltimes$	$\rtimes$:+1	$\rtimes\underline{\text{C}}\text{tVCVC}\ltimes$	λ	
3.	q_1	k:+1	$\rtimes\text{k}\underline{\text{s}}\text{b}\ltimes$	u:0	$\rtimes\underline{\text{u}}\text{i}\ltimes$	C:+1	$\rtimes\text{C}\underline{\text{t}}\text{VCVC}\ltimes$	k	k
4.	q_1	s:0	$\rtimes\text{k}\underline{\text{s}}\text{b}\ltimes$	u:0	$\rtimes\underline{\text{u}}\text{i}\ltimes$	t:+1	$\rtimes\text{Ct}\underline{\text{V}}\text{CVC}\ltimes$	t	kt
5.	q_1	s:0	$\rtimes\text{k}\underline{\text{s}}\text{b}\ltimes$	u:+1	$\rtimes\text{u}\underline{\text{i}}\ltimes$	V:+1	$\rtimes\text{CtV}\underline{\text{C}}\text{VC}\ltimes$	u	ktu
6.	q_1	s:+1	$\rtimes\text{ks}\underline{\text{b}}\ltimes$	i:0	$\rtimes\text{u}\underline{\text{i}}\ltimes$	C:+1	$\rtimes\text{CtVC}\underline{\text{V}}\text{C}\ltimes$	s	ktus
7.	q_1	b:0	$\rtimes\text{ks}\underline{\text{b}}\ltimes$	i:+1	$\rtimes\text{ui}\underline{\ltimes}$	V:+1	$\rtimes\text{CtVCV}\underline{\text{C}}\ltimes$	i	ktusi
8.	q_1	b:+1	$\rtimes\text{ksb}\underline{\ltimes}$	$\ltimes$:0	$\rtimes\text{ui}\underline{\ltimes}$	C:+1	$\rtimes\text{CtVCVC}\underline{\ltimes}$	b	ktusib
9.	q_1	$\ltimes$:+1	$\rtimes\text{ksb}\ltimes$	$\ltimes$:+1	$\rtimes\text{ui}\ltimes$	$\ltimes$:+1	$\rtimes\text{CtVCVC}\ltimes$	λ	ktusib

A.3.4 1-to-many slot-filling with final spread of vowels

In Figure 32a, the vocalism **V**=*a* has fewer vowels than the template **P**=*CVCVC*. This triggers final spread of vowels. With a root **C**=*ktb*, the output is *katab*. This function is modeled with the [1,2,1]-MISL MT-FST in Figure 45a, illustrated with a sample derivation in Figure 45b. The vowel alphabet is only {a,u}. In Standard Arabic, only the vowels *a,u* spread; the vowel *i* does not. This is discussed in Dolatian and Rawski (2020b).

Figure 45: *1-to-1 slot-filling with final spread of vowels*

(a) *MT-FST for 1-to-many slot-filling with final spread of vowels*



(b) *Derivation of* katab *using the MT-FST in* Figure 45a

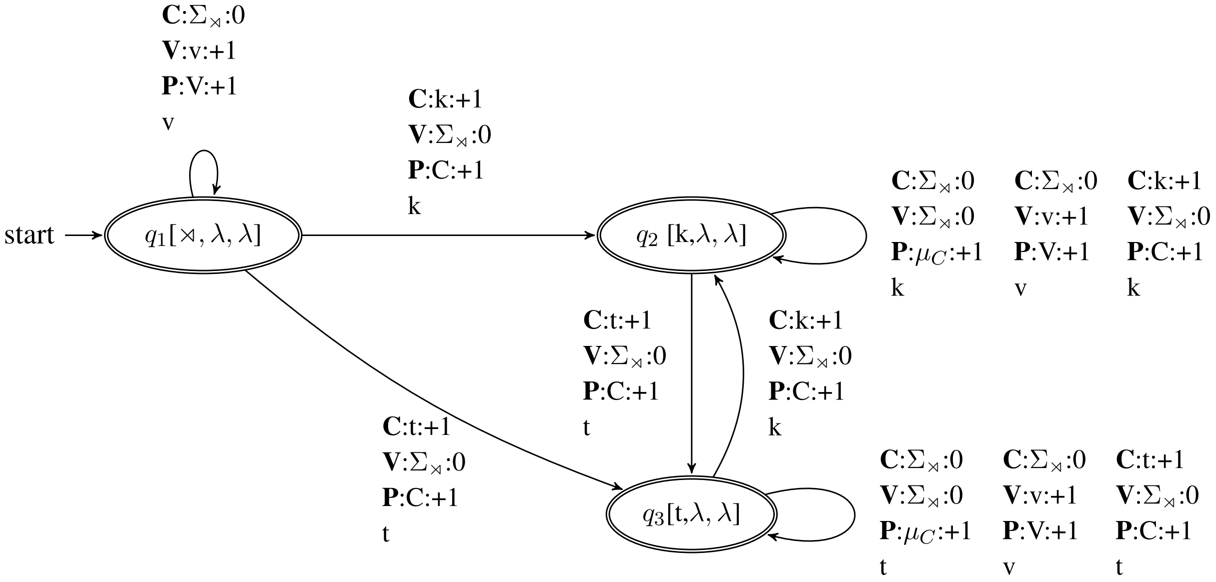
	Current State	**C**-tape Arc	**C**-tape Next	**V**-tape Arc	**V**-tape Next	**P**-tape Arc	**P**-tape Next	Output Symbol	Output String
1.	q_0		$\underline{\rtimes}$ktb$\ltimes$		$\underline{\rtimes}$a$\ltimes$		$\underline{\rtimes}$CVCVC$\ltimes$		
2.	q_1	$\rtimes$:+1	$\rtimes\underline{\text{k}}tb\ltimes$	$\rtimes$:+1	$\rtimes\underline{\text{a}}\ltimes$	$\rtimes$:+1	$\rtimes\underline{\text{C}}$VCVC$\ltimes$	λ	
3.	q_1	k:+1	$\rtimes$k$\underline{\text{t}}$b$\ltimes$	a:0	$\rtimes\underline{\text{a}}\ltimes$	C:+1	$\rtimes$C$\underline{\text{V}}$CVC$\ltimes$	k	k
4.	q_2	t:0	$\rtimes$k$\underline{\text{t}}$b$\ltimes$	a:+1	$\rtimes$a$\underline{\ltimes}$	V:+1	$\rtimes$CV$\underline{\text{C}}$VC$\ltimes$	a	ka
5.	q_2	t:+1	$\rtimes$kt$\underline{\text{b}}\ltimes$	$\ltimes$:0	$\rtimes$a$\underline{\ltimes}$	t:+1	$\rtimes$CVC$\underline{\text{V}}$C$\ltimes$	t	kat
6.	q_2	b:0	$\rtimes$kt$\underline{\text{b}}\ltimes$	$\ltimes$:0	$\rtimes$a$\underline{\ltimes}$	V:+1	$\rtimes$CVCV$\underline{\text{C}}\ltimes$	a	kata
7.	q_2	b:+1	$\rtimes$ktb$\underline{\ltimes}$	$\ltimes$:0	$\rtimes$a$\underline{\ltimes}$	C:+1	$\rtimes$CVCVC$\underline{\ltimes}$	b	katab
8.	q_f	$\ltimes$:+1	$\rtimes$ktb$\ltimes$	$\ltimes$:+1	$\rtimes$a$\ltimes$	$\ltimes$:+1	$\rtimes$CVCVC$\ltimes$	λ	katab

A.3.5 1-to-many slot filling with medial spread of consonants

In Figure 34b, the template **P**=*CVCμ_CVC* contains a marker for gemination. With root **C**=*ktb* and vocalism **V**=*ui*, the output is *kuttib*. This is modeled by the [2,1,1]-MISL MT-FST in Figure 46a, with a sample derivation in Figure 46b for a nonce word *kuttik* with root **C**=*ktk*. For illustrative reasons, the consonant alphabet is only {k,t}.

Figure 46: *1-to-1 slot-filling with final spread of vowels*

(a) *MT-FST for 1-to-many slot-filling with medial spread of consonants*


C:$\Sigma_{\rtimes}$:0
V:v:+1
P:V:+1
v
C:k:+1
V:$\Sigma_{\rtimes}$:0
P:C:+1
k
start
$q_1[\rtimes, \lambda, \lambda]$
q_2 [k,λ, λ]
C:$\Sigma_{\rtimes}$:0
V:$\Sigma_{\rtimes}$:0
P:μ_C:+1
k
C:$\Sigma_{\rtimes}$:0
V:v:+1
P:V:+1
v
C:k:+1
V:$\Sigma_{\rtimes}$:0
P:C:+1
k
C:t:+1
V:$\Sigma_{\rtimes}$:0
P:C:+1
t
C:k:+1
V:$\Sigma_{\rtimes}$:0
P:C:+1
k
C:t:+1
V:$\Sigma_{\rtimes}$:0
P:C:+1
t
q_3[t,λ, λ]
C:$\Sigma_{\rtimes}$:0
V:$\Sigma_{\rtimes}$:0
P:μ_C:+1
t
C:$\Sigma_{\rtimes}$:0
V:v:+1
P:V:+1
v
C:t:+1
V:$\Sigma_{\rtimes}$:0
P:C:+1
t

(b) *Derivation of* kuttik *using the MT-FST in* Figure 46a

	Current State	**C**-tape Arc	**C**-tape Next	**V**-tape Arc	**V**-tape Next	**P**-tape Arc	**P**-tape Next	Output Symbol	Output String
1.	q_0		$\underline{\rtimes}$ktk$\ltimes$		$\underline{\rtimes}$ui$\ltimes$		$\underline{\rtimes}$CVCμ_CVC$\ltimes$		
2.	q_1	$\rtimes$:+1	$\rtimes$$\underline{\text{k}}tk\ltimes$	$\rtimes$:+1	$\rtimes$$\underline{\text{u}}i\ltimes$	$\rtimes$:+1	$\rtimes$$\underline{\text{C}}VC\mu_CVC\ltimes$	λ	
3.	q_2	k:+1	$\rtimes$k$\underline{\text{t}}$k$\ltimes$	u:0	$\rtimes$$\underline{\text{u}}i\ltimes$	C:+1	$\rtimes$C$\underline{\text{V}}$Cμ_CVC$\ltimes$	k	k
4.	q_2	k:0	$\rtimes$k$\underline{\text{t}}$k$\ltimes$	u:+1	$\rtimes$u$\underline{\text{i}}$$\ltimes$	V:+1	$\rtimes$CV$\underline{\text{C}}$$\mu_CVC\ltimes$	u	ku
5.	q_3	t:+1	$\rtimes$kt$\underline{\text{k}}$$\ltimes$	i:0	$\rtimes$u$\underline{\text{i}}$$\ltimes$	C:+1	$\rtimes$CVC$\underline{\mu_C}$VC$\ltimes$	t	kut
6.	q_3	k:0	$\rtimes$kt$\underline{\text{k}}$$\ltimes$	i:0	$\rtimes$u$\underline{\text{i}}$$\ltimes$	μ_C:+1	$\rtimes$CVC$\mu_C$$\underline{\text{V}}C\ltimes$	t	kutt
7.	q_3	k:0	$\rtimes$kt$\underline{\text{k}}$$\ltimes$	i:+1	$\rtimes$ui$\underline{\ltimes}$	V:+1	$\rtimes$CVCμ_CV$\underline{\text{C}}$$\ltimes$	i	kutti
8.	q_3	k:+1	$\rtimes$ktk$\underline{\ltimes}$	$\ltimes$:0	$\rtimes$ui$\underline{\ltimes}$	C:+1	$\rtimes$CVCμ_CVC$\underline{\ltimes}$	k	kuttik
9.	q_f	$\ltimes$:+1	$\rtimes$ktk$\ltimes$	$\ltimes$:+1	$\rtimes$ui$\ltimes$	$\ltimes$:+1	$\rtimes$CVCμ_CVC$\ltimes$	λ	kuttik