Computational locality in nonlinear morphophonology

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

November 26, 2020

Contents

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 languages 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. They depend only on properties that are necessarily common to all computational models, grammars, or machines that can distinguish the pattern from others. Cognitively, these are properties which any cognitive mechanism needs to be sensitive to in order to correctly classify and process the phenomenon. The transparent, analytical guarantees given by formal languages define explicit classes of phonological grammars, whether for the purpose of getting the right generalizations or for learning (Heinz 2018).

The resurgence in formal languages has been accompanied by a methodological angst regarding their use. Segmental phonology can be easily modeled with string-based representations, and this has been adequate to show that segmental phonology has a very limited generative capacity (Chandlee 2014; Chandlee and Heinz 2018; Chandlee et al. 2018). Specifically, most segmental phonology is computationally local, formally defined as being computed by **Input Strictly Local** (ISL) functions.

In contrast, suprasegmental processes like tone are computationally more expressive than segmental phonology (Jardine 2016a), requiring the expressivity of the **Regular** functions, a strict superset. This dichotomy has provoked many responses. On the one end, some segmental processes

like vowel harmony display the same high level of expressivity as tone (McCollum et al. 2020). On the other end, there are arguments over whether formal language analyses are relevant at all to phonology and over the role of constraint-interaction grammars such as Optimality Theory (Pater 2018, 2019; Jardine 2019a).[1]

A representational assumption underlying much of this computational work treats the inputs and outputs of these functions as a single string of segments with tonal marking. Strings are a simple and efficient 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 multiple autosegmental tiers (Williams 1976; Goldsmith 1976). This representation was first proposed for tone, and was then extended to non-concatenative morphology (McCarthy 1979, 1981a).

For tone, recent work in mathematical phonology has tried to incorporate this autosegmental insight by changing the data structure or representation of the input from strings to graph structures. 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 tonal representation and computation: a more elaborate representation yields a (formally) simpler computation.[2]

The present paper provides a different perspective on nonlinear computation based on early computational work in Semitic root-and-pattern morphology (Kay 1987; Kiraz 2001).[3] Instead of using autosegmental graphs, we combine string-based and tier-based representations in order to exploit the simple computation of strings while still benefiting from the relativized locality of autosegmental tiers. Our insight is to treat nonlinear processes as n-ary functions, or functions which can take multiple strings as input and/or output. This is in contrast to the assumption of string-to-string or graph-to-graph transductions, which are unary functions that take only one input representation and produce one output. Whereas string-to-string functions are implemented with simple single-tape finite-state transducers (FSTs), n-ary functions are implemented with multi-tape FSTs. This new perspective balances the focus between representation and computation in nonlinear processes. In this sense, autosegmental structure is the residue or trace of the computation. This derivational perspective is similar to the notion of tree structure as the trace of syntactic computation (Stabler

[1] Constraint-interaction grammars are computationally quite nebulous. They are known to be Turing complete in general (Smolensky 1993; Hale and Smolensky 2006), can both over- and under-generate by creating bizarre and complex processes (Lamont 2019a,b, prep; Hao 2019), are prone to hard-to-discover errors (Karttunen 2006a,b), have issues with computational tractability (Idsardi 2006; Heinz et al. 2009), and have found 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. This is the case for debates on the use of Q-theory (Shih and Inkelas 2018) vs. autosegmental structures (Jardine et al. 2020), the feature geometry of tones (Oakden 2020), and different syllable representations (Strother-Garcia 2019).

[3] This is a synthesis of our previous work (Dolatian and Rawski 2020a,b; Rawski and Dolatian 2020).

1996; Hunter 2019).

Our central technical 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 how the MISL automata elegantly characterize a majority of nonlinear suprasegmental phonology. Additionally, local segmental phonology is ISL, and ISL is just a special case of MISL functions. We demonstrate our results both for tonal phonology and for template-filling processes in root-and-pattern morphology. This characterization shows the salience of strict locality even 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. With a solid formal grounding, we contribute to discussionson the role of directionality, tier conflation, prosodic templates, and the tension between finiteness and infinity in non-concatenative morphology.

We lay out our argument in the following way. Section 2 discusses the computational notion of locality over phonological representations. Section 3 relativizes such locality over multiple inputs using muli-tape automata. Section 4 analyzes the typology of tonal processes in phonology, showing that the bulk of them are computationally local. Section 5 discusses the nuances that this approach brings, with issues concerning directionality, contour tones and so on. Sections 6 & 7 extend our approach to consider non-concatenative morphological processes, using Semitic template filling as a case study, which again turn out to be 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 in phonological processes (Odden 1994). There is a substantial body of work which shows that segmental processes tend to be local processes. These intuitive notions of locality are formalizable with subclasses of finite-state transducers and their corresponding functions. There is comparatively little work on analyzing locality in autosegmental processes, i.e., tone and root-and-pattern morphology. In this section, we discuss locality and break it down into a set of formal parameters couched within Formal Language Theory.

2.1 Locality of segmental phonology over single-strings

Consider the cross-linguistically common segmental process of post-nasal voicing (Pater 1999). The target of the rule is an underlying stop. The trigger of the rule is the context of the stop. A nasal will trigger the voicing of the stop if the nasal is *immediately* before the stop.

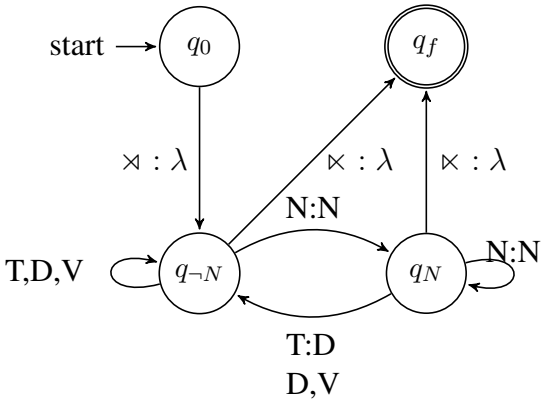
(1) a. T → D / N _

b.

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

Post-nasal voicing is controlled by strict adjacency. Computationally, the process of post-nasal voicing is a Regular relation, for which there are many different procedural implementations. Within computational linguistics, the most common implementation for computing such Regular phonological processes are finite-state transducers (FST) (Kaplan and Kay 1994; Roche and Schabes 1997). Figure 1 visually 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 can only read the start-boundary $\rtimes$, the latter can only be 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/

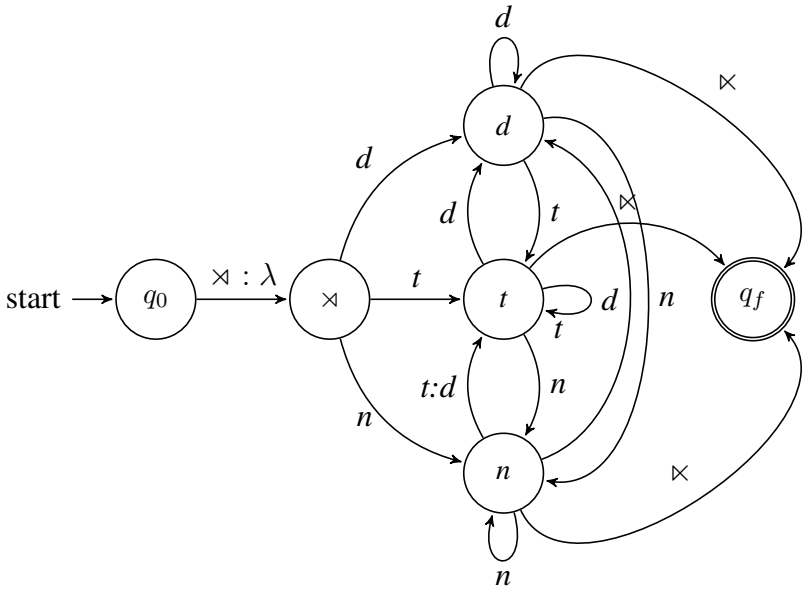
	Current state	Used Arc	Input tape	Output symbol	Output string
1.	q_0		$_r\underline{\rtimes}$anta$\ltimes$		
2.	$q_{\neg N}$	$\rtimes$:λ	$\rtimes_r\underline{a}$nta$\ltimes$	λ	
3.	$q_{\neg N}$	a:a	$\rtimes$a$_r\underline{n}$ta$\ltimes$	a	a
4.	q_N	n:n	$\rtimes$an$_r\underline{t}$a$\ltimes$	n	an
5.	$q_{\neg N}$	t:t	$\rtimes$ant$_r\underline{a}\ltimes$	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 like the one above compute regular relations, but we do not need the full power of FSTs in order to compute post-nasal voicing. In fact, 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). Informally, let x be a string made up of m elements: $x_1x_2...x_{j-1}x_jx_{j+1}...x_m$. Given this input, 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 what is 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 segments ($x_{j-k}...x_{j-1}$).[4] There is a different state for every possible substring of size $k-1$. These FSTs can have a substantially larger number of states than the minimal FST.

Tying this formal concept back to post-nasal voicing, post-nasal voicing is local, a 2-ISL process that only requires looking at a window of size *k*=2 over the input: the current input symbol and the previous input symbol. As a 2-ISL function, post-nasal voicing 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 canonical ISL FST in Figure 3 has 6 states: q_0, q_f are the initial and final state, $q_{\rtimes}$ reads the start boundary, and states q_t, q_d, q_n do the brunt of the work and read the input

[4]The intuitive explanation gets complicated when we look at what segments follow x_j. An FST for the mirror process of pre-nasal voicing is 2-ISL. A stop x_j is nasalized if x_{j+1} a nasal. An FST would need to read the entire substring x_jx_{j+1} before outputting anything (cf. Chandlee 2014:ch:3.1).

segments. Each state encodes the last $k-1$ input symbols ($= 2-1 = 1$ symbols). Intuitively, the states encode whether the previous segment was a nasal, voiceless stop, or voiced stop.[5] In contrast, the minimal FST from Figure 1 has fewer states. It merges states $q_{\rtimes}, q_t, q_d$ into a single state $q_{\neg N}$. This state encodes whether the previous input symbol was a non-nasal. However, even though the ISL FST has more states, this should not be taken as evidence that it is somehow more complex than the minimal FST. State size is not correlated with generative capacity.

2.2 Non-locality of tone over single strings

The previous section showed how a segmental process like post-nasal voicing is intuitively local, computable by FSTs, and 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 (IPA symbols or feature bundles) and this string is transformed into an output string. 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.[6]

The importance of input representation is clearer when we look at a tonal process like final-tone spreading. In Mende, words appear with a restricted set of possible tone sequences. Figure 4 shows these surface toned vowels. For each word, we also show a separate string of just the tones: *kɔ́ H*. We break up contour tones to their component tones in parentheses: *mbû* (HL). The basic 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 toneless vowels at the right-edge of the string.

Figure 4: *Distribution of tones in Mende* (Jardine 2016b:17, citing Leben 1973, 1978)

	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/	mbă	(LH)	‘rice’	nìká	LH	‘cow’	ndàvúlá	LHH	‘sling’
/LHL/	mbã	(LHL)	‘companion’	nyàhâ	L(HL)	‘woman’	nìkílì	LHL	‘groundnut’

Many tonal processes like Mende left-to-right spreading are intuitively local, but this intuition

[5] It is not the case that the canonical FST has one state per natural class. It has one state per possible substring of segments of size $k-1$.

[6] 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.

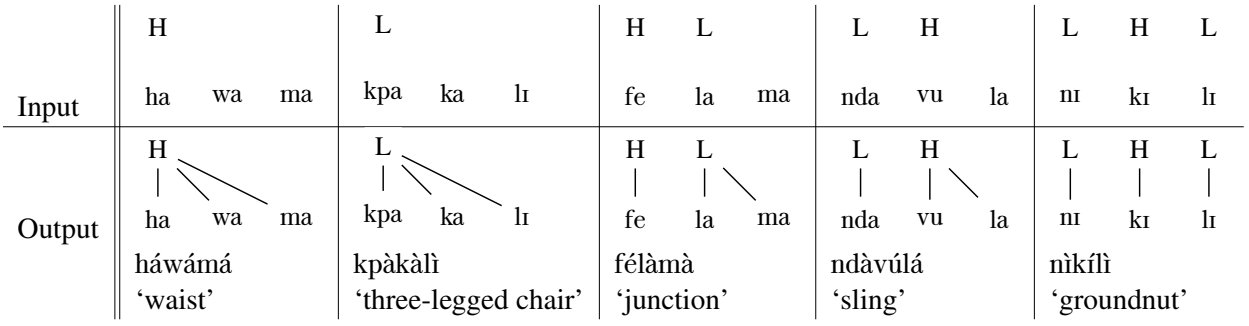
depends on how we represent the input. An early trend in generative phonology was to represent tones as part of a vowel, and thus part of a single input string: /háwama/ or /V́VV/. Over this single-string representation, this tonal process can be computed by a single-tape FST (cf. Gibbon 1987, 2001). 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, to form *h<u>á</u>wám***á** or [<u>V́</u>V́**V́**], the final vowel (**target**) receives its high tone feature the initial vowel (<u>trigger</u>).

The Mende example demonstrates a recurrent problem in the computation of tone.[7] Over single-string representations, the computation of certain tonal processes is non-local, even though the autosegmental notation implies that the computation *should* be local (Jardine 2016b,a, 2017a; Chandlee and Jardine 2019a). Because of this problem and other problems, early work in computational phonology tried to define refinements to finite-state systems in order to make tonal computation easier to model and design (Bird and Ellison 1994; Kornai 1995). The main strategy was to notationally incorporate autosegmental structure into a single-string representation (Yli-Jyrä 2013, 2015, 2019). The motivation for this early strategy was mostly practical. Single-tape FSTs are efficient and useful tools in computational phonology and NLP. By using single-string representations, it is likewise easier to make connections between finite-state treatments of tone with early work on logical-treatments of tone (Bird and Klein 1990; Bird 1995; Coleman and Local 1991; Coleman 1998). However, these notational strategies do not affect the generative capacity of tone. Over a single-string representation, Mende left-to-right spread is computationally non-local.

2.3 Multi-input representations for tone

Since the advent of autosegmental phonology (Williams 1976; Goldsmith 1976), the representation of tone has broken away from single-string representations. Instead, tonal information is 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

[7] In contrast, a recurrent exception to this problem is the tonal phonology of East Asian languages. Tone sandhi in these languages tends to be local even over segmental strings (Chandlee 2019; Oakden 2020).

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 for both the input and output. In fact, 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). An autosegmental grammar is a grammar of possible graphical relations. The benefit of using these autosegmental grammars is their transparent interpretation. Their transparent interpretation 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 simple linear 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 describe an alternative perspective which exploits 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), and then illustrate them for tone (§3.2). We then introduce locality to our grammars (§3.3).

3.1 Multi-tape finite-state transducers

Multi-tape FSTs are FSTs where the inputs are tuples 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 single tape or multiple tapes as well. We focus on the relevant case where the MT-FST has multiple input tapes, but a single output tape. Every input tape has its own unique read-head. The benefit of our MT-FST formalizations is that it connects the computation of tone with pre-existing results on the use of MT-FSTs, whether these results are based on their computational properties or based on their implementability. Autosegmental grammars are conceptually attractive. But because they are novel, they do not have the benefit of having

pre-existing computational results.[8]

We are not the first to use multi-tape FSTs for tone or computational phonology. In computational linguistics, the earliest use of MT-FSTs is Kay (1987) who developed a sketch of an MT-FST for Semitic root-and-pattern morphology. Larger-scale formalizations were later developed by Wiebe (1992) for tone and Kiraz (2000, 2001) for Semitic. Implementations exist for the computational morphology of Arabic (Kiraz 2001; Habash and Rambow 2006). Our work extends this line of work by focusing on the what sufficient information is needed to compute autosegmental processes. By developing the MISL subclass, we emphasize that autosegmental phonology is computational local.

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 formalizing multi-input functions and 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. In contrast, if the read-heads do not always move simultaneously, then 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 between synchronous MT-FSTs and single-tape MT-FSTs, synchronous MT-FSTs have been used to model intermediate representations in computational morphology and phonology (Hulden 2017a,b). The benefit is that synchronous MT-FSTs make it easier to debug large-scale finite-state systems. In contrast, asynchronous MT-FSTs are more expressive than single-tape FSTs. 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). We discuss synchronous MT-FSTs more in §7.1 for Semitic. Both have likewise been used for the computation of feature tiers in speech processing (Carson-Berndsen 1998; Shu 2006). In this paper, we use the more expressive class of asynchronous MT-FSTs. We are the first however to define a locally-based subclass for MT-FSTs.

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

[8] An open and interesting 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, one need only enrich the alphabet to include both preassociated and non-associated symbols, as endorelations are not restricted by set union.

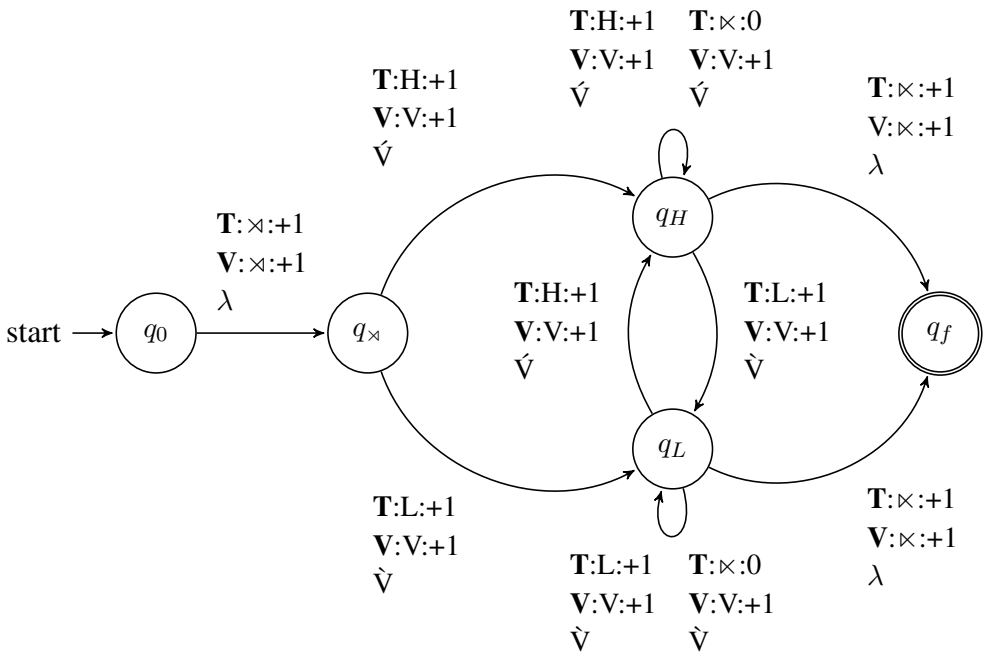
the **T**-string and **V**-string respectively. For illustration, we omit consonants from the **V**-string and we do not specify vowel quality. The output is a single string over a new alphabet of toned-vowels. The strings are flanked by the end boundaries ⋊,⋉.

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 separate tapes: **T**-tape and **V**-tape. The machine has 2 read-heads r_V, r_T. The read-head r_V moves over the string of vowels on the **V**-tape, and the read-head r_T moves over the string of tones or the **T**-tape. 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.[9]

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 other 3 states

[9]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 §5.2.

have mnemonic labels. Their labels match the state's 'role'. 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.[10] Each arc has three lines. For the arc from $q_{\rtimes}$ to q_H, the arc is:

T:H:+1
V:V:+1
V́

This transition arc will output a high vowel V́, if the machine read a H tone on the **T**-tape and vowel V on the **V**-tape. 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 substring for this transition arc as the toned vowel 'V́'. The transition arcs use a lower-case variable V to denote any vowel. 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

Figure 8 shows a simplified 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 easier readability, we do not show the two read-heads r_T and r_V, but they are situated before the underlined symbol, e.g., $\rtimes_{r_T}$HL$\ltimes$and $\rtimes_{r_V}$eaa$\ltimes$.

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

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

The MT-FST above is *asynchronous* because the two read-heads do not simultaneously advance at every step or transition. At point 5 for *felama*, the **T**-tape and **V**-tape have the configurations

[10] We notationally diverge from Dolatian and Rawski (2020b). There, the transition arcs are represented in terms of vectors and tuples. We use a simpler notation in this paper for illustration.

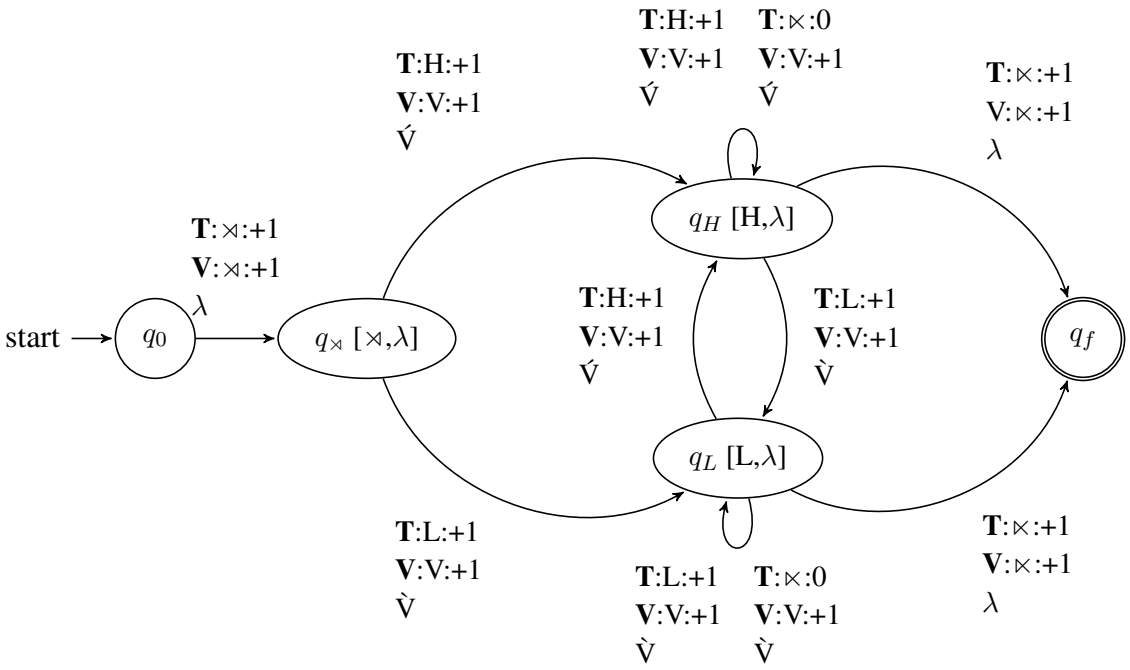
⋊HL⋉ and ⋊eaa⋉. The MT-FST has thus already read the final tone but not the final vowel. At state q_L, the interpretation is that the read-head r_T on the **T**-tape has recently read an L tone, and the next symbol on the **T**-tape is the end-boundary ⋉. In contrast on the **V**-tape, the read-tape r_V has read some vowels and will read a new vowel V (=*a*). With this configuration, the MT-FST will not advance on the **T**-tape (= **T**:⋉:0), but advance on the **V**-tape (= **V**:*a*:+1) and produce a low vowel *à*. This is how the machine generates final spreading.

3.3 Locality over multi-input representations

The previous section showed that MT-FSTs are a procedural and computational description which captures the fact that tonal phonology operates over two separate tiers, formalized as separate strings or tapes. The finite-state nature of MT-FSTs restricts the space of possible tonal processes. We now show that as a multi-input function, an intuitively simple tonal process like 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 combination of these two windows form a vector $\vec{k} = [k_T, k_V]$.

For the state q_L, the 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 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 machine captures the fact that left-to-right spreading is a computationally local process over the multiple input representations: the tone tier and the vowel tier.

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 multiple-tape FSTs. Second, we used our finite-state characterization to establish a restrictive subclass of MT-FSTs that are computationally local. Their locality is based on how the states of the MT-FST 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 briefly go through a typology of tonal phenomena. We show that all these processes are computable by MT-FSTs, and that the majority are computable by local MT-FSTs. There are many typology-based or theory-based classifications of tonal phonology (Yip 2002; Hyman 2011). But, there are little to no *computationally*- or *mathematically*-based classifications for tone. To our knowledge, the two most significant surveys are Koser et al. (2019) and Chandlee and Jardine (2019a), alongside other work by Jardine (2016a,b, 2017a,b, 2019b). Because of this limitation, we evaluated MT-FSTs against the two surveys: 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 some tones are underlyingly preassociated (§4.3). For readability, we place any example MT-FSTs and derivations in the appendix.

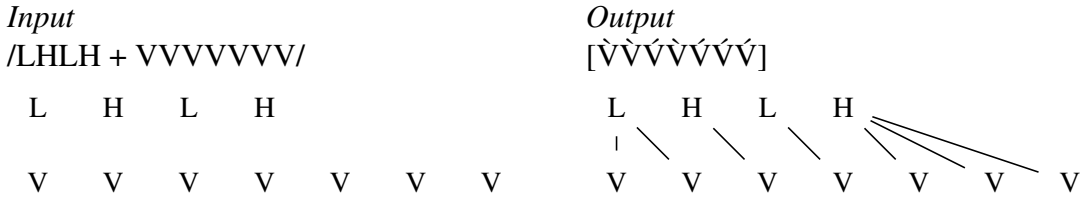
4.1 Locality of tone without preassociation

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 *not* preassociated, i.e., 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. To clarify, the input /LHLH + VVVVVVV/ is made up of two strings: /LHLH/ and /VVVVVVV/. The first tone associates with the first two vowels. The remaining tones and vowels are associated 1-to-1. If there are more

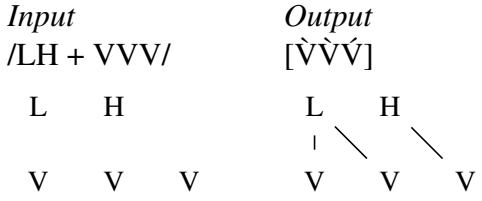
vowels than tones, the final tone is spread: [V̀V̀V́V̀V́V́V́]. Initial spreading up to two vowels is [2,3]-MISL. The function requires the context [⋊L,⋊VV] in order to spread L to the first two vowels. Final spread is [2,1]-MISL as in Mende (§3.3). Together, Kikuya is [2,3]-MISL. A sample MT-FST and derivation are in the appendix (Figure 39).

Figure 10: *Limited spreading in Kikuyu* (Koser et al. 2019 citing Clements and Ford 1979)



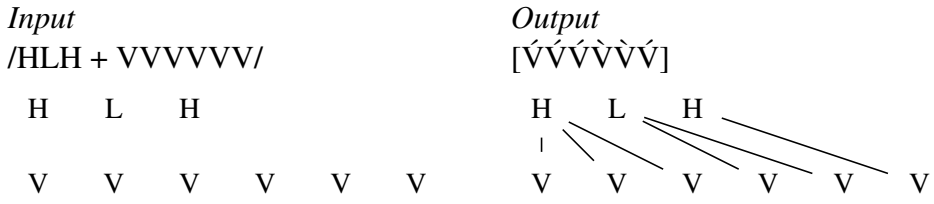
Hausa behaves analogously to Mende but tones are associated *right-to-left* with *initial*-spreading: /LH + VVV/ → [V̀V̀V́]. This is [2,1]-MISL *when* the input string is read right-to-left. A sample MT-FST and derivation are in the appendix (Figure 40). We further discuss the role of directionality in §5.1.

Figure 11: *Right-to-left spreading in Hausa* (Koser et al. 2019 citing Newman 1986, 2000)



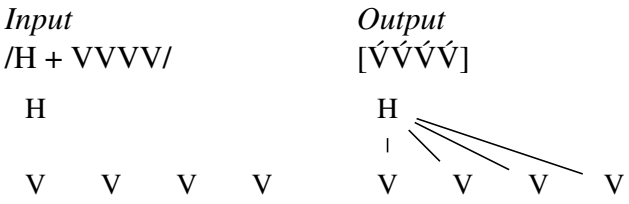
North Karanga Shona is more complex. The initial and final tones are associated to the first and last vowels respectively. The first tone can spread up until the first 3 vowels *but not* to the penultimate vowel. The medial tone can spread up until the penultimate vowel: /HLH + VVVVVV/ → [V́V́V́V̀V̀V́].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

Figure 12: *Edge-in spread in Northern Karanga Shona* (Koser et al. 2019 citing Odden 1981; Hewitt and Prince 1989)



Lastly, Kukuya allows a H tone to spread if it is the only tone: /H + VVV/ → [V́V́V́]. Otherwise, if the input is HL, the L tone spreads: /HL + VVV/ → [V́V́V̀]. If LH, the L spreads up until the penultimate vowel: /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.[11] We do not draw an MISL MT-FST because of its size.

Figure 13: *Quality-sensitive spreading in Kukuya* (Koser et al. 2019 citing Hyman 1987)



Thus just like Mende, the above tonal processes are MISL. Depending the specific pattern, the locality window can be quite large but still MISL.

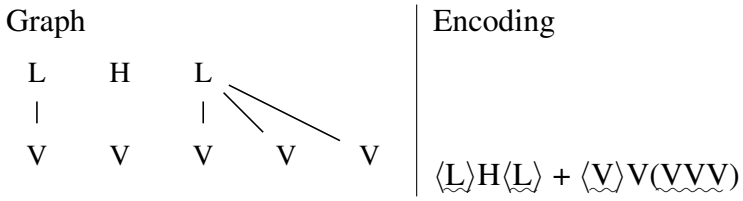
4.2 Linear encoding of tonal preassociation

In contrast to the previous section, tonal processes may include inputs where a tone is underlying *preassociated* to one or more vowels. This dependency between the two strings is a reason why graphical structures are useful representations for tone. But, this is also the reason why many linear encodings of tone require some special markup system (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, 2015). We do not use other proposed encoding systems (Wiebe 1992; Kornai 1995; Yli-Jyrä 2013, 2015, 2019) because they either are designed for single-tape FSTs or do not maintain strict locality.

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

Figure 14: Encoding preassociation



If a tone T or single vowel V is preassociated, it is has wavy underlining and is demarcated with angle brackets: ⟨T⟩, ⟨V⟩. If a span of multiple vowels are associated to the same tone, they are marked with parentheses instead of angle brackets: (V V V ... V V). Note that is sequence of preassociated vowels uses only three unique multi-character symbols which crucially can include a parenthesis boundary: '(V' and 'V' and 'V)'. This encoding creates the following enriched input alphabets of multi-character units:

- Σ_T ={ H, L, ⟨H⟩, ⟨L⟩}
- Σ_V ={ V, ⟨V⟩, (V, V, V) }

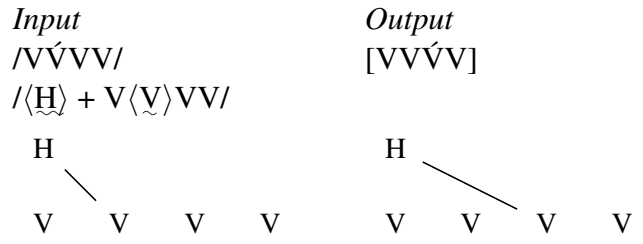
Other possible configurations, such as word-medial contour tones require a more elaborate encoding which we do not discuss here. We set these aside until §5.2 because the preassociation data in Chandlee and Jardine (2019a) did not have such case studies. We set aside the evaluation of our encoding mechanism based on Kornai (1995)'s desirada. However, because the alphabet uses multiple preassocation symbols for vowels, the alphabet can MT-FSTs with a large number of states and transition arcs. This has prevented us from drawing illustrative MT-FSTs for some of the following case studies; but that doesn't contradict the locality of the patterns.

4.3 Locality of preassociated tones

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

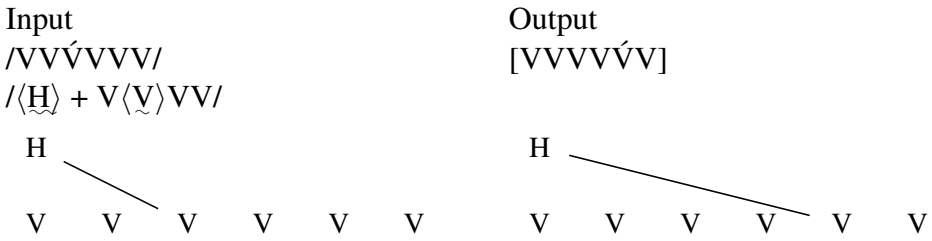
In Rimi, a process of bounded tone shift will cause a preassociated tone to delink from its vowel and re-associate with the subsequent vowel: /VV́VV/→ [VVV́V]. In our encoding, the input is /⟨H⟩ + V⟨V⟩VV/. This function is [1,2]-MISL. We need a locality window of size 1 over the **T**-string because we care if the current tone symbol is a preassociated ⟨H⟩. If yes, then we need a locality window of size 2 over the **V**-string in order to first delink the current preassociated vowel ⟨V⟩ and then associate the tone with the next vowel. A sample MT-FST and derivation are in the appendix (Figure 41).

Figure 15: *Bounded tone shift in Rimi* (Chandlee and Jardine 2019a citing Meyers 1997)



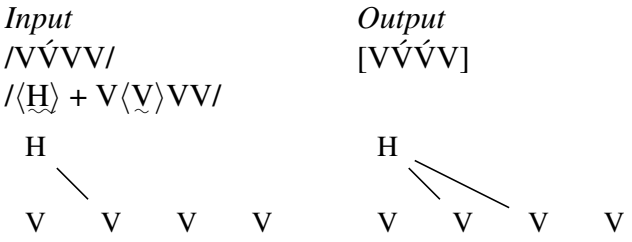
Unlike Rimi, Zigula displays unbounded tone shift whereby a preassociated H is delinked from its preassociated vowel. The tone is re-associated with the *penultimate* vowel which can be at any distance away from the underlyingly preassociated vowel: / VVV́VVV/ or /⟨H̲⟩ + VV⟨V̲⟩VVV/ → [VVVVV́V]. This function is [1,3]-MISL. Given a preassociated ⟨H̲⟩ as a current input tone symbol, an underlying preassociated vowel ⟨V̲⟩ is delinked regardless of context, while the current tone symbol ⟨H̲⟩ 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 42).

Figure 16: *Unbounded tone shift in Zigula* (Chandlee and Jardine 2019a citing Kenstowicz and Kisseberth 1990)



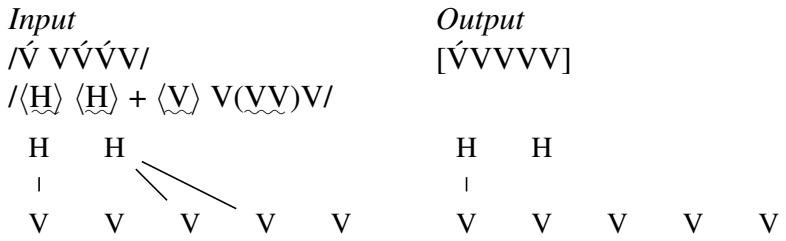
Similar to Rimi, Copperbelt Bemba (heneceforth Bemba) shows bounded tone spread. But instead of delinking a preassociated tone-vowel pair, the preassociated tone spreads to the subsequent vowel: / VV́VV/ or /⟨H̲⟩ + V⟨V̲⟩VV/ → [VV́V́V]. This is [1,2]-MISL. The only difference from Rimi is that an input preassociated vowel ⟨V̲⟩ is not delinked, i.e. it keeps its tone in the output. A sample MT-FST and derivation are in the appendix (Figure 43).

Figure 17: *Bounded tone spread in Bemba* (Chandlee and Jardine 2019a citing Bickmore and Kula 2013)



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 delinked and toneless: /V́ VV́V́V/ or /⟨H⟩ ⟨H⟩ + ⟨V⟩ V(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 high vowels. But given the two multi-input representation, this function is [3,1]-MISL. A locality window of size 3 is needed on the **T**-string in order to check if the current input tone symbol is a phrase-final ⟨H⟩ and succeeds another high tone. If yes, then any currently read input vowels are delinked. We do not provide an MT-FST; it is difficult to draw with our preassociation alphabet, but it is still local.

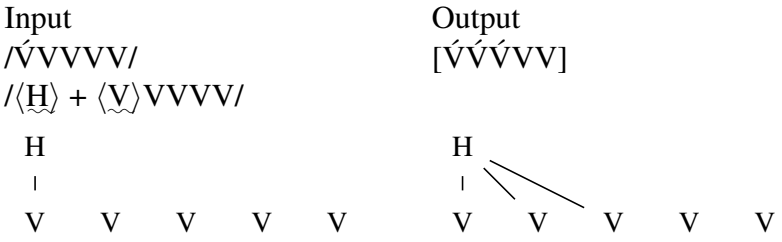
Figure 18: *Unbounded deletion in Arusa* (Chandlee and Jardine 2019a citing Odden 1994)



Finally, Ndebele has unbounded spreading of a preassociated H tone up until the ante-penultimate vowel: /V́VVVV/ or /⟨H⟩ + ⟨V⟩VVVV/ → [V́V́V́VV]. This process is [1,3]-MISL. Reading from right-to-left, the last two vowels surface as toneless. But if the current tone symbol is a preassociated ⟨H⟩, then any vowel before the penult and up till ⟨V⟩ surfaces as high 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 44).[12]

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

Figure 19: *Unbounded spreading in Ndebele* (Chandlee and Jardine 2019a citing Sibanda 2004; Hyman 2011)



5 Nuances in locality and non-locality

The previous sections show how various tonal processes are computationally local when defined over multiple input strings. In this section, we go over some complications which can cause non-locality. To clarify, we do not argue that all tonal processes are MISL, but that a substantial proportion are. But to make tone local, we need flexibility in our input representations, i.e., using multi-input representations. Specifically, we show the following:

1. Tonal patterns can be directional, and that directionality affects locality (§5.1).
2. Contour tones can be feasibly incorporated into MT-FSTs while still maintaining locality (§5.2).
3. Tier-conflation can create non-locality (§5.3).
4. There are tonal patterns that are not local over the input (§5.4).

5.1 Computation of directionality

It it is a common fact that phonological processes show directionality (Itô 1986; Howard 1972; Itô 1989). Computationally, directionality matters (Johnson 1972; Eisner 2000). 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.

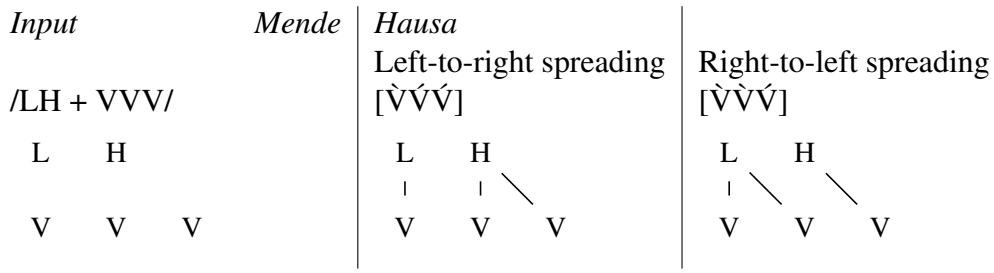
For directionality in segmental phonology, left-to-right vs. right-to-left rules require different classes of formal functions. A simple example comes from the computation of vowel harmony (Gainor et al. 2012; Heinz and Lai 2013; Heinz 2018). Progressive (left-to-right) harmony requires left-subsequential functions. These are functions computed by deterministic FSTs that read the input left-to-right. Analogously, regressive (right-to-left) harmony requires right-subsequential functions. Outside of vowel harmony, directionality is likewise 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 ISL and right-to-left ISL functions (Chandlee and Heinz 2018). This is because ISL functions essentially act like the application of simultaneous SPE rules.

For tone however, directionality matters. An MISL function can have a different expressivity depending on its reading direction. Tone provides a simple illustration. 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. For Mende, the MT-FST must read the input left-to-right, while the MT-FST for Hausa must read the input right-to-left. For Hausa, if the input were read left-to-right, then the MT-FST would not be MISL, and it 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.

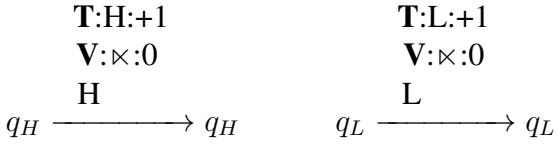
5.2 Representation and factorization of contour tones

Besides directionality, another computational complication concerns contour tones. In typical left-to-right spreading, contour tones are generated when there are more tones than vowels: /HL + V/ $\rightarrow$ [V̂]. Contour tones however can be feasibly incorporated.

To generate contour tones, we adopt a *factorized* approach whereby we first generate 1-to-1 association and final spread. For the Mende MT-FST in Figure 9, we know there are more tones than vowels if we have reached the end $\ltimes$ of the **V**-tape but still see tones on the **T**-tape. In this

situation, we can configure the MT-FST to output unassigned *tones* instead of toned vowels: /HL + V/ → //V́ L//. This requires adding the following two transition arcs:

Figure 21: Transition arcs for outputting unlinked tone in Mende



The output string of this modified MISL function is then fed to a post-processing ISL function. The function will turn any contiguous substring of a toned vowel and some tones into a contour-toned vowel: //V́ L// → [V̂]. This process would work as long the number of possible types of contour tones is finite: rising V̌, falling V̂, etc.

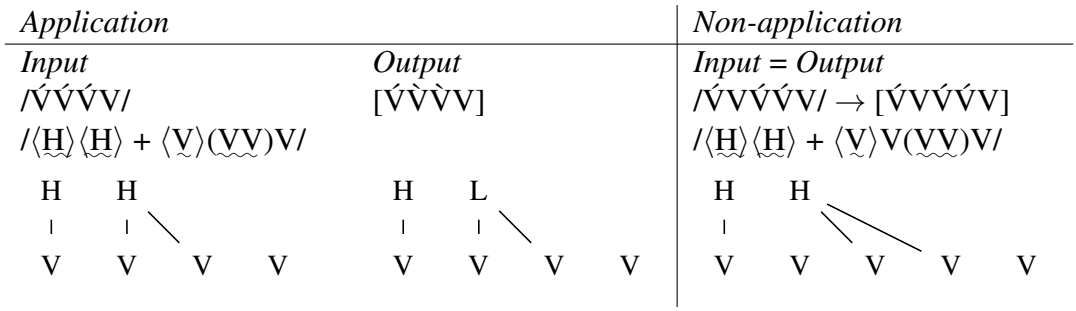
Besides the factorized approach, there are many conceivable alternatives. One alternative is a *compositional* approach, whereby the MT-FST's function and the post-processing ISL function are composed into a single multi-input function. This function would be MISL but with potentially large locality windows. Another alternative is representational. Inspired from Yli-Jyrä (2015), we can enrich our alphabet to include the symbols / and \ as part of multi-character symbols on the vowel-string. Given an input of /⟨H⟩ ⟨L⟩, (V/ V)/ where space marks the separation of multicharacter symbols, the slash / means that the first tone is associated to the first vowel while the second tone to the two vowels. Similarly for /⟨H⟩ ⟨L⟩, (V \V)/, the first tone is associated with the two vowels while the second tone with the second vowel. For space, we do not develop or entertain these alternatives.

5.3 Tier-conflation and output structure

The third issue we discuss concerns the 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 essentially doesn't apply tier-conflation.

In general, the mechanism of tier-conflation does have 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. Here, 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: /V́V́V/ or /⟨H⟩⟨H⟩ + ⟨V⟩(VV)V/ maps [V́V̀V]. If the two tones aren't contiguous over the **V**-string, then the rule does not apply: /V́VV́V/ or /⟨H⟩⟨H⟩ + ⟨V⟩V(VV)V/ is faithfully mapped to [V́VV́V].

Figure 22: *Bounded Meussen's rule in Luganda* (Chandlee and Jardine 2019a citing Hyman and Katamba 2010)



Because of the dependence on contiguity over the both the tone-string and vowel-string, this tonal process is MISL if and if only we do not apply tier-conflation. To illustrate, assume that the function generates only *one* output string. Consider first the case where the rule applies in /V́V́V́V/. The **T**-string is ⟨H⟩⟨H⟩, and the **V**-string contains two vowels preassociated to the two different tones which we *represent* with butting brackets: /⟨H⟩⟨H⟩ + ⟨V⟩(VV...V)/. The first vowel ⟨V⟩ maps to a high V́. The second vowel (V will map to a surface low toned vowel V́ because the two tones are contiguous.

The second vowel (V starts a span of preassociated vowels. But for the other vowels like the span-final V), an MISL function cannot keep track if this vowel was part of a preassociated vowel span which succeeded another span, i.e. it can't know if V) is preceded by the substring ⟨V⟩ (V or not. Over the **V**-string, this type of information is long-distant and thus non-local. In contrast, consider the case of non-application in /V́VV́V́V/ or /⟨H⟩⟨H⟩ + ⟨V⟩V(VV...V)/. The span-final vowel V) should not undergo the rule because its span (VV...V) did not immediately follow the the preassociated vowel ⟨H⟩.

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

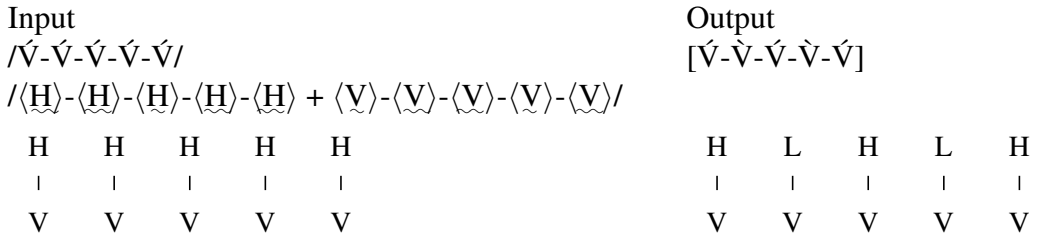
To summarize, the representation of the output *does* affect the computation of tone. Tier-conflation has subtle empirical effects which are difficult to find (McCarthy 1986; Bat-El 1988). In this section, we showed that tier-conflation also has computational effects which impact the generative capacity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

5.4 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 do not 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 23: *Alternating Meussen's rule in Shona* (Chandlee and Jardine 2019a citing Odden 1986)



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 Chandlee (2014)'s Output Strictly Local (OSL) functions over n-ary functions, in the same way that MISL generalizes ISL, 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 how MT-FSTs can be used to compute tone, and that most tonal possesses are computationally local when computed over these machines. In the following 3 sections, we extend the local multi-input analysis to Semitic root-and-pattern morphology. But first, this section goes 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, 1981a) utilized these non-linear representations to model the non-concatenative phonology of root-and-pattern

morphology (RPM) in Semitic. Since then, these structures have likewise been used for processes that reference prosodic templates, such as truncation, reduplication, and clipping (McCarthy and Prince 1986).

For most reported 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 unit: *Jeffrey* → *Jeff*.

Figure 24: 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 must output only the segments which are licensed by the template. The concatenated approach is subtly different. 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 1981b), 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 utilized affixed templates. Early work in prosodic morphology argued that the template was composed of CV slots (McCarthy 1979, 1981a; 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; Gafos 1998a; Downing 2006; Inkelas and Zoll 2005; Alber and Arndt-Lappe 2012). Currently, attention has swung back to the concatenated approach, oftentimes with the addition of sub-segmental representations such as moras (Bye and Svenonius 2012; Bermúdez-Otero 2012; Saba Kirchner 2013; Trommer and Zimmermann 2014; Guekguezian 2017; Paschen 2018).

Computationally however, the three approaches are distinct in terms of their input structure. Each have their own computational consequences and different histories of use. In computational morphology, 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)

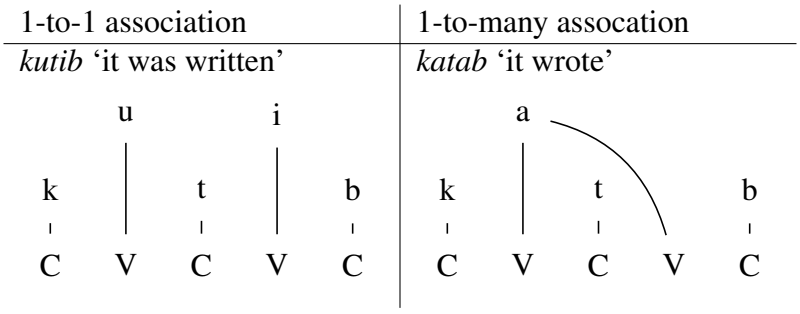
adopted a derivational approach and goes through a litany of morphological processes. She shows many of them are local, including truncation. But, a concatenated approach would also be computationally local as long as the template is concatenated on the same side of its prosodic targets: CVC-*Jeffrey*, 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 1981a), an Arabic word like *kutib* consists of three morphemes or 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.[13]

Figure 25: *Associations in Semitic root-and-pattern morphology*



For Semitic RPM, there is work on designing large-scale computational resources for industrial use (Kiraz 2001; Beesley and Karttunen 2003; Soudi et al. 2007; Farghaly and Shaalan 2009; Attia et al. 2011). There is also progress on learning Semitic morphology (Daya et al. 2007; Clark 2007; Fullwood and O'Donnell 2013; Dawdy-Hesterberg and Pierrehumbert 2014). But in terms understanding the formal properties of templatic morphology, there has been less work.

Mathematically, there is little discussion on the locality or non-locality of RPM (cf. Chandlee 2017). One 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. As with tone, these linear representations cause problems in terms of state complexity, memory complexity, or state explosion, because they utilize

[13]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.

a significant amount of listing, whether listing all finitely possible roots, vocalized templates, or whole words. We discuss this problem further in §8.3.

With this background, we argue that unlike the rest of prosodic morphology, 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 in models of 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, it is also computationally local in most cases. We focus on Standard Arabic because of its rich array of templatic processes. Although we focus on Semitic RPM as our main case study, our results extend to other potential multi-input analysis for prosodic morphology.

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 ues 3 input strings for consonants **C**-string, vowels **V**-string, and the prosodic template **P**-string. The MT-FST in Figure 26 computes this one-to-one association. Unlike for tone, this MT-FST for RPM uses 3 input tapes: a **C**-tape, **V**-tape, and **P**-tape.[14]

Figure 26: *MISL MT-FST for one-to-one template association in* kutib

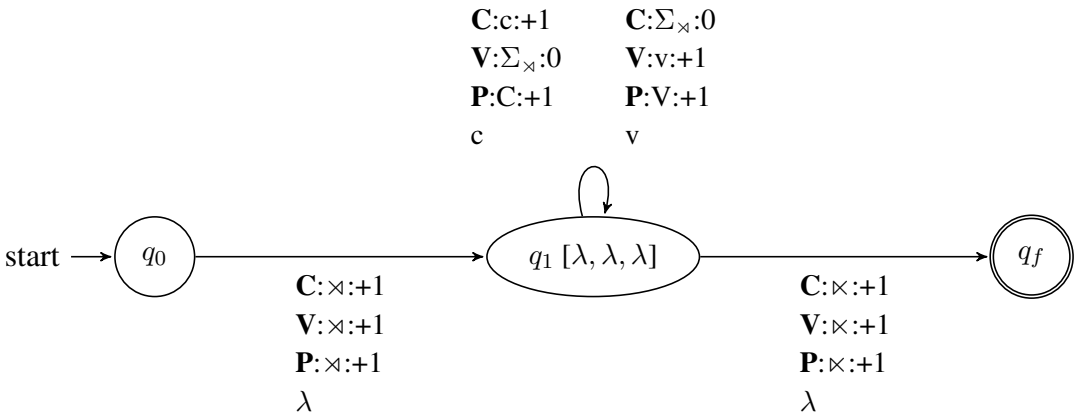


Figure 27: MT-FST for 1-to-1 slot-filling.

[14]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**.

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

	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$\ltimes$		$\underline{\rtimes}$ui$\ltimes$		$\underline{\rtimes}$CVCVC$\ltimes$		
2.	q_1	$\rtimes$:+1	$\rtimes\underline{\text{k}}tb\ltimes$	$\rtimes$:+1	$\rtimes\underline{\text{u}}i\ltimes$	$\rtimes$:+1	$\rtimes\underline{\text{C}}$VCVC$\ltimes$	λ	
3.	q_1	k:+1	$\rtimes$k$\underline{\text{t}}$b$\ltimes$	u:0	$\rtimes\underline{\text{u}}i\ltimes$	C:+1	$\rtimes$C$\underline{\text{V}}$VC$\ltimes$	k	k
4.	q_1	t:0	$\rtimes$k$\underline{\text{t}}$b$\ltimes$	u:+1	$\rtimes$u$\underline{\text{i}}\ltimes$	V:+1	$\rtimes$CV$\underline{\text{C}}$VC$\ltimes$	u	ku
5.	q_1	t:+1	$\rtimes$kt$\underline{\text{b}}\ltimes$	i:0	$\rtimes$u$\underline{\text{i}}\ltimes$	C:+1	$\rtimes$CVC$\underline{\text{V}}$C$\ltimes$	t	kut
6.	q_1	b:0	$\rtimes$kt$\underline{\text{b}}\ltimes$	i:+1	$\rtimes$ui$\underline{\ltimes}$	V:+1	$\rtimes$CVCV$\underline{\text{C}}\ltimes$	i	kuti
7.	q_1	b:+1	$\rtimes$ktb$\underline{\ltimes}$	$\ltimes$:0	$\rtimes$ui$\underline{\ltimes}$	C:+1	$\rtimes$CVCVC$\underline{\ltimes}$	b	kutib
8.	q_f	$\ltimes$:+1	$\rtimes$ktb$\ltimes$	$\ltimes$:+1	$\rtimes$ui$\ltimes$	$\ltimes$:+1	$\rtimes$CVCVC$\ltimes$	λ	kutib

To better understand why 1-1 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, 1-1 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. This cannot be said for many other computational models of Semitic RPM. Semitic RPM has been a frequent target of computational modeling and 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 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).

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. But although synchronous MT-FSTs can model Arabic morphology, they sacrifice locality. The reason is because synchronous

MT-FSTs are equivalent to single-tape FSAs, thus effectively enforcing non-local computation in RPM. To illustrate, Figure 29 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 29: *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 association in *kutib* was computationally local when computed over multiple inputs. Arabic morphology uses many other possible combinations of roots, vowels, and templates. In this section, we go through a substantial chunk of Arabic morphology and show that it is also computationally local. Among the many Semitic languages, we focus solely on Arabic.[15]

For illustration, we distinguish between cases of 1-to-1 association or slot-filling, and 1-to-many association. Both types are local. We likewise discuss cases of non-locality from reduplication and loanword adaptation in the subsequent section. The data is amassed from McCarthy (1981a). When readable, we provide example MT-FSTs in the appendix. In this section, we informally explain why some process is local or not.

7.2.1 1-to-1 slot filling

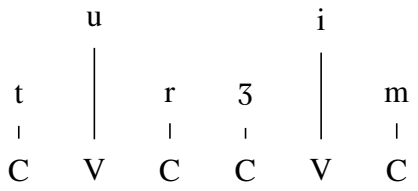
The simple example of 1-to-1 association in *kutib* was MISL. In Arabic, various other patterns of 1-to-1 association are likewise local. This includes cases of quadriliteral roots, nativization of borrowed roots, and affixed templates.

The word *kutib* represents the default case in Semitic RPM whereby triliteral root surfaces in a CVCVC template. Some Arabic roots are however quadriliteral and have four consonants:

[15]We do not formalize RPM functions in broken plurals (Hammond 1988; 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.

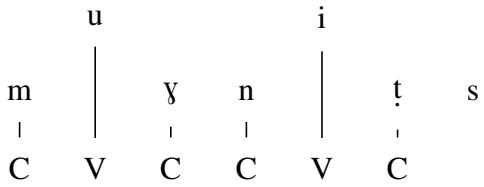
C=*trʒm*. These roots usually take templates that have at least four consonantal slots: **P**=*CVCCVC*. Combined with the vocalism *ui*, the combination of this root and template generates the word *turʒim*. The generation is local and it uses the same [1,1,1]-MISL function that's modeled by the MT-FST in Figure 27. A sample derivation is in the appendix (Figure 45).

Figure 30: *1-to-1 association in 4-consonant roots:* turʒim 'was translated'



The reason why this is local is simple. There is no interaction between one consonant and another in terms of which slot to take. Upon seeing a consonant *c* on the **C**-string and a consonantal slot *C* on the **P**-string, the two elements are associated and surface as a pronounced consonant *c*. Similar locality is found in words where there are more root consonants **C**=*mɣnṭs* than consonant slots **P**=*CVCCVC* (Figure 31). The output shows deletion of the additional consonant: *muɣniṭ* not **muɣniṭs*. This is also [1,1,1]-MISL. It requires minor modifications to the MT-FST in Figure 27. A sample MT-FST and derivation are in the appendix (Figure 46).

Figure 31: *1-to-1 association in 5-consonant roots:* maɣnaṭ 'be magnetized'



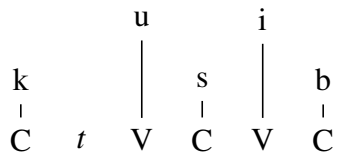
Intuitively, the machine processes the strings left-to-right. By the time we reach the final consonant *m* on the **C**-string, all the consonantal slots on the **P**-string have been associated and filled. The machine simply does not output any more consonants after reaching this point.

Lastly, just like tone, some situations involve both 1-to-1 slot-filling and pre-associated elements. Given a root **C**=*ksb*, some outputs show an additional affix, e.g. the infix ⟨*t*⟩ in *k*⟨*t̲*⟩*usib*. The affix ⟨t̲⟩ is pre-associated to a slot after the first consonant. Pre-associated templates can be computed either representationally or derivationally. Both are local.[16] The representational route is to enrich the template with the affix itself: **P**=*CtVCVC* (Hudson 1986). The root and template are then

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

combined to generate *k⟨t⟩usib*. This function is [1,1,1]-MISL. It requires a minor modification to the MT-FST from Figure 27. A sample MT-FST and derivation are in the appendix (Figure 47).

Figure 32: *1-to-1 association with preassociation:* k⟨t⟩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**-string, just as the **C**-string ignores all vocalic slots on the **P**-string. A derivational alternative is to derive *k⟨t⟩usib* from an un-affixed base *kusib* by infixing ⟨*t*⟩ (McCarthy 1993). Generating *kusib* from [*ksb, ui, CVCVC*] is [1,1,1]-MISL. Infixing ⟨*t*⟩ onto *kusib* is 2-ISL. The representational route can be interpreted as 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 input root consonant or inflectional 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. Many of these processes are still computationally local. However, as with pre-associated affixes, they require a more nuanced representations.

In some situations, Arabic morphology utilizes final spreading as in *katab* (Figure 33a). The word consists of the following input strings: **C**=*ktb*, **V**=*a*, **P**=*CVCVC*. The vocalism **V** consists of only one vowel *a* because of the Obligatory Contour Principle (McCarthy 1981a). The vowel *a* undergoes final spread by being 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, not $k_V = 1$. Finding the final vowel requires a window of size 2 on the **V**-string (...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 48).[17]

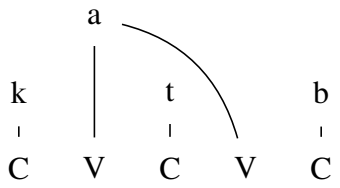
Consonants can also undergo final spread, especially in biliteral or 2-consonant roots. The combination of **C**=*sm*, **V**=*a*, and **P**=*CVCVC* generates *samam* (Figure 33b).[18] This is [2,1,1]-MISL,

[17]Interestingly, final *i* does not spread (McCarthy 1981a: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 (§7.3.2).

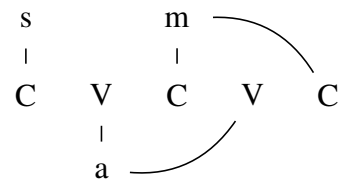
[18]Since McCarthy (1981a), the analysis of final consonant spread has been controversial (Hudson 1986; Hoberman

Figure 33: *1-to-many association with final spreading*

(a) Vowel spreading: katab 'it wrote'



(b) Consonant spreading: samam 'he poisoned'



analogous to the final spread of vowels except that the locality window is now larger over the **C**-string instead of the **V**-string.

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 not local. In that situation, the **P**-string is still CVCVC, but the input vocalism *a* would be □*a*□□□ (Figure 34). 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 34: *Alignment of* katab *with a synchronous MT-FSA.*

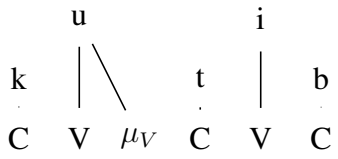
Input Tapes	**C**:	k	□	t	□	b
	V:	□	a	□	□	□
	P:	C	V	C	V	C
Output Tape		k	a	t	a	b

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 35a) or *kuttib* with a geminate *t* (Figure 35b). Like pre-associated affixes, medial spread can be analyzed either representationally or derivationally. An alternative edge-in analysis is discussed in §7.3.2.

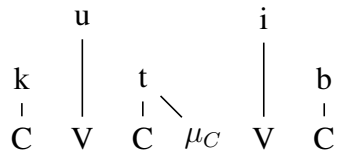
1988; Yip 1988; McCarthy 1993; Gafos 1998b; Bat-El 2006). Alternative analyses involving reduplication, preference for local spreading, or right-to-left association can be potentially non-local and are discussed in §7.3.2. Computationally, Beesley (1998) formalizes final consonant spread with a special symbol X. This formalization is [2,1,1]-MISL, just like for medial spread in *kuttib*.

Figure 35: *1-to-many association with medial spreading*

(a) Vowel spreading: kuutib 'be corresponded'



(b) Consonant spreading: kuttib 'be caused to write'



We focus on consonant spreading or gemination. The representational route involves enriching the template with a special symbol, i.e., a consonant mora μ_C in **P**=*CVC*μ_V*VC* (Kay 1987; McCarthy 1993; Beesley 1998). With this template, generating *kuttib* is [2,1,1]-MISL with k_C=2 over the **C**-string. A sample MT-FST and derivation are in the appendix (Figure 49). Informally, whenever the machine sees a consonantal mora node 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.

7.3 Apparent non-locality

The previous section went through a litany of Arabic morphological patterns. The take-away was that, over multiple inputs, the bulk of Arabic root-and-pattern morphology is local. We now go over a handful of cases which can be local or not depending on one's analysis. These involve reduplication and loanword adaptation. Both can be given local analyses with suppletion. The latter brings up the interesting issue of edge-in association and lookahead.

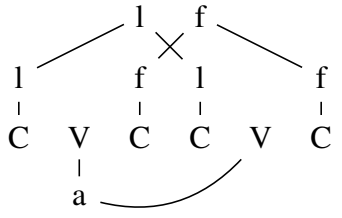
7.3.1 Reduplication

Arabic RPM shows intensive reduplication which varies by root size (Broselow and McCarthy 1983): root doubling for biconsonantal roots in *laflaf* (Figure 36a) and first-C copying for triconsonantal roots in *barbad* (Figure 36b). Root-doubling is analogous to total reduplication. Initial-C copying involves copying the first consonant of the root and placing it in a prespecified spot on the template.[19]

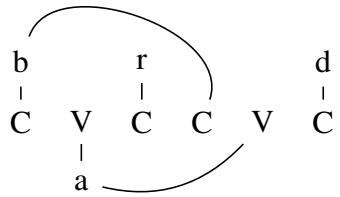
[19]Technically, the relevant inputs need to be annotated to trigger reduplication, e.g. initial-C copying with **P**=*CVCFVC* and root doubling with either **C**=*lf*-RED or **P**=CVCRVR. As with loanword adaptation, this looks like suppletive allomorphy for the **P**-string conditioned by root size. We abstract away from this for clarity.

Figure 36: Intensive reduplication in Arabic

(a) *2-consonant roots:* laflaf 'wrapped'



(b) *3-consonant roots:* barbad 'shaved unevenly'



Reduplication is computationally challenging. Cross-linguistically, partial reduplication patterns can range from being ISL to subsequential (Chandlee and Heinz 2012). Total reduplication is above the subsequential threshold and cannot be modeled by 1-way FSTs but requires deterministic 2-way FSTs (Dolatian and Heinz 2018, 2020). If we assume that there's no bound on the size of the root, then root-doubling cannot be computed by a MISL function for any $\vec{k}$. The function would need a 2-way MT-FST which could go back and forth on the **C**-tape. Similarly for first-C copying in *barbad*, if we assume that there's no bound on the number *n* of consonants between the two copies of the root-initial consonant, then the function is not MISL for any $\vec{k}$. Analogously to subsequential functions over single-input FSTs, root-initial copying would be Multi-Subsequential.

However, the assumption on root size is not correct. All roots which undergo the above reduplication processes have a bounded size (2 or 3). If we discard this assumption, then both reduplicative processes are MISL for a large value of $\vec{k}$.[20]

7.3.2 Local spreading in loanword adaptation

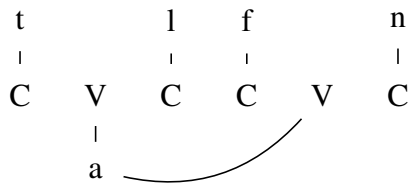
The second case of potential non-locality is loanword adaptation in Arabic verbs. For these verbs, the most productive template is *CVCCVC* with the vocalism *a*: *CaCCaC* (Bat-El 2011). When a borrowed consonantal root has four consonants, the template is filled with 1-to-1 slot filling of consonants: *telephone* [telefon] and *talfan* (Figure 37a). But when a borrowed root has three consonants, then the input undergoes medial gemination: *SMS* and *sammas*, not final spread **samsas* (Figure 37b).

There are many ways to analyze this difference between three vs. four-consonant roots. One is suppletive allomorphy for the prosodic template **P**-string: 4-consonant roots use the template *CVC-CVC*, while 3-consonant roots use the template $CVC\mu_C VC$. Choosing the template is ISL-4. Once chosen, the root, vocalism, and template can then be submitted to an MISL function. This analysis is plausible because, outside of loanword adaptation, Semitic templates *do* have suppletion condi-

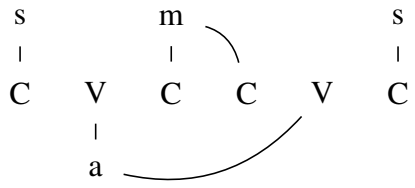
[20]The value of the $\vec{k}$ is [3,1,1] for initial-C copying, but [3,1,3] for root-doubling because the function keeps track of the root size and the current C-slot. It should be noted that Dolatian and Heinz (2020) argue that all types reduplication should be handled by 2-way devices in order to express the linguistically-informed strong-generative capacity of reduplication.

Figure 37: Loanword adaptation in Arabic verbs

(a) *4-C root:* 'telephone' > [talfan] 'he phoned'



(b) *3-C root:* 'SMS' > [sammas] 'he SMS-ed'



tioned by root-size: the comparative in Egyptian Arabic is *VCCVC* for triconsonantal roots: *kbr* → *akbar*, but *VCVCC* for biconsonantal roots: *ʃd* → *aʃadd* (Davis 2017; Davis and Tsujimura 2018).

An alternative is to use a template *CVCCVC* without any representational markup for gemination. The correct outputs are generated based on avoiding non-local spreading. For a three-consonant root, medial gemination is generated because the grammar (in OT-parlance) prefers outputs with local spreading of consonants *sa***mm***as* instead of outputs with non-local spreading **sam***s***a***s*. An analogous anti-long-distance spreading mechanism has been proposed for medial gemination and for the fact that final *i* cannot spread (§7.2.2) (Hudson 1986; Hoberman 1988; Yip 1988).[21] Computationally, the choice of local spreading depends on the following information:

1. Having the context CCV on the template: $k_P = 3$ on the **P**-string
2. Being the final consonant in the root or not: $k_C = 2$ on the **C**-string
3. The existence of an additional *C* slot on the template: $k_P = |V_y| + 1$ on the **P**-string where the template is either $\text{XCCV}_y\text{C}\ltimes$ or $\text{XCCV}_y\ltimes$

The last condition is important. Consider the contrast in *kuttib* and *kutba* 'writers' derived from the templates $C_1V_xC_2C_3V_yC_4$ and $C_1V_xC_2C_3V_y$. The C_2C_3 substring in $C_1V_xC_2C_3V_yC_4$ maps to gemination: *kuttib*, while the *CC* substring in *CVCCV* maps to 1-to-1 spreading: *kutba*. The choice depends on if the C_2C_3 substrings precedes an extra consonant slot C_4 on the template or not. If there is no bound on the number of intervening vowels V_y, then the function is not MISL for any k_P. If there is a bound, then it is MISL for a k_P which is sufficiently large enough to encode these contexts. In Arabic, V_y can be at most two vowels slots in order to encode long vowels: *kuttaab* 'writers'. This makes the function MISL with $k_P = 5$ on the **P**-string, $k_C = 3$ on the **C**-string.

For loanword adaptation, the main take-away is that the choice of analysis has computational consequences. A generalized spreading analysis without any bounds on the size of intervening material is not computationally local. In contrast, suppletion is local.

[21] These have also been analyzed with edge-in association. Instead of association operating from left-to-right, Yip (1988) argues that these templates are simultaneously or consecutively right-to-left and left-to-right. Such an analysis though has unclear computational expressivity; we conjecture that it may be analogous to Weak Determinism (Heinz and Lai 2013) over multiple inputs.

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. For the most part, the multi-input analysis is agnostic towards many theoretical 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 interpretation of elements on the tone tier and segmental tier 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.

The bulk of theoretical and psycholinguistic results show that Semitic RPM *does* involve template-filling (Prunet 2006; Aronoff 2013; Kastner 2016). But the formulation of templates is controversial (Ussishkin 2011; Bat-El 2011). One hypothesis is that the template is composed of CV slots (McCarthy 1981a). 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 (Tucker 2010; Kastner 2016; Zukoff 2017). Alternatively, the job of the template is done by deriving words from other words via *overwriting* or changing the vowels and consonants (Ussishkin 2005), e.g. *katab+ui→kutib*. Crucially, these debates on the content of prosodic templates in Semitic RPM are likewise manifested in other cases of prosodic morphology, as reviewed in §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. Mathematically, many of the formalizations of templates are notationally equivalent. Whether the prosodic template or **P**-string is up CV units, moras, or both 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 the case of pre-association (§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 discontinuity. That is, the function *katab+ui→kutib* implicitly assumes that the vowels can be separated: *kVtVb+ui→kutib*. The fact that one of the inputs is a template with filled consonants *kVtVb* can be equally 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 and this makes a template be implicitly present in the function. 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

The previous section discussed how our computational results are agnostic towards the phonological meanings of prosodic templates. 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. As before, we show that this issue likewise has dubious computational consequences.

Our MT-FSTs took as input three morphological items, each on its own input tape. For the output *kutib*, the inputs were the root consonants **C**=*ktb*, the inflectional vowels **V**=*ui*, and a prosodic template **P**=*CVCVC*. Crucially, we included the template **P** as part of the input as a morphological primitive.[22] This assumption was made in earlier work on Semitic morphology (McCarthy 1981a). However, recent work on Semitic argues that there is no pre-specified input template (Tucker 2010; Ussishkin 2011; Bat-El 2011; 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 from the phonology via optimizing phonological constraints on syllable structure, autosegmental docking, and word-size requirements. To illustrate, consider a toy Optimality-Theoretic derivation for the word *kutib*. The toy constraints in Figure 38 illustrate the basic idea: constraints on syllable structure will choose the optimal candidate *kutib* for the input *ktb,ui*.

Figure 38: *OT tableau for* /ktb,ui/ *without an input template*

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

Given this difference in input-output structure, it *seems* that MT-FSTs are now superfluous because there are no longer any templates to compute. But this is premature. The function still takes as input two elements (a root and vocalism). Thus, it 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 candidates, which is itself a difficult process to formalize in the first place (Eisner 1997; Karttunen 1998; Frank and Satta 1998; Riggle 2004; Gerdemann and Hulden 2012).[23]

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

[23] If alternatively the input was a single concatenated string *ktb-ui*, the computation involve long-distance metathesis. If there is no bound on the original and final location of metathesized vowels, then this is local. However, the locality

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 infinity and 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 1981a). 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 (Savitch 1993).

For example, given the *hypothetical* root consonants **C**=*ktbm*, vocalism **V**=*uaui*, and 4-syllable template **P**=*CVCVCVCV*, the same MT-FST from Figure 26 for *kutib* would output *ku.ta.bu.mi* with 1-1 matching for the consonants and vowels. 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 do not 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 linear string where the 3 morphological items are separated by some boundary: *ktb-ui-CVCVC*→*kutib*. All existing 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 (the size of the smallest input string). This is essentially the approach taken by Beesley and Karttunen (2000, 2003), among other models.

Alternative 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 the right output: *ktb*+[PASSIVE]→*kutib*; vs. *ktb*+[ACTIVE]→*katab*. This system would work as long as there is a finite list of all possible vocalized templates: CaCaC, CuCiC, etc. This system would again be ISL

window would be substantially large, and at least as large as the maximum size of input roots.

for a potentially large locality window. This window would match 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 with these approaches, either as a multi-input function over an MT-FST or a single-input function over a single-tape FST. For the latter, the single-input function could either use a concatenated approach or a derivational approach.[24] 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, both single-input approaches require that some part of Semitic morphology be finitely sized. This finite bound is either for 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 1981a, 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 also 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.

Thus 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. The output of this process is fed 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 (Golston 1995; 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.

[24]There is likewise a fourth approach. 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. These models work for practical concerns, but they are unlikely to have any cognitive or linguistic support.

9 Conclusion

This paper showed the salience of computational locality in nonlinear morphophonology using formal language theory and automata. 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. Local segmental phonology is only a special case of local suprasegmental phonology.

We demonstrated these results both for tonal phonology and for template-filling processes in root-and-pattern morphology. Our approach is strengthened by our computation rigour in factorizing the representation and computation. By doing so, 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 these analyses and insights to hold for other phonological domains, like harmony, as well as morphological ones.

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

References

Alber, B. and Arndt-Lappe, S. (2012). Templatic and subtractive truncation. In Trommer (2012), pages 289–325.

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.

Bickmore, L. S. and Kula, N. C. (2013). Ternary spreading and the OCP in Copperbelt Bemba. Studies in African Linguistics, 42(2):101–132.

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.

Broselow, E. and McCarthy, J. (1983). A theory of internal reduplication. The Linguistic Review, 3(1):25–88.

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. (2012). Bounded copying is subsequential: Implications for metathesis and reduplication. In Proceedings of the 12th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology, SIGMORPHON '12, pages 42–51, Montreal, Canada. Association for Computational Linguistics.

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.

Clark, A. (2007). Supervised and unsupervised learning of Arabic morphology. In Soudi, A., Neumann, G., and Van den Bosch, A., editors, Arabic computational morphology: Knowledge-based and empirical methods, pages 181–200. Springer.

Clements, G. N. and Ford, K. C. (1979). Kikuyu tone shift and its synchronic consequences. Linguistic inquiry, pages 179–210.

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.

Davis, S. (2017). Some issues for an analysis of the templatic comparative in Arabic with a focus on the Egyptian dialect. pages 129–150.

Davis, S. and Tsujimura, N. (2018). Arabic nonconcatenative morphology in construction morphology. In Booij, G., editor, The Construction of Words: Advances in Construction Morphology, volume 4, pages 315–340. Springer.

Dawdy-Hesterberg, L. G. and Pierrehumbert, J. B. (2014). Learnability and generalisation of Arabic broken plural nouns. Language, cognition and neuroscience, 29(10):1268–1282.

Daya, E., Roth, D., and Wintner, S. (2007). Learning to identify Semitic roots. In Soudi, A., Neumann, G., and Van den Bosch, A., editors, Arabic computational morphology: Knowledge-based and empirical methods, pages 143–158. Springer.

Dolatian, H. and Heinz, J. (2018). Modeling reduplication with 2-way finite-state transducers. In Proceedings of the 15th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, Brussells, Belgium. Association for Computational Linguistics.

Dolatian, H. and Heinz, J. (2020). Computing and classifying reduplication with 2-way finite-state transducers. Journal of Language Modeling, 8:79–250.

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.

Eisner, J. (2000). Directional constraint evaluation in optimality theory. In COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics.

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.

Fullwood, M. and O'Donnell, T. (2013). Learning non-concatenative morphology. In Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL), pages 21–27.

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

Gafos, D. (1998a). A-templatic reduplication. Linguistic Inquiry, 29(3):515–527.

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

Gainor, B., Lai, R., and Heinz, J. (2012). Computational characterizations of vowel harmony patterns and pathologies. In Choi, J., Hogue, E. A., Punske, J., Tat, D., Schertz, J., and Trueman, A., editors, The Proceedings of the 29th West Coast Conference on Formal Linguistics, pages 63–71, Somerville, MA. Cascillida Press.

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.

Golston, C. (1995). Syntax outranks phonology: Evidence from Ancient Greek. Phonology, 12(3):343–368.

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.

Hammond, M. (1988). Templatic transfer in Arabic broken plurals. Natural Language & Linguistic Theory, 6(2):247–270.

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.

Hewitt, M. and Prince, A. (1989). OCP, locality, and linking: The N. Karanga verb. In Fee, E. J. and Hunt, K., editors, Proceedings of the Eighth WCCFL, pages 176–191, Stanford. CSLI Publications.

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.

Hunter, T. (2019). What sort of cognitive hypothesis is a derivational theory of grammar? Catalan Journal of Linguistics, pages 89–138.

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.

Hyman, L. and Katamba, F. X. (2010). Tone, syntax, and prosodic domains in Luganda. In Downing, L., Rialland, A., Beltzung, J.-M., Manus, S., Patin, C., and Riedel, K., editors, Papers from the Workshop on Bantu Relative Clauses, volume 53 of ZAS Papers in Linguistics, page 69–98. Berlin.

Hyman, L. M. (1987). Prosodic domains in kukuya. Natural Language & Linguistic Theory, 5(3):311–333.

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. (1986). Syllable theory in prosodic phonology. PhD thesis, University of Massachusetts, Amherst.

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.

Johnson, C. D. (1972). Formal aspects of phonological description. Mouton, The Hague.

Kager, R., van der Hulst, H., and Zonneveld, W., editors (1999). The prosody-morphology interface. Cambridge University Press, Cambridge.

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.

Kenstowicz, M. and Kisseberth, C. (1990). Chizigula tonology: The word and beyond. In The phonology-syntax connection, pages 163–194. University of Chicago Press, Chicago.

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 Inc.

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. (2019a). Majority rule in Harmonic Serialism. In Hout, K., Mai, A., McCollum, A., Rose, S., and Zaslansky, M., editors, Supplemental Proceedings of the 2018 Annual Meeting on Phonology, pages 243–249, Washington, D.C. Linguistic Society of America.

Lamont, A. (2019b). Precedence is pathological: The problem of alphabetical sorting. In Stockwell, R., O'Leary, M., Xu, Z., and Zhou, Z., editors, Proceedings of the 36th West Coast Conference on Formal Linguistics, pages 243–249, Somerville, MA. Cascadilla Press.

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

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

Leben, W. R. (1978). The representation of tone. In Fromkin, V., editor, Tone: A Linguistic Survey, pages 177–219. Academic Press, Orlando.

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

McCarthy, J. J. (1981b). The role of the evaluation metric in the acquisition of phonology.

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.

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 et al. (1999), pages 218–309.

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.

Meyers, S. (1997). OCP effects in Optimality Theory. Natural Language & Linguistic Theory, 15(4):847–892.

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

Newman, P. (1986). Tone and affixation in Hausa. Studies in African Linguistics, 17(3):249–67.

Newman, P. (2000). The Hausa Language: An encyclopedic reference grammar. Yale University Press, New Haven.

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

Odden, D. (1981). Problems in tone assignment in Shona. PhD thesis, University of Illinois at Urbana-Champaign.

Odden, D. (1986). On the role of the Obligatory Contour Principle in phonological theory. Language, pages 353–383.

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. (1999). Austronesian nasal substitution and other nc effects. In Kager et al. (1999), pages 310–343.

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.

Sibanda, G. (2004). Verbal phonology and morphology of Ndebele. PhD thesis, University of California, Berkeley.

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.

Stabler, E. (1996). Derivational minimalism. In International Conference on Logical Aspects of Computational Linguistics, pages 68–95. 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:⋊:+1

V:⋊:+1

λ

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

C:⋊:+1

V:⋊:+1

P:⋊:+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:⋉:+1

V:⋉:+1

λ

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

C:⋉:+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

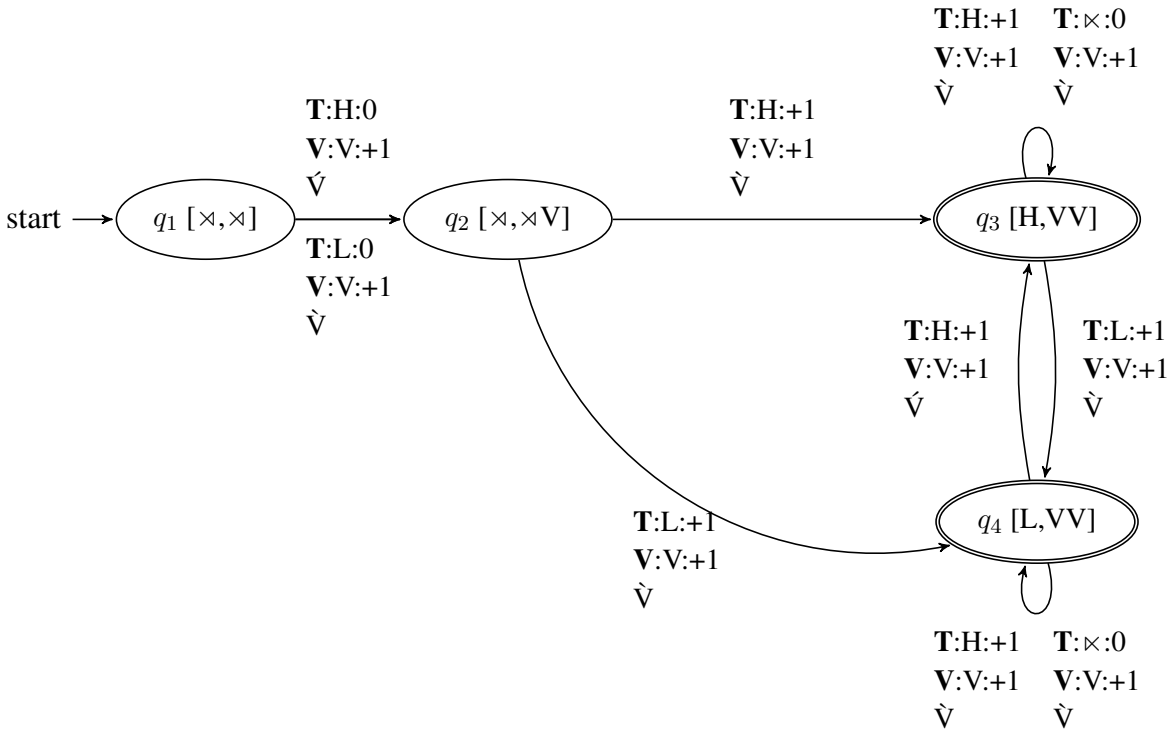
A sample MT-FST and derivation are given for some of the tone processes from §4.1 and §4.3. These include patterns with preassociation, and without preassocation.

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 ṾV̀V́V̀V́V́V́. A [2,3]-MISL MT-FST is provided in Figure 39a, with a sample derivation in Figure 39b.

Figure 39: *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		⋊LHLH⋉		⋊VVVVVVV⋉		
2.	q_1	⋊:+1	⋊LHLH⋉	⋊:+1	⋊VVVVVVV⋉	λ	
3.	q_2	L:0	⋊LHLH⋉	V:+1	⋊VVVVVVV⋉	V̀	V̀
4.	q_4	L:+1	⋊LHLH⋉	V:+1	⋊VVVVVVV⋉	V̀	V̀V̀
5.	q_3	H:+1	⋊LHLH⋉	V:+1	⋊VV⟨V⟩VVVV⋉	V́	V̀V̀V́
6.	q_4	L:+1	⋊LHLH⋉	V:+1	⋊VV⟨V⟩VVVV⋉	V̀	V̀V̀V́V̀
7.	q_3	H:+1	⋊LHLH⋉	V:+1	⋊VV⟨V⟩VVVV⋉	V́	V̀V̀V́V̀V́
8.	q_3	⋉:0	⋊LHLH⋉	V:+1	⋊VV⟨V⟩VVVV⋉	V́	V̀V̀V́V̀V́V́
9.	q_3	⋉:0	⋊LHLH⋉	V:+1	⋊VV⟨V⟩VVVV⋉	V́	V̀V̀V́V̀V́V́V́
10.	q_f	⋉:+1	⋊LHLH⋉	⋉:+1	⋊VV⟨V⟩VVVV⋉	λ	V̀V̀V́V̀V́V́V́

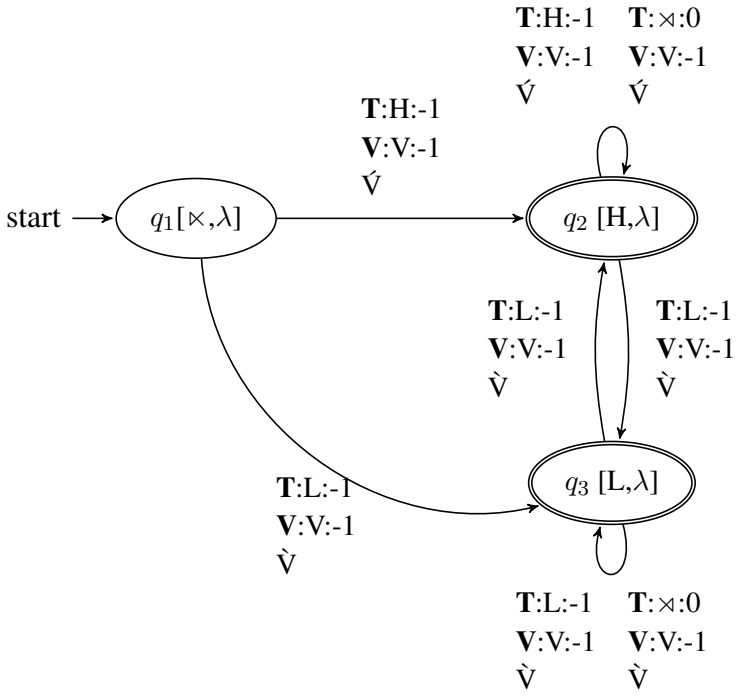
A.2.2 Hausa right-to-left spreading

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 40a, with a sample derivation

in Figure 39b. The FST processes the input string-tuple from right to left using the -1 direction parameter.

Figure 40: *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		⋊LH⋉		⋊VVV⋉		
2.	q_1	⋉:-1	⋊L**H**⋉	⋉:-1	⋊VV**V**⋉	λ	
3.	q_2	H:-1	⋊**L**H⋉	V:-1	⋊V**V**V⋉	V́	V́
4.	q_3	L:-1	⋊**L**H⋉	V:-1	⋊**V**VV⋉	V̀	V̀V́
5.	q_3	⋊:0	⋊**L**H⋉	V:-1	⋊**V**VV⋉	V̀	V̀V̀V́
6.	q_f	⋊:-1	⋊LH⋉	⋊:-1	⋊VVV⋉	λ	V̀V̀V́

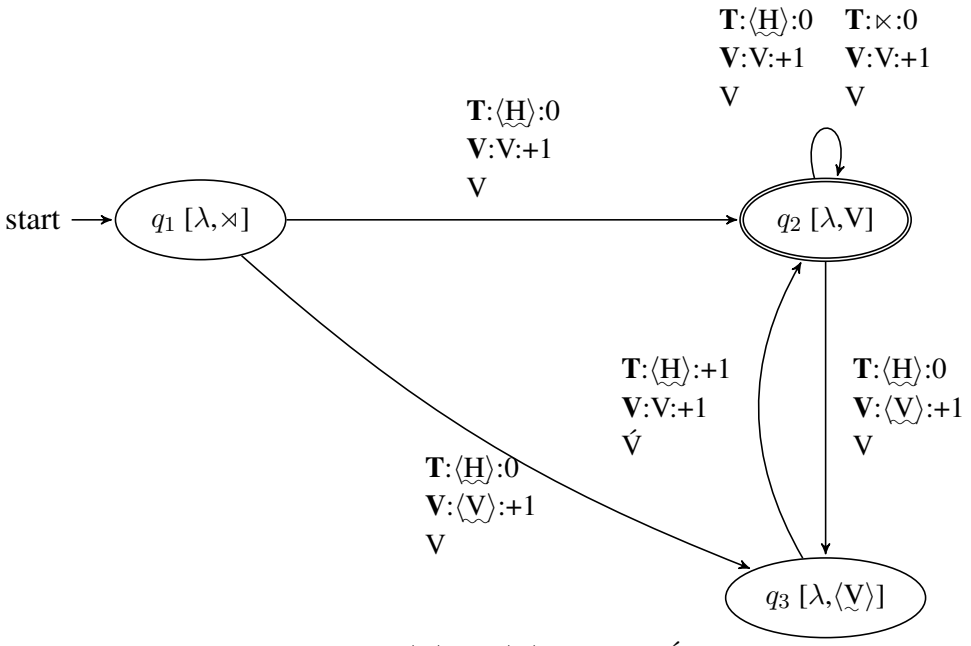
A.2.3 Rimi bounded tone shift

In Rimi (Figure 15), a preassociated tone will shift one vowel to the right: /⟨H⟩ + V⟨V⟩VV/ is mapped to VVV́V. This function is modeled by the [1,2]-MISL MT-FST in Figure 41a, with a sample derivation in Figure 41b. We assume that the only possible underlying tone string is a

string of preassociated H tones.[25]

Figure 41: *Bounded tone shift in Rimi*

(a) *MT-FST for Rimi*



(b) *Derivation of* /⟨H⟩ + V⟨V⟩VV/→VVV́V *in Rimi*

	Current State	**T**-tape Arc	**T**-tape Next	**V**-tape Arc	**V**-tape Next	Output symbol	Output String
1.	q_0		⋊⟨H⟩⋉		⋊V⟨V⟩VV⋉		
2.	q_1	⋊:+1	⋊⟨H⟩⋉	⋊:+1	⋊V⟨V⟩VV⋉	λ	
3.	q_2	⟨H⟩:0	⋊⟨H⟩⋉	V:+1	⋊V⟨V⟩VV⋉	V	V
4.	q_3	⟨H⟩:0	⋊⟨H⟩⋉	⟨V⟩:+1	⋊V⟨V⟩VV⋉	V	VV
5.	q_2	⟨H⟩:+1	⋊⟨H⟩⋉	V:+1	⋊V⟨V⟩VV⋉	V́	VVV́
6.	q_2	⋉:0	⋊⟨H⟩⋉	V:+1	⋊V⟨V⟩VV⋉	V	VVV́V
7.	q_f	⋉:+1	⋊⟨H⟩⋉	⋉:+1	⋊V⟨V⟩VV⋉	λ	VVV́V

A.2.4 Zigulu unbounded tone shift

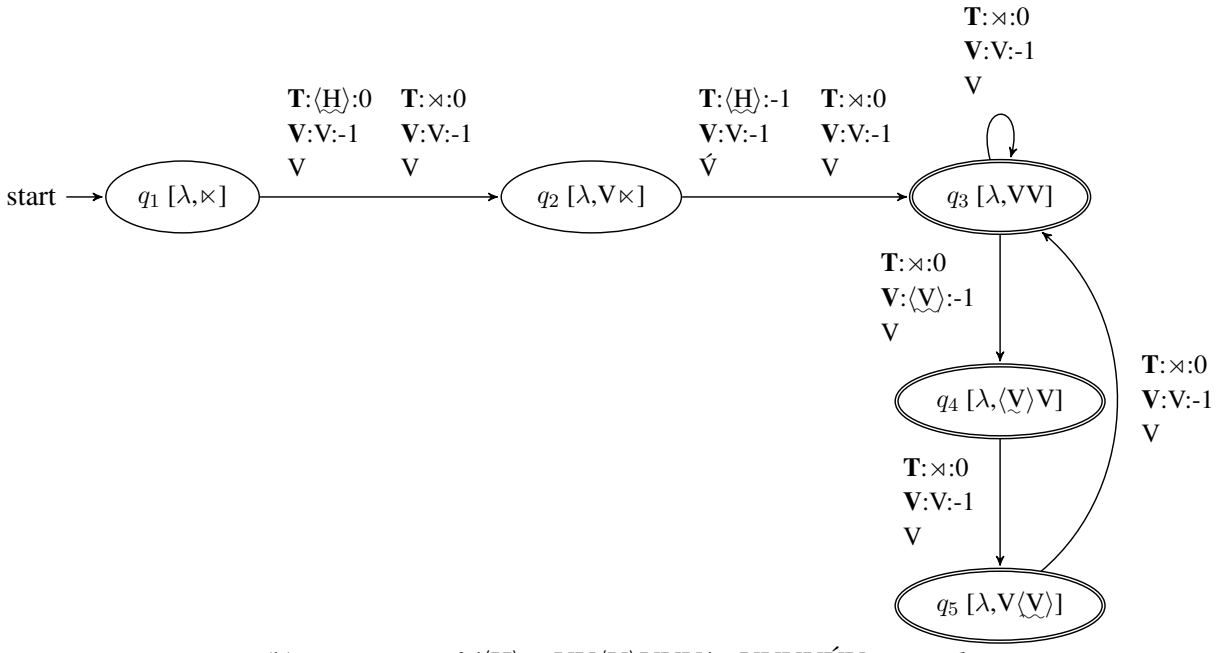
In Zigulu (Figure 16), unbounded tone shift causes a preassociated H tone to shift to the penultimate vowel: /⟨H⟩ + VV⟨V⟩VVV/ maps to VVVVV́V. This function is modeled by the [1,3]-MISL MT-FST in Figure 42a, with a sample derivation in Figure 42b. For easier illustration, the MT-FST

[25]Final preassociated vowels do not undergo tone shift: /⟨H⟩ + VVV⟨V⟩/→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 ⋊⋉ or a single preassociated H tone ⋊⟨H̲⟩⋉.[26]

Figure 42: *Unbounded tone shift in Zigulu*

(a) *MT-FST for Zigulu*



(b) *Derivation of* /⟨H̲⟩ + VV⟨V̲⟩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		⋊⟨H̲⟩⋉		⋊VV⟨V̲⟩VVV⋉		
2.	q_1	⋉:-1	⋊⟨H̲⟩⋉	⋉:-1	⋊VV⟨V̲⟩VV̲V⋉	λ	
3.	q_2	⟨H̲⟩:0	⋊⟨H̲⟩⋉	V:-1	⋊VV⟨V̲⟩V̲VV⋉	V	V
4.	q_3	⟨H̲⟩:-1	⋊⟨H̲⟩⋉	V:-1	⋊VV⟨V̲⟩V̲VV⋉	V́	V́V
5.	q_3	⋊:0	⋊⟨H̲⟩⋉	V:-1	⋊VV⟨V̲⟩VVV⋉	V	VV́V
6.	q_4	⋊:0	⋊⟨H̲⟩⋉	⟨V̲⟩:-1	⋊VV̲⟨V̲⟩VVV⋉	V	VV́V
7.	q_5	⋊:0	⋊⟨H̲⟩⋉	V:-1	⋊V̲V⟨V̲⟩VVV⋉	V	VVV́V
8.	q_3	⋊:0	⋊⟨H̲⟩⋉	V:-1	⋊VV⟨V̲⟩VVV⋉	V	VVVV́V
9.	q_f	⋊:-1	⋊⟨H̲⟩⋉	⋊:-1	⋊VV⟨V̲⟩VVV⋉	λ	VVVV́V

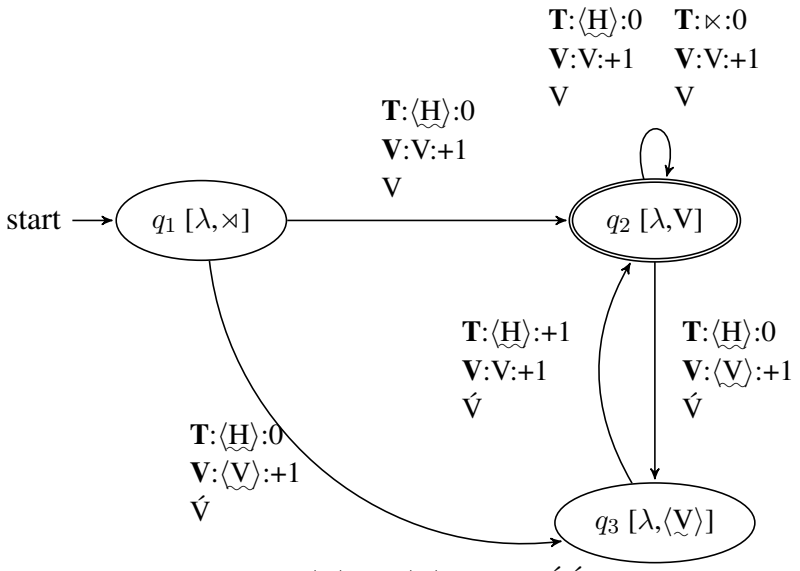
[26]Note that this MISL MT-FST cannot guarantee that there exists a preassociated vowel ⟨V̲⟩ for every preassociated tone ⟨H⟩. It would map /⟨H̲⟩ + VVVV/ to VVV́V. This is because the distance between the preassociated vowel and the penultimate vowel is unbounded.

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: /⟨H⟩ + V⟨V⟩VV/ maps to VV́V́V. This function is modeled by the [1,2]-MISL MT-FST in Figure 43a, with a sample derivation in Figure 43b. We assume that the input tone string contains either an empty string ⋊⋉ or a sequence of preassociated H tones ⋊⟨H⟩*⋉.

Figure 43: *Unbounded tone spread in Bemba*

(a) *MT-FST for Bemba*



(b) *Derivation of* /⟨H⟩ + V⟨V⟩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		⋊⟨H⟩⋉		⋊V⟨V⟩VV⋉		
2.	q_1	⋊:+1	⋊⟨H⟩⋉	⋊:+1	⋊V⟨V⟩VV⋉	λ	
3.	q_2	⟨H⟩:0	⋊⟨H⟩⋉	V:+1	⋊V⟨V⟩VV⋉	V	V
4.	q_3	⟨H⟩:0	⋊⟨H⟩⋉	⟨V⟩:+1	⋊V⟨V⟩VV⋉	V	VV́
5.	q_2	⟨H⟩:+1	⋊⟨H⟩⋉	V:+1	⋊V⟨V⟩VV⋉	V́	VV́V́
6.	q_2	⋉:0	⋊⟨H⟩⋉	V:+1	⋊V⟨V⟩VV⋉	V	VV́V́V
7.	q_f	⋉:+1	⋊⟨H⟩⋉	⋉:+1	⋊V⟨V⟩VV⋉	λ	VV́V́V

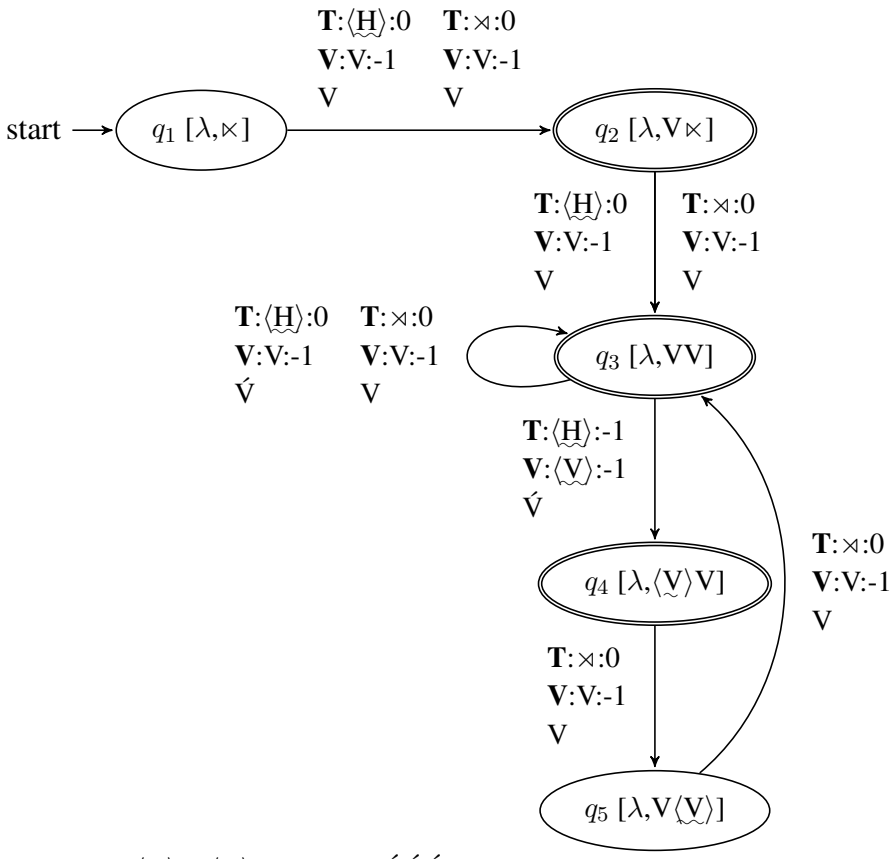
A.2.6 Ndebele unbouned spreading

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

44a, with a sample derivation in Figure 44b. We assume that the input tone string contains either an empty string $\rtimes\ltimes$ or a single preassociated H tone $\rtimes\langle\underline{\text{H}}\rangle\ltimes$. 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 44: *Unbounded spreading in Ndebele*

(a) *MT-FST for Ndebele*



(b) *Derivation of* /⟨H⟩ + ⟨V⟩VVVV/→[V́V́V́VV] *in Ndebele with the MT-FST in* Figure 44a

	Current State	**T**-tape Arc	**T**-tape Next	**V**-tape Arc	**V**-tape Next	Output symbol	Output String
1.	q_0		⋊⟨H⟩⋉		⋊⟨V⟩VVVV⋉		
2.	q_1	⋉:-1	⋊⟨H⟩⋉	⋉:-1	⋊⟨V⟩VVVV⋉	λ	
3.	q_2	⟨H⟩:0	⋊⟨H⟩⋉	V:-1	⋊⟨V⟩VVVV⋉	V	V
4.	q_3	⟨H⟩:0	⋊⟨H⟩⋉	V:-1	⋊⟨V⟩VVVV⋉	V	VV
5.	q_3	⟨H⟩:0	⋊⟨H⟩⋉	V:-1	⋊⟨V⟩VVVV⋉	V́	V́VV
6.	q_3	⟨H⟩:0	⋊⟨H⟩⋉	V:-1	⋊⟨V⟩VVVV⋉	V́	V́V́VV
7.	q_4	⟨H⟩:-1	⋊⟨H⟩⋉	⟨V⟩:-1	⋊⟨V⟩VVVV⋉	V́	V́V́V́VV
8.	q_f	⋊:-1	⋊⟨H⟩⋉	⋊:-1	⋊⟨V⟩VVVV⋉	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 30, 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 45 using the [1,1,1]-MISL MT-FST from Figure 27.

Figure 45: *Derivation of* turʒim *using the MT-FST in* Figure 27

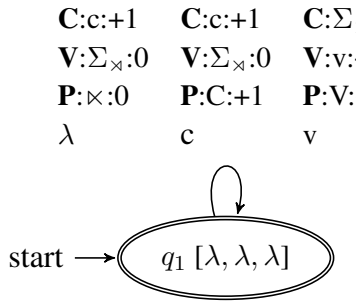
	Current State	**C**-tape Arc	**C**-tape Next	**V**-tape Arc	**V**-tape Next	**P**-tape Arc	**P**-tape Next	Output Symbol	Output String
1.	q_0		⋊trʒm⋉		⋊ui⋉		⋊CVCCVC⋉		
2.	q_1	⋊:+1	⋊trʒm⋉	⋊:+1	⋊ui⋉	⋊:+1	⋊CVCCVC⋉	λ	
3.	q_1	t:+1	⋊trʒm⋉	u:0	⋊ui⋉	C:+1	⋊CVCCVC⋉	t	t
4.	q_1	r:0	⋊trʒm⋉	u:+1	⋊ui⋉	V:+1	⋊CVCCVC⋉	u	tu
5.	q_1	r:+1	⋊trʒm⋉	i:0	⋊ui⋉	C:+1	⋊CVCCVC⋉	r	tur
6.	q_1	ʒ:+1	⋊trʒm⋉	i:0	⋊ui⋉	C:+1	⋊CVCCVC⋉	ʒ	turʒ
7.	q_1	m:0	⋊trʒm⋉	i:+1	⋊ui⋉	V:+1	⋊CVCCVC⋉	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 31, 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 46a, illustrated with the derivation in Figure 46b.

Figure 46: *1-to-1 slot-filling in 5-consonant roots*

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



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

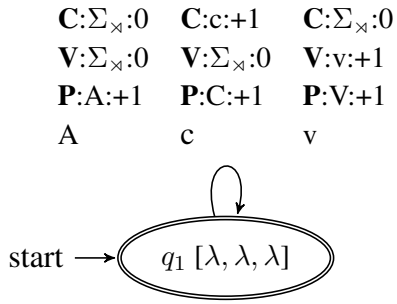
	Current State	**C**-tape Arc	**C**-tape Next	**V**-tape Arc	**V**-tape Next	**P**-tape Arc	**P**-tape Next	Output Symbol	Output String
1.	q_0		⋊mɣnṭs⋉		⋊ui⋉		⋊CVCCVC⋉		
2.	q_1	⋊:+1	⋊mɣnṭs⋉	⋊:+1	⋊ui⋉	⋊:+1	⋊CVCCVC⋉	λ	
3.	q_1	m:+1	⋊mɣnṭs⋉	u:0	⋊ui⋉	C:+1	⋊CVCCVC⋉	m	m
4.	q_1	ɣ:0	⋊mɣnṭs⋉	u:+1	⋊ui⋉	V:+1	⋊CVCCVC⋉	u	mu
5.	q_1	ɣ:+1	⋊mɣnṭs⋉	i:0	⋊ui⋉	C:+1	⋊CVCCVC⋉	ɣ	muɣ
6.	q_1	n:+1	⋊mɣnṭs⋉	i:0	⋊ui⋉	C:+1	⋊CVCCVC⋉	n	muɣn
7.	q_1	ṭ:0	⋊mɣnṭs⋉	i:+1	⋊ui⋉	V:+1	⋊CVCCVC⋉	i	muɣni
8.	q_1	ṭ:+1	⋊mɣnṭs⋉	⋉:0	⋊ui⋉	C:+1	⋊CVCCVC⋉	t	muɣnit
9.	q_1	s:+1	⋊mɣnṭs⋉	⋉:0	⋊ui⋉	⋉:0	⋊CVCCVC⋉	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 32, the template **P**=*CtVCVC* has a preassociated affix ⟨t⟩. With a root **C**=*ksb* and vocalism **V**=*ui*, the output is *ktusib*. A [1,1,1]-MISL MT-FST is provided in Figure 47a along with a sample derivation in Figure 47b. The symbol *A* represents any input symbol from the input alphabet of segments {t,n,m} which are possible segmental affixes in McCarthy (1981a).

Figure 47: *1-to-1 slot-filling with pre-associated affixes*

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



(b) *Derivation of* k⟨t⟩usib

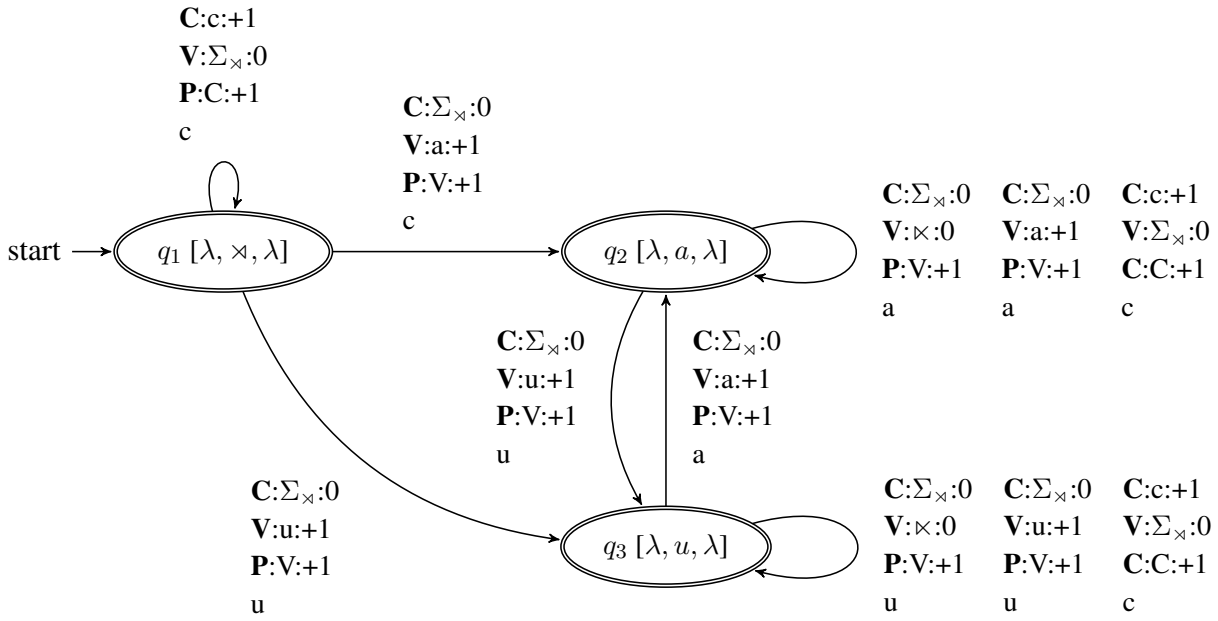
	Current State	**C**-tape Arc	**C**-tape Next	**V**-tape Arc	**V**-tape Next	**P**-tape Arc	**P**-tape Next	Output Symbol	Output String
1.	q_0		⋊ksb⋉		⋊ui⋉		⋊CtVCVC⋉		
2.	q_1	⋊:+1	⋊ksb⋉	⋊:+1	⋊ui⋉	⋊:+1	⋊CtVCVC⋉	λ	
3.	q_1	k:+1	⋊ksb⋉	u:0	⋊ui⋉	C:+1	⋊CtVCVC⋉	k	k
4.	q_1	s:0	⋊ksb⋉	u:0	⋊ui⋉	t:+1	⋊CtVCVC⋉	t	kt
5.	q_1	s:0	⋊ksb⋉	u:+1	⋊ui⋉	V:+1	⋊CtVCVC⋉	u	ktu
6.	q_1	s:+1	⋊ksb⋉	i:0	⋊ui⋉	C:+1	⋊CtVCVC⋉	s	ktus
7.	q_1	b:0	⋊ksb⋉	i:+1	⋊ui⋉	V:+1	⋊CtVCVC⋉	i	ktusi
8.	q_1	b:+1	⋊ksb⋉	⋉:0	⋊ui⋉	C:+1	⋊CtVCVC⋉	b	ktusib
9.	q_1	⋉:+1	⋊ksb⋉	⋉:+1	⋊ui⋉	⋉:+1	⋊CtVCVC⋉	λ	ktusib

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

In Figure 33a, 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 48a, illustrated with a sample derivation in Figure 48b. 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 §7.3.2.

Figure 48: *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 48a

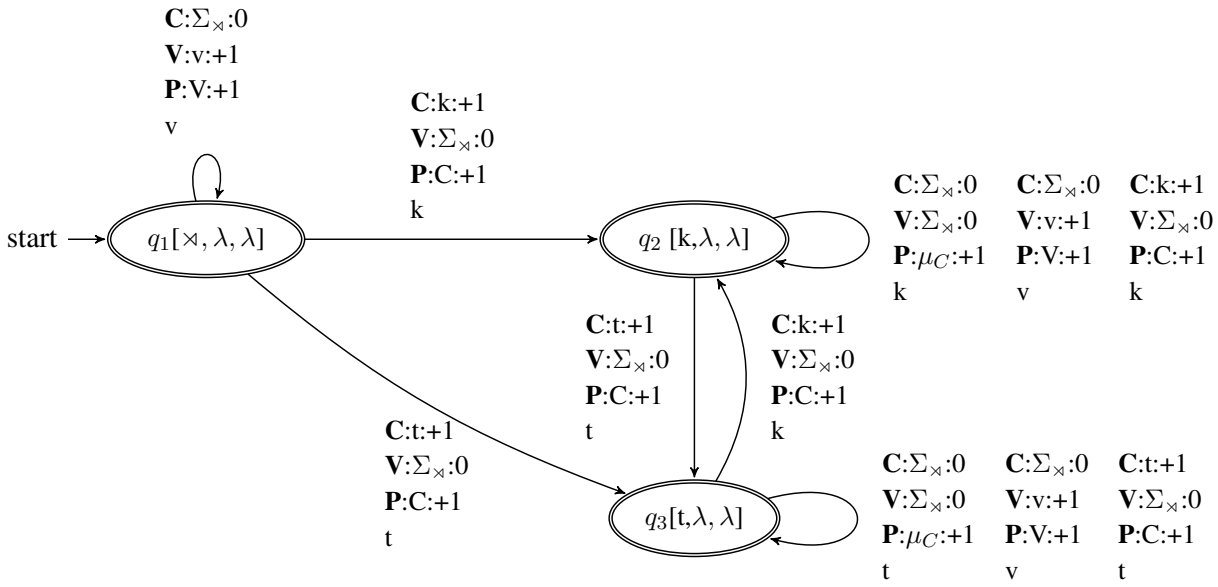
	Current State	**C**-tape Arc	**C**-tape Next	**V**-tape Arc	**V**-tape Next	**P**-tape Arc	**P**-tape Next	Output Symbol	Output String
1.	q_0		⋊ktb⋉		⋊a⋉		⋊CVCVC⋉		
2.	q_1	⋊:+1	⋊k̲tb⋉	⋊:+1	⋊a̲⋉	⋊:+1	⋊C̲VCVC⋉	λ	
3.	q_1	k:+1	⋊kt̲b⋉	a:0	⋊a̲⋉	C:+1	⋊CV̲CVC⋉	k	k
4.	q_2	t:0	⋊kt̲b⋉	a:+1	⋊a⋉̲	V:+1	⋊CVC̲VC⋉	a	ka
5.	q_2	t:+1	⋊ktb̲⋉	⋉:0	⋊a⋉̲	t:+1	⋊CVCV̲C⋉	t	kat
6.	q_2	b:0	⋊ktb̲⋉	⋉:0	⋊a⋉̲	V:+1	⋊CVCVC̲⋉	a	kata
7.	q_2	b:+1	⋊ktb⋉̲	⋉:0	⋊a⋉̲	C:+1	⋊CVCVC⋉̲	b	katab
8.	q_f	⋉:+1	⋊ktb⋉	⋉:+1	⋊a⋉	⋉:+1	⋊CVCVC⋉	λ	katab

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

In Figure 35b, the template **P**=*CVC*μ_C*VC* 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 49a, with a sample derivation in Figure 49b for a nonce word *kuttik* with root **C**=*ktk*. For illustrative reasons, the consonant alphabet is only {k,t}.

Figure 49: *1-to-1 slot-filling with final spread of vowels*

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



(b) *Derivation of* kuttik *using the MT-FST in* Figure 49a

	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