Finite-state locality in Semitic root-and-pattern morphology

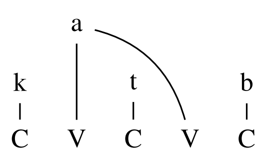
Hossep Dolatian and Jonathan Rawski[*]

1 Introduction

A common goal in mathematical phonology and morphology is determining necessary and sufficient conditions regarding the computational power needed for a given linguistic process. Cross-linguistically, most morphological processes are local across different domains (Embick 2010, Marantz 2013) and even in Semitic (Arad 2003, Kastner 2016), e.g. concatenative morphology or suffixation (Chandlee 2017). However, Semitic languages use non-concatenative templatic morphology or *root-and-pattern morphology* (RPM). Semitic RPM has superficially unclear generative power (McCarthy 1981, 1993, Hudson 1986, Hoberman 1988, Yip 1988, McCarthy and Prince 1990a,b). All our examples are from Modern Standard Arabic.

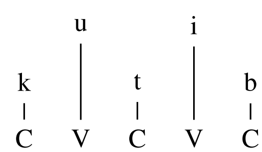
To illustrate, an Arabic active verb like *katab* 'it wrote' consists of three morphological items: the root consonants *ktb*, the inflectional vowels *a*, and a prosodic template *CVCVC*. The three items are inter-digitated to form *katab*. Contrast this with a passive verb *kutib* 'it was written', which has the same consonants *ktb* and template *CVCVC*, but different vowels *ui* marking the passive voice. Their decomposition is shown in (1) as autosegmental graphs. A small paradigm for Arabic templatic morphology is in Table (1).

1. (a) *katab* 'it wrote'



2. (a) *kutib* 'it was written'

(b)



katab	'he wrote'
kutib	'it was written'
kattab	'he dictated
ta-kattab	'it was dictated
kaatab	'he corresponded
sta-ktab	'he subscribed'

Table 1: List of words derived from the root *ktb*.

In computational morphology, there is more work on concatenative morphology than non-concatenative morphology (Sproat 1992, Roark and Sproat 2007). This paper focuses on the non-concatenative process of template-filling. We show that template-filling *is* a local process over *multiple* input items. By being *Multi-Input Strictly Local* (MISL), Semitic RPM can be effectively computed by a corresponding class of *multi-tape* finite-state transducers (Kay 1987, Kiraz 2001). At face value, the claim that Semitic RPM can be modeled by multi-tape transduces is not novel. Our contribution is to show that full finite-state power is sufficient, but not necessary, and that RPM is restricted computationally in a way that matches the generalizations of other morpho-phonological processes.

We first provide background knowledge on Semitic computational morphology in section 2. We discuss locality in section 3. In section 4, we define and illustrate MT-FSTs and discuss their history in Semitic morphology. We show that template-filling is a *local* process over multiple tapes.

[*]Our gratitude to Jeffrey Heinz, Mark Aronoff, Sedigheh Moradi, and audiences at PhoNE, PLC, ASAL, and AIMM.

Conceptual issues are discussed in section 5. Crucially, we show that our computational result holds even if the template is treated as phonologically derived instead of a morphological input; and, it holds because we disentangle the process from the fact that Semitic templates have a bounded finite size.

2 History of Computing Templatic Morphology

For templatic morphology, 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).

In terms of computational models and properties for templatic morphology, there has been less work. For *concatenative* morphology, the most common computational model is single-tape finite-state transducers (1T-FSTs) which have one input tape and one output tape (Roark and Sproat 2007). 1T-FSTs are not ideally designed for non-linear processes like templatic morphology. Implementations of template-filling with 1T-FSTs suffer from state explosion because they list all existing words in the language as a finite list.

Modifications have been developed to curb state-explosion problem with 1T-FSTs. These modifications push the burden of computation onto alternative representations for FSAs (Bird and Ellison 1994), onto run-time procedures (Beesley and Karttunen 2003), or onto trading off space complexity with time complexity (Cohen-Sygal and Wintner 2006). For a review, see Kiraz (2000:92), Kiraz (2001:Ch4), and Wintner (2014:47). But these solutions do not address the computational properties of template-filling, specifically its locality.

One alternative computational model to 1T-FSTs is *multi-tape* finite-state machines (MT-FSM) (Furia 2012). MT-FSMs have a long history of use of for Semitic template-filling, whether as non-deterministic transducers (Kay 1987), deterministic transducers (Wiebe 1992), or as synchronous acceptors (Kiraz 2000, 2001, Hulden 2009). Synchronous MT-FSAs suffer from a similar state-explosion as 1T-FSTs (Hulden 2009). Synchronous MT-FSAs are equivalent to 1T-FSAs, i.e. regular languages (Wiebe 1992). Asynchronous MT-FSTs are more powerful and are the focus of our work.

Most implementations of Semitic morphology avoid this problem by not *directly* modeling templatic morphology. As a function, template-filling takes as input a root *ktb*, a vocalism *ui*, and an *unfilled* template *CVCVC*. Its output is the *filled* template *kutib*. Many computational implementations avoid implementing this function by instead listing all existing *filled* templates. The implementation is then just a finite but large lexicon. The listing approach works in practice because roots unpredictably combine with different templates.

However, listing is not a linguistically, mathematically, or cognitively useful model. The bulk of theoretical and psycholinguistic results show that template-filling is a real process (Prunet 2006, Aronoff 2013, Kastner 2016). By reducing template-filling into a list of filled templates, generalizations on template-filling and its computational properties are lost. One generalization we focus on is the *computational locality* of template-filling.

3 Computional locality over single inputs: Input Strictly Local

In mathematical phonology, there has been work on mapping different phonological processes to different subclasses of regular languages and functions (McNaughton and Papert 1971, Heinz and Lai 2013, Chandlee 2014, Heinz 2011a,b, 2018, Jardine 2016a,b). A partial hierarchy for the relevant functions is shown in Figure 1. They allow a precise characterization of attested and unattested morpho-phonological processes. Note that rational functions are computed by 1-way finite-state transducers (1-way FSTs) which process the input string once in one direction. They are a subclass of regular functions which are computed by 2-way FSTS which process the input string multiple times by going back and forth on the string (Filiot and Reynier 2016, Dolatian and Heinz 2018).

One of the simplest subclasses is the class of Input Strictly Local (ISL) functions. They determine an output string for a given input string based only on contiguous substrings of bounded length

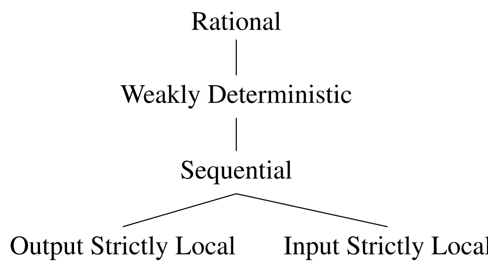


Figure 1: Hierarchy of subclasses for rational functions

(Chandlee 2014, Chandlee et al. 2014). Despite their reduced expressivity, they capture a striking majority of local segmental phonological and morphological maps (Chandlee 2017) and have been argued to provide a precise, well-defined notion of linguistic locality (Chandlee and Heinz 2018, Chandlee et al. 2018).

Cross-linguistically, most morphological processes are local because they are concatenative. They consist of adding a string of segments at one end of the input. For example, the FST in Figure 2 computes progressive suffixation in English: *hold*→*hold-ing*. It models a 1-ISL function where $k = 1$ because only the *current* input symbol is needed in order to know if the suffix should be outputted or not, i.e. if the read head is on the end boundary $\ltimes$. Except for the initial and final state, each state keeps track of the last $k-1$ input symbols before the current input symbol;, here it is the empty string ε of size 0.

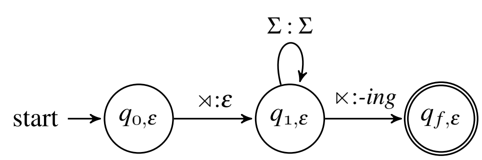


Figure 2: 1-ISL 1T-FST for English suffixation

There is little discussion on the locality or non-locality of Semitic template-filling. Chandlee (2017) shows that template-filling cannot be easily modeled with single-tape FSTs. Superficially, the fact that templatic morphology involves dis-contiguous units (interdigitation) implies that it is a non-local process. In contrast, this paper shows that template-filling *is* a local process, using an appropriate computational representation. In what follows, we use Multi-Tape FSTs as an alternative computational model for template-filling. They are an early and intuitive model for Semitic morphology (Kay 1987, Kiraz 2001).

4 Multi-Input Strict Locality in Semitic RPM

In order to compute Semitic RPM, we generalize from the notion of functions and ISL which work over a *single* input to *multi*-input functions and *multi*-ISL which work over multiple input items. The class of *Multi-Input Strictly Local* functions (MISL) is computed by a specific subclass of *multi-tape* finite-state transducers: determinstic asynchronous MISL MT-FSTs. We explain by illustration. All concepts are kept at a high enough level in order to facilitate comprehension without getting into technical details.

Figure 3 is an MT-FST for *simple* Semitic template-filling. It has 3 input tapes but 1 output tape. The three input tapes are morphologically defined: the root consonant tape **C**, the inflectional vocalism tape **V**, and the prosodic template tape **T**.

The input corresponds to a tuple of 3 strings: the consonant tape **C**, the vowel tape **V**, and the template tape **T**. The alphabets for the consonant tape **C**, vowel tape **V**, and template **T** are Σ_1 (= the set of all possible consonants), Σ_2 (= the set of all possible vowels), and Σ_3 (= the set of

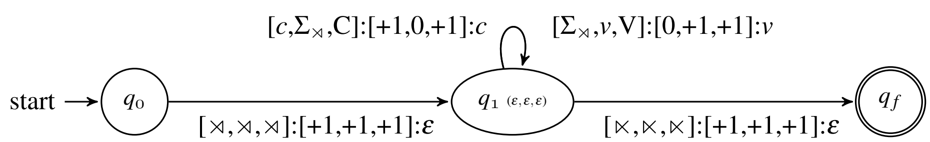


Figure 3: MT-FST for 1-to-1 slot-filling.

prosodic slots $\{C,V\}$). Each tape alphabet includes the start boundary ⋊ and the end boundary ⋉: $\Sigma_{1⋊} = \Sigma_1 \cup \{⋊,⋉\}$, $\Sigma_{2⋊} = \Sigma_2 \cup \{⋊,⋉\}$, $\Sigma_{3⋊} = \Sigma_3 \cup \{⋊,⋉\}$. The output alphabet is the output segments. Richer template structures, e.g. templates with moras, require larger template alphabets (Kay 1987). Readers are encouraged to read our other progress reports on this topic.

If an MT-FST has 3 input tapes, then it has three read heads with one on each tape. Transition arcs are drawn between an input state *q* and an output state *p*. Each transition arc is made up of 3 chunks delimited by ':'. These 'chunks' tell the MT-FST 1) what to read on the 3 input tapes, 2) where to move on the three inputs, and 3) what to output on the output tape. Movement on the tapes is represented by the direction parameters +1 for advancing left-to-right and 0 for staying put.[1] The choice of the output symbol depends on what is read on *each* input tape, not just the **T** tape.

Expanding on the role of the transition arcs, Transition arcs in the form of:

[input symbol on **C** tape,	input symbol on **V** tape,	input symbol on **T** tape]:
[direction on **C** tape,	direction on **V** tape,	direction on **T** tape]:
output string		

The transition arcs in the MT-FST in Figure 3 are in shorthand. A transition arc like [Σ⋊,*v*,V]: [0,+1,+1]:*v* is interpreted as follows. Lower case letters are interpreted as variables. If the template tape **T** reads a vocalic slot *V*, while the vowel tape **V** reads some symbol *v*, then the output of this transition is that symbol *v*.

The machine in Figure 3 can model any 1-to-1 match between a prosodic template **T** and the consonants **C** and vowels **V**. A mapping is 1-1 if a) the number of consonant slots in **T** matches the number of consonants in **C**, b) the number of vowel slots in **T** matches the number of vowels in **V**. To illustrate, given **T**=*CVCVC*, **C**=*ktb*, and **V**=*ui*, template-filling is 1-1 because a) **T**=*CVCVC* consists of 3 consonant slots for the 3 consonants in **C**=*ktb*, and b) **T**=*CVCVC* consists of 2 vowel slots for the two vowels in **V**=*ui*.

We illustrate a derivation for *kutib* with the MT-FST in Table 2. Each row keeps track of the:

1. current state
2. location of the read heads on the 3 input tapes
3. transition arc used on each 3 input tapes
4. outputted symbol
5. current output string

Although the MT-FST seems powerful, it actually displays locality over multiple inputs: it is MISL. The MT-FST computes a (1,1,1)-MISL function which uses a bound locality window of size 1 on each of the input tapes. All the work is done by state q_1 which kept track of the last 0 segments on the 3 input tapes. When deciding on what to output and which state to go to, only the current input symbols on the 3 tapes were needed.

To illustrate 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 1T-FSTs. 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 **T** tapes.

Many other types of attested template-filling processes are likewise local. A partial list is in Table 3. The different patterns involve different types of matching: 1-to-many mappings, medial vs.

[1]The MT-FSTs which we use are 1-way and not 2-way. They cannot retract on the input tapes. 2-way MT-FSTs are strictly more powerful than 1-way MT-FSTs (Furia 2012).

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

Table 2: Derivation of *kutib* using the MT-FST in Figure 3

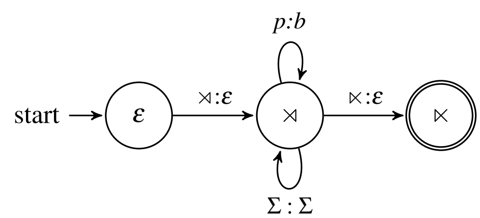


Figure 4: 1-ISL FST for absolute neutralization

final spread, pre-associated slots, autosegments, affixes, reduplication, and edge-in effects. Locality in these patterns depends on the computational representation and derivation. A full discussion of other template patterns (McCarthy 1981) is found in our other progress reports .[2]

Matching	Input			Output	
1-1 Matching	*ktb*	*ui*	*CVCVC*	*ku.tib*	1-MISL
Final spread	*ktb*	***a***	*CVCVC*	*k**a**.t**a**b*	2-MISL
Gemination	*ktb*	*ui*	*CVC.µVC*	*ka**t**.**t**ab*	2-MISL
Pre-association	*ksb*	*a*	*C**t**VCVC*	*k**t**a.sab*	1-MISL
C-spreading	*trʒm*	*ui*	*CVC.CVC*	*tar.ʒam*	3-MISL
	ktb	*ui*	*CVC.CVC*	*kut.tib*	3-MISL
Partial copying	*brd*	*a*	*CVC.CVC*	*bar.**b**ad*	2-way
Total copying	*zl*	*ia*	*CVC.CVC*	***zil.zal***	2-way
Edge-in	*ktb*	*ua**i***	*mV-tV-CVC.CVC*	*mu-ta-kat.t**i**b*	unclear

Table 3: Locality parameters for different Arabic templates

5 Conceptual Issues in multi-tape transducers for templates

In this section we go through theoretical issues on the role of templates in Semitic morphology and how they affect computation.

5.1 Phonological emergence of templates

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 **T**=*CVCVC*. Crucially, we included the template **T** as part of the input as a morphological

[2] We do not discuss template-fillling in broken plural formation (McCarthy and Prince 1990b). See Kiraz (2001) on how it could be formalized.

primitive.[3] This assumption was made in earlier work on Semitic morphology (McCarthy 1981). Alternative formulizations were later proposed, such as that the template is made of prosodic units like moras, syllables, and feet (McCarthy and Prince 1990a,b) or is derived from other templates via affixation (McCarthy 1993).[4]

However, recent work on Semitic argues that there is no pre-specified template input (Tucker 2010, Ussishkin 2011, Bat-El 2011, Kastner 2016, Zukoff 2017). In constraint-based theories, the only inputs are root consonants **C**, vowels **V**, and a set of phonological constraints CON.[5] The prosodic organization of these morphological items emerges from the phonology via optimizing phonological constraints on syllable structure, autosegmental docking, and word-size requirements.

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. Even though the template is emergent, there must be still be a mechanism which will interdigitate the root and vowels into this emergent template.

To illustrate, consider a toy Optimality-Theoretic derivation for the word *kutib* using no templates, but only the root consonants **C**, the vowels **V**, and the phonological constraints CON. The toy constraints in Figure 6 illustrate the basic idea: constraints on syllable structure will choose the optimal candidate *kutib* for the input *ktb-ui*.

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

Figure 5: OT tableau for /ktb+ui/ without an input template

The phonological derivation in Figure 6 consists of two stages. The first stage, GEN, generates all possible permutations or organizations of consonants and vowels. The second stage, EVAL, evaluates the optimal permutation or organization based on ranked phonological constraints on syllable structure. Most theoretical work on modeling Semitic templates as emergent focus only on the phonological constraints CON and the evaluation step EVAL. In contrast, GEN is treated as a black-box with relatively little work on its computational modeling (Karttunen 1993).[6]

But the candidates in GEN *imply* a template, i.e. a specific manner of organizing the consonants and vowels. We explicitly show this organization in Figure 6. We hypothesize that the MT-FSTs model how GEN computes the phonologically emergent template. Thus, regardless if we consider templates as morphologically primitives or phonological emergent, the templates still need to be computed.

ktb + ui	*[CC	Onset	Contiguity
a. ☞ kutib CV.CVC			∗ ∗ ∗
b. ktbui CC.CVV	∗!		
c. uktib VC.CVC		∗!	

Figure 6: OT tableau with for /ktb+ui/ implicit templates

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

[4]Decomposing template filling into a set of affixation processes is simply the composition of multiple ISL and MISL processes. We do not discuss this here but see our other work.

[5]We can include certain derivational morphology such as infixes <*t*> or moraic autosegments μ.

[6]There is more work on modeling GEN in serialist versions of OT like *Harmonic Serialsm* (McCarthy 2010). See Hao (2017, to appear) for results on its expressive power.

5.2 Role of infinity vs. finiteness

The second 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).

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

As a function, template-filling takes as input a root *ktb*, a vocalism *ui*, and an *unfilled* template *CVCVC*. Its output is the *filled* template *kutib*. in section 4, we showed that template-filling can be modeled by MISL MT-FST with a small window of locality, *k*=1 for 1-1 matching. The MT-FST did not need to memorize all possible shapes for roots, vocalisms, and templates. It would work for inputs of any size.

For example, given the *hypothetical* root consonants **C**=*ktbm*, vocalism **V**=*uaui*, and 4-syllable template **T**=*CVCVCVCV*, the MT-FST from Figure 3 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 model this bound on verb size.

Because of how filled templates in Arabic are at most 2 syllables, with additional slots of affixes, an alternative computational implementation is a single-taped FST over *finite* languages. The 1T-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 1T-FST would be ISL with a large locality window of at least size 9.

In fact, most computational implementations avoid *directly* implementing the template-filling function by instead listing all existing *filled* templates. The implementation is then just a finite but large lexicon. The listing approach works in practice because roots unpredictably combine with different templates (= the choice is lexicalized).

However, the 1T-FST approach is problematic in terms of implementation, cognition, and computation. In terms of implementation, there is a trade-off between the state explosion in 1T-FSTs. vs. using richer computational structure in MT-FSTs. And in terms of cognition, listing is not a useful model. The bulk of theoretical and psycholinguistic results show that template-filling is a real process (Prunet 2006, Aronoff 2013, Kastner 2016).

As for computation, the 1T-FST reduces Semitic morphology into a large but finite set of words. By being finite, it cannot generalize to novel types of roots, vocalisms, or templates. 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 exists because a filled template has at most 8 segments. This finiteness is an *independent* generalization.

In sum, the fact that Semitic templates have a maximum size is independent from the process of template-filling. Encoding this finiteness directly into an MT-FST creates a finite language. This causes the loss of generalization. However, an 1T-FST needs to directly encode this finiteness otherwise it cannot do template-filling at all. An MT-FST does not have this setback.

6 Conclusion

This paper overviewed the computational nature of Semitic template filling, and found it to be restricted in an enlightening way. While the usual one-tape finite-state analyses of morphology showed an unsatisfying list view of template filling, we showed that incorporating the insight of multiple template parts produced an enlightening analysis. Furthermore, we showed that the functions over these multi-input structures are strictly local. This generalizes insights that have emerged from studying concatenative morphology, connecting non-concatenative processes to the computational landscape of subregularity. This provides necessary and sufficient conditions on the information a speaker must be sensitive to in order to generalize such processes, and it provides sets of non-trivial adequacy conditions on what any linguistic theory that attempts to model Semitic template-filling must incorporate.

References

Arad, Maya. 2003. Locality constraints on the interpretation of roots: The case of hebrew denominal verbs. *Natural Language & Linguistic Theory* 21:737–778.

Aronoff, Mark. 1994. *Morphology by itself: Stems and inflectional classes*. 22. MIT press.

Aronoff, Mark. 2013. The roots of language. In *The boundaries of pure morphology*, ed. Silvio Cruschina, Martin Maiden, , and John Charles Smith, 161–180. Oxford: Oxford University Press.

Attia, Mohammed, Pavel Pecina, Antonio Toral, Lamia Tounsi, and Josef van Genabith. 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*, 125–133. Association for Computational Linguistics.

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

Beesley, Kenneth, and Lauri Karttunen. 2003. *Finite-state morphology: Xerox tools and techniques*. Stanford, CA: CSLI Publications.

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

Chandlee, Jane. 2014. Strictly Local Phonological Processes. Doctoral dissertation, University of Delaware, Newark, DE.

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

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

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

Chandlee, Jane, Jeffrey Heinz, and Adam Jardine. 2018. Input strictly local opaque maps. *Phonology* 35:171–205.

Clark, Alexander. 2007. Supervised and unsupervised learning of arabic morphology. In *Arabic computational morphology: knowledge-based and empirical methods*, ed. Abdelhadi Soudi, Günter Neumann, and Antal Van den Bosch, 181–200. Springer.

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

Dawdy-Hesterberg, Lisa Garnand, and Janet Breckenridge Pierrehumbert. 2014. Learnability and generalisation of Arabic broken plural nouns. *Language, cognition and neuroscience* 29:1268–1282.

Daya, Ezra, Dan Roth, and Shuly Wintner. 2007. Learning to identify Semitic roots. In *Arabic computational morphology: knowledge-based and empirical methods*, ed. Abdelhadi Soudi, Günter Neumann, and Antal Van den Bosch, 143–158. Springer.

Dolatian, Hossep, and Jeffrey Heinz. 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.

Embick, David. 2010. *Localism versus globalism in morphology and phonology*, volume 60 of *Linguistic Inquiry Monographs*. Cambridge, MA: MIT Press.

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

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

Fullwood, Michelle, and Tim O'Donnell. 2013. Learning non-concatenative morphology. In *Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*, 21–27.

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

Hao, Yiding. 2017. Harmonic serialism and finite-state optimality theory. In *Proceedings of the 13th International Conference on Finite State Methods and Natural Language Processing (FSMNLP 2017)*, 20–29.

Hao, Yiding. To Appear. Finite-State Optimality Theory: Non-Rationality of Harmonic Serialism. *Journal of Language Modelling* 7:49–99.

Heinz, Jeffrey. 2011a. Computational phonology part I: Foundations. *Language and Linguistics Compass* 5:140–152.

Heinz, Jeffrey. 2011b. Computational phonology part II: Grammars, learning, and the future. *Language and Linguistics Compass* 5:153–168.

Heinz, Jeffrey. 2018. The computational nature of phonological generalizations. In *Phonological Typology*, ed. Larry Hyman and Frans Plank, Phonetics and Phonology, chapter 5, 126–195. De Gruyter Mouton.

Heinz, Jeffrey, and Regine Lai. 2013. Vowel harmony and subsequentiality. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, ed. Andras Kornai and Marco Kuhlmann, 52–63. Sofia, Bulgaria: Association for Computational Linguistics.

Hoberman, Robert D. 1988. Local and long-distance spreading in semitic morphology. *Natural Language & Linguistic Theory* 6:541–549.

Hudson, Grover. 1986. Arabic root and pattern morphology without tiers. *Journal of Linguistics* 22:85–122.

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

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

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

Karttunen, Lauri. 1993. Finite-state constraints. In *The last phonological rule: reflections on constraints and derivations*, ed. John Goldsmith, 173–194. University of Chicago Press.

Kastner, Itamar. 2016. Form and meaning in the Hebrew verb. Doctoral dissertation, New York University.

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

Kiraz, George Anton. 2000. Multitiered nonlinear morphology using multitape finite automata: a case study on syriac and arabic. *Computational Linguistics* 26:77–105.

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

Marantz, Alec. 2013. Locality domains for contextual allomorphy across the interfaces. In *Distributed Morphology Today*, ed. Ora Matushanksy and Alec Marantz, 95–115. MIT Press.

McCarthy, John, and Alan Prince. 1990a. Prosodic morphology and templatic morphology. In *Perspectives on Arabic linguistics II: papers from the second annual symposium on Arabic linguistics*, 1–54. John Benjamins Amsterdam.

McCarthy, John J. 1981. A prosodic theory of nonconcatenative morphology. *Linguistic inquiry* 12:373–418.

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

McCarthy, John J. 2010. An introduction to harmonic serialism. *Language and Linguistics Compass* 4:1001–1018.

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

McNaughton, Robert, and Seymour A Papert. 1971. *Counter-Free Automata*. The MIT Press.

van Oostendorp, Marc, Colin Ewen, Elizabeth Hume, and Keren Rice, ed. 2011. *The Blackwell companion to phonology*. Malden, MA: Wiley-Blackwell.

Prunet, Jean-François. 2006. External evidence and the semitic root. *Morphology* 16:41.

Roark, Brian, and Richard Sproat. 2007. *Computational Approaches to Morphology and Syntax*. Oxford: Oxford University Press.

Savitch, Walter J. 1993. Why it might pay to assume that languages are infinite. *Annals of Mathematics and Artificial Intelligence* 8:17–25.

Soudi, Abdelhadi, Günter Neumann, and Antal Van den Bosch, ed. 2007. *Arabic computational morphology: knowledge-based and empirical methods*. Springer.

Sproat, Richard William. 1992. *Morphology and computation*. Cambridge:MA: MIT press.

Tucker, Matthew A. 2010. Roots and prosody: the iraqi arabic derivational verb. *Recherches linguistiques de Vincennes* 31–68.

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

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

Wintner, Shuly. 2014. Morphological processing of semitic languages. In *Natural language processing of Semitic languages*, ed. Imed Zitouni, 43–66. Springer.

Yip, Moira. 1988. Template morphology and the direction of association. *Natural Language & Linguistic Theory* 6:551–577.

Zukoff, Sam. 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, 295—-314. Amherst, MA: Graduate Linguistics Student Association.

Department of Linguistics
Institute for Advanced Computational Science
Stony Brook University
Stony Brook, NY 11794
hossep.dolatian@stonybrook.edu
jonathan.rawski@stonybrook.edu