A Deterministic, Local Hypothesis for Tonal Processes*

Tajudeen Mamadou & Adam Jardine

Rutgers University

1. Introduction

In this paper, we investigate the question: What kind of functions are tone processes? Segmental phonology has been characterized as being *subsequential*, meaning that segmental processes can be described with deterministic finite-state transducers (Mohri 1997, Heinz and Lai 2013, Heinz 2018). In fact, segmental phonology has been shown to be overwhelmingly *local*, meaning output strings are determined based solely on contiguous substrings of bounded length *k* either in the input (Input Strictly Local) or in the output (Ouput Strictly Local) (Chandlee 2014, Chandlee et al. 2014, 2015).

However, Jardine (2016a) establishes that tone exhibits many patterns that are more computationally complex than this bound, by giving a number of examples of *unbounded circumambient* (UC) patterns, in which triggers or blockers can be arbitrarily far away on either side of any target or span thereof. An example is unbounded tone plateauing (UTP) in Luganda (Hyman and Katamba 2010) where H(igh) tones on either side of an unbounded span of toneless tone-bearing units (TBUs) form a single H-toned plateau. As Jardine (2016a) discusses, UC processes like UTP are not only non-local but also non-subsequential.

A yet unanswered question, then, is: what is a computational characterization of tonal processes that is both restrictive but also sufficiently expressive to capture UC patterns? We answer this question by extending the *melody-local* phonotactic grammars of **?** to processes and show that most tonal processes are *input melody-local (IML)* functions. The IML class is restrictive in that its functions can be computed using deterministic finite-state automata, which can only make one decision with respect to the output at any step of the computation. Crucially, however, the IML computing finite state machines are distinct from those that can compute segmental processes in the sense that they are computed by simultaneously reading multiple tiers (also cf. Rawski and Dolatian (2020)). In other words, the key to understanding the computational nature of tonal processes is local, deterministic computations

*We would like to thank audiences at NECPhon 2020, the Rutgers' MathLing group, and NELS 2021 for their questions and comments that improved the quality of the present work. All errors are obviously ours.

over an autosegmental-style representation. Furthermore, a subsequential computation of UTP-type of complex tone processes strengthens the hypothesis that phonological patterns are subregular Heinz and Lai (2013). This also promises to make progress in understanding non-subsequential *segmental* processes that have been uncovered in subsequent work (e.g., in McCollum et al. 2020).

This paper is structured as follows: §2 presents the theoretical and empirical background of the paper; §3 introduces the Input Melody Local (IML) functions and §4 presents the Finite State Transducers used to compute the IML functions. §5 presents a brief empirical survey of IML functions, §6 discusses the findings and §7 concludes.

2. Background

2.1 Subsequentiality

The notion of subsequentiality (Schützenberger 1977, Frougny and Sakarovitch 1993, Mohri 1997) is a measure of computational complexity for which deterministic computation is a defining property. A computation is said to be deterministic when at any point in the process of reading an input, there is only one decision the computing machine can make with respect to the output.

As a hypothesis for phonology, subsequentiality can intuitively be thought of as restricting phonological processes to only have *bounded look-ahead*. As Heinz and Lai (2013) and Jardine (2016a) note, this makes it similar to (but slightly less restrictive) than Wilson (2003, 2006)'s generalization that spreading is *myopic*. The subsequential hypothesis for phonology is attractive because it excludes a number of pathologies from the predicted typology (Heinz 2018) and the subsequential class of functions has proven learnability properties (Oncina et al. 1993). In other words, determinism is a strong property for a theory of phonology.

However, as Jardine (2016a) shows, there are a number of tone processes that are non-subsequential, because they require information that can be unboundedly far away from the target in both directions. In other words, non-subsequential processes require an unbounded look-ahead (in both directions) to see whether or not a second trigger or blocker is present before the process applies. A prominent example is UTP, which is illustrated as it is found in Luganda in (1).

(1) UTP in Luganda (Hyman and Katamba 2010, Jardine 2020)

a.	/kitabo/	[kitabo]	'book'	LLL
b.	/mutéma/	[mutéma]	'chopper'	LHL
c.	/kisikí/	[kisikí]	'log'	LLH
d.	/mutéma+bisikí/	[mutémá+bísíkí]	'log choper'	LHHHHH
e.		*[mutéma+bisikí]	_	*LHLLLH

In Luganda, any span of unspecified TBUs in between two H tones become H. As Jardine (2016a) shows, this is non-deterministic as any unspecified TBU must know if there is a H tone *both* to the right *and* to the left—and furthermore, that these triggers can appear unboundedly far to the left or the right. Jardine (2016a) shows that there are a number of examples of non-subsequential UTP processes in tone. (Furthermore, McCollum et al. (2020) give examples of non-subsequential segmental processes.)

The goal of the present paper is to use representation to deterministically capture these processes that are non-deterministic over strings by introducing the notion of melody, an idea that leverages the autosegmental theoretic insights that non-local tone processes on the timing tier are local over the tonal tier (Goldsmith 1976, Odden 1994).

2.2 Melody Locality

A great number of segmental phonotactics are *local*, in a computational sense of only being evaluated within a fixed window (Heinz 2007, 2009, 2010). Formally, constraints that are local in this sense are referred to as *strictly local* (SL; McNaughton and Papert 1971, Rogers and Pullum 2011). This notion of locality has been extended, inspired by autosegmental notions, to a formal notion of a *tier* (TSL; Heinz et al. 2011), which has been successfully applied to representing and learning long-distance segmental phonotactics (Jardine 2016b, McMullin and Hansson 2019).

However, Jardine (2020) shows that this notion of a tier is insufficient for representing tonal phonotactics. Instead, he proposes *melody-local* (ML) grammars based on principles of autosegmental phonology (Goldsmith 1976). ML grammars enforce SL contraints both on the surface string and the *melody* of the string, where the melody is a tier-like structure that assumes the Obligatory Contour Principle (OCP) (Leben 1978, McCarthy 1986, Odden 1986, 1988).

Jardine (2020) derives the melody in ML grammars with a melody function as follows. The melody function (defined in (2)) recursively applies to each span of tonally-specified TBUs, and returns a single H or L until no span of H or L toned TBU is left. It then takes the empty string (i.e λ, a string of length 0) as its final input and outputs an empty string (nothing). This function can be thought of as enforcing the OCP, retaining only one tone in a sequence of adjacent identical tones. It is also close in spirit to Heinz et al. (2011)'s '*erasing*' function in that the melody function erases all but one in a sequence of adjacent like-tones on the melody tier.

(2) Melody Function (Adapted from Jardine (2020))

$$\mathtt{mel}(w) \stackrel{\text{def}}{=} \begin{cases} \lambda & \text{if } w = \lambda, \\ \mathtt{mel}(v)\sigma & \text{if } w = v\sigma^n,\ v \neq u\sigma \text{ for some } u \in \Sigma^* \end{cases}$$

The function in (2) reads as follows: if `mel( )` applies to a *string* w, when w (henceforth, timing tier string) is an empty string (i.e λ), the `mel( )` function outputs an empty string; but when w equals $v\sigma^n$, where v is a variable representing a substring of any sequence and combination of tone symbols (e.g: LHH), and σ^n is one with a uniform n sequence of Ls or Hs, the function outputs the σ^n part of the input as a single σ. Applying iteratively, the function breaks up v into another substring of the form $v\sigma^n$ and goes through the steps described above again until there is no substring left to break up into yet smaller substrings.

Crucially, σ^n has to contain all and only the like-symbols (either Hs or Ls but not both) in a given unbroken stretch. An example is shown in (3) below, with each bolded symbol representing σ, the output of the melody function applying to the σ^n substring. In the example, w = HHHLLLHHL:

(3) Melody Function Derivation

$$\begin{aligned}
\texttt{mel}(\text{HHHLLLHHL}) &= \texttt{mel}(\text{HHHLLLHH})\textbf{L} \\
&= \texttt{mel}(\text{HHHLLL})\textbf{H}\text{L} \\
&= \texttt{mel}(\text{HHH})\textbf{L}\text{HL} \\
&= \texttt{mel}(\lambda)\textbf{H}\text{LHL} \\
&= \textbf{HLHL}
\end{aligned}$$

As in autosegmental phonology, the tier output by the the melody function is represented separately from that of the timing tier. The timing tier strings *and* the newly derived melody tier strings are conjunctively monitored by ML grammars via markedness-like constraints that ban some substrings from the grammar. To capture UTP in Luganda, for example, one simply only needs to posit a constraint *HLH on the melody tier that forbids a sequence of L-toned TBUs in between two H TBUs (Jardine 2020).

We can now import this idea directly into the study of functions to posit a deterministic, *local* theory of tone processes based on melodies.

3. Input Melody Local Functions

To achieve the need to extend subsequentiality to non-subsequential processes, we introduce the *input melody local* (IML) functions, which use insights from Jardine's ML grammars, but differ from the latter in that IML are functions computed locally on the input, in parallel to the *input strictly local* (ISL) functions of Chandlee (2014), Chandlee and Heinz (2018). The locality aspect of these functions means that they can only see a finite number of symbols at any given time on both the melody and timing tiers. The significant difference between IML functions and ISL functions is that IML functions operate over *two* inputs: the timing tier string itself *and* the melody tier derived from the application of the melody tier to that string.

To give an example, UTP in Luganda can be thought of as a function that maps input strings and output strings of L- and H-toned TBUs such that any span of input Ls in between two Hs is converted to a span of Hs.[1]

(4) a. LHLLLL $\mapsto$ LHLLLL
b. LHLLLH $\mapsto$ LHHHHH

It is this single-string function that is non-subsequential, for the reasons stated above. However, we can equally view this function as operating over *two* input strings—the original

[1] For now, we abstract away from underspecification of TBUs for tone; we return to this in the discussion section.

input string *and* the melody tier derived by applying the melody function to that string. An example from UTP is shown in (5).

(5) Schematization of an IML representation of UTP

Input

`mel`(w)	L	H	L			H
w	L	H	L	L	L	H

Output

output	L	H	**H**	**H**	**H**	H

Note that no matter how big the number of Ls between the Hs on the timing tier gets, the number of characters in the melody will still have a fixed HLH sequence. As we show below, this guarantees that UTP can be computed locally over the melody.

We now give a more rigorous defintion of IML based on automata theory.

4. Deterministic Multi-tier Finite State Transducers

Subsequential functions are computed by deterministic finite state transducers (FSTs) (Schützenberger 1977). We define the IML functions in terms of deterministic *multi-tape* FSTs (DM-FSTs), an automaton which takes input strings and computes output strings (a transducer) but in which the machine has access to multiple input strings (called a *tape*), each read by an independent *read head*. As shown this below, these machines can describe strictly more functions than traditional FSTs, while maintaining the restrictive property of determinism. For a thorough discussion of local DM-FSTs and their application to various phonological domains, see Rawski and Dolatian (2020).

Formally, a two-tape FST is a tuple T=$\langle\Sigma,\Gamma,Q,q_0,q_F,\Delta,\omega\rangle$ where:

- Σ and Γ are the input and output alphabets, respectively
- Q is the finite set of states; $q_0 \in Q$ is the single initial state (i.e where the computation begins), and $q_F \subseteq Q$ is the set of final or *accepting* states (i.e where successful computations end);
- $\Delta \subseteq Q \times (\Sigma \cup \lambda) \times (\Sigma \cup \lambda) \times \Gamma^* \times Q$ is the finite set of transitions. We represent a transition as $q \to X|Y : Z \to r$, which means that when the transducer is in state q and the next input symbol is X on the melody tape and Y on the timing tape, the machine goes to state r and outputs Z. If X or Y is λ, this means that the read head on that tape does not move during that step.

In this paper, we assume that $\Sigma = \{\rtimes, \text{H}, \text{L}, \ltimes\}$ and $\Gamma = \{\text{H}, \text{L}\}$, where $\rtimes$ and $\ltimes$ are special word-beginning and word-end markers that we assume only occur in the beginning and the end of strings.

For IML functions, we need exactly two tapes. An example two-tape FST that computes UTP is is given in Fig. 1, and a derivation for the example input/output map in (5) is given in Table 1.

To ensure that a two-tape machine is *deterministic*, we add the following restriction:

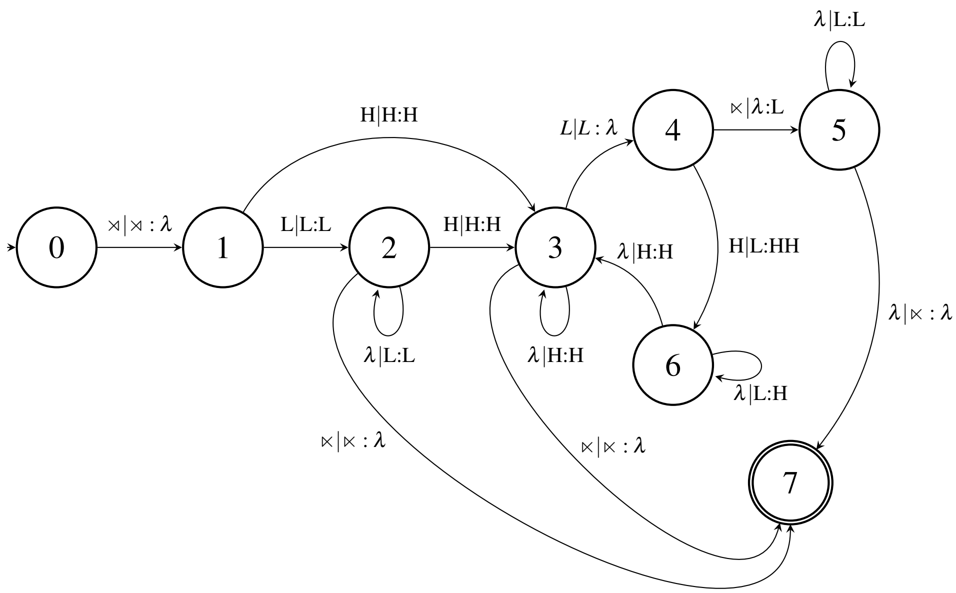

Figure 1: A 2-tape DM-FST for UTP.

(6) Properties of a two-tape DM-FST

a. For any transition on $X|Y$, at least X or Y is not λ; i.e., both input symbols cannot be empty strings).

b. For any two transitions on $X_1|Y_1$ and $X_2|Y_2$ out of the same state, it must be the case that either $X_1 \neq X_2 \neq \lambda$ or $Y_1 \neq Y_2 \neq \lambda$; i.e., transitions cannot differ only by replacing λ.

Restriction (6a) avoids non-determinism by disallowing moves in the machine without moves on the tape, and (6b) avoids nondeterminism introduced by two transitions of the form $\lambda|$H and H|H, for example.

Finally, we define a functional version of melody-locality by restricting ourselves to two-tape DMFSTs whose states keep track of local windows on the melody tier and timing tier.

(7) Definition of IML functions

A j,k-IML function f is a function from Σ^* to Γ^* such that $f(w)$ is computed by a two-tape DM-FST such that: 1) its input tapes are $\texttt{mel}(w)$ and w, respectively, for any string; and 2) each state of the machine corresponds to the previous $j-1$ symbols on the first tape and the previous $k-1$ on the second tape, respectively.

That is, an IML function is one that is computed deterministically over a string and its melody and where at any point in the computation, the output is determined entirely by the $j-1$ symbols in the input and the $k-1$ symbols in the input, respectively.

The DM-FST in Fig. 1 derives the UTP process. Consider the underlying sequence of TBUs LHLLLH, whose derived melody is LHLH. A full derivation of the process is shown in Table 1 below. Each row in that table represents a step in the derivation. Information to the left of the double line in the table indicates the current state of the machine at the given step; information to the right indicates the action specified in the machine given that state. Specifically, the cells in the Melody and Timing columns indicate the current position of the read heads in the input. The 'j-win.' column indicates the current window of j symbols on the melody tier ending on the current symbol in the read head, and the 'k-win.' indicates the parallel window of k symbols on the timing tier. Together, the q, $X|Y$ and r columns indicate the current state the machine is in, the transition it takes (i.e. what the machine reads on the Melody and Timing tier), and the state it moves to, respectively. Z column is the output of each step in the computation.

The Table in 1 reads as follows: as a first step in the computation, the machine starts out in state 0 where it reads the boundary symbols on both tapes (i.e. $\rtimes|\rtimes$), outputs nothing (i.e. λ) and moves to state 1. While in state 1, it reads Ls on both tapes, faithfully outputs L and moves to state 2. It then reads Hs on both tapes, outputs H and move to state 3; this state 3 means that the machine has seen the first trigger. Next, the machine moves again on both tapes, reading Ls and outputing λ, while moving to state 4. Outputting λ means the machine is waiting to see whether the second H trigger is coming down the line. For this purpose, the machine, now in state 4 reads H on the melody and L on the timing tapes. Now that the machine has seen the second H trigger, it knows that the L it previously saw on the timing tier as well as the current L have to be output as Hs, which is exactly what it does by outputting HH and moving to state 6. From this point until an H is encountered on the timing tier, every L the machine sees on the timing is output as H, which is why the machine's read-heads only oves on the timing tier and not on the melody, while looping twice in state 6. In step 7, the machine eventually encounters an H on the timing tier, it then outputs it as an H and transitions back to state 3. Upon reading the right boundary symbol on the timing tier, the machine then moves to state 7.

Now we consider the underlying tone sequence LHLLL, in which only one of the two required triggers is present in the input and as such, UTP will not apply. Note that the derived melody of this string is `mel`(LHLLL)=LHL. In such cases and as shown in Table 2, the machine proceeds in a similar way as in the derivation in Table 1. The key difference happens in step 5, where the machine reads the right boundary symbol on the melody tier after seeing an H followed by an L on that same melody tier. At this point, the machine knows there is no other H coming down the line and from that point on the machine only moves on the timing tier, outputting the Ls faithfully until it reads the right boundary symbol, which marks the end of the timing tier string and thus the end of the computation.

Note that the values for j and k are, in this machine, 3 and 1, respectively, and that the DM-FST meets the criteria for an IML function given above. Thus, UTP can be modeled with an IML function.

Step	q	Melody	Timing	j-win.	k-win.	$X \mid Y$	r	Z
1.	0	$\boxed{\rtimes}\text{LHLH}\ltimes$	$\boxed{\rtimes}\text{LHLLLH}\ltimes$		$\rtimes$	$\rtimes \mid \rtimes$	1	λ
2.	1	$\rtimes\boxed{\text{L}}\text{HL}\ltimes$	$\rtimes\boxed{\text{L}}\text{HLLLH}\ltimes$		L	$\text{L} \mid \text{L}$	2	L
3.	2	$\rtimes\text{L}\boxed{\text{H}}\text{LH}\ltimes$	$\rtimes\text{L}\boxed{\text{H}}\text{LLLH}\ltimes$	LH	H	$\text{H} \mid \text{H}$	3	H
4.	3	$\rtimes\text{LH}\boxed{\text{L}}\text{H}\ltimes$	$\rtimes\text{LH}\boxed{\text{L}}\text{LLH}\ltimes$	LHL	L	$\text{L} \mid \text{L}$	4	λ
5.	4	$\rtimes\text{LHL}\boxed{\text{H}}\ltimes$	$\rtimes\text{LHL}\boxed{\text{L}}\text{LH}\ltimes$	HLH	L	$\text{H} \mid \text{L}$	6	HH
6.	6	$\rtimes\text{LHL}\boxed{\text{H}}\ltimes$	$\rtimes\text{LHLL}\boxed{\text{L}}\text{H}\ltimes$	HLH	L	$\lambda \mid \text{L}$	6	H
7.	6	$\rtimes\text{LHL}\boxed{\text{H}}\ltimes$	$\rtimes\text{LHLLL}\boxed{\text{H}}\ltimes$	HLH	H	$\text{H} \mid \text{H}$	3	H
8.	3	$\rtimes\text{LHLH}\boxed{\ltimes}$	$\rtimes\text{LHLLLH}\boxed{\ltimes}$	$\text{LH}\ltimes$	$\ltimes$	$\ltimes \mid \ltimes$	7	λ
End	7							LHHHHH

Table 1: Derivation of $\langle$LHLH,LHLLLH$\rangle \rightarrow$ LHHHHH.

Step	q	Melody	Timing	j-win.	k-win.	$X \mid Y$	r	Z
1.	0	$\boxed{\rtimes}\text{LHL}\ltimes$	$\boxed{\rtimes}\text{LHLLL}\ltimes$		$\rtimes$	$\rtimes \mid \rtimes$	1	λ
2.	1	$\rtimes\boxed{\text{L}}\text{HL}\ltimes$	$\rtimes\boxed{\text{L}}\text{HLLL}\ltimes$		L	$\text{L} \mid \text{L}$	2	L
3.	2	$\rtimes\text{L}\boxed{\text{H}}\text{L}\ltimes$	$\rtimes\text{L}\boxed{\text{H}}\text{LLL}\ltimes$	LH	H	$\text{H} \mid \text{H}$	3	H
4.	3	$\rtimes\text{LH}\boxed{\text{L}}\ltimes$	$\rtimes\text{LH}\boxed{\text{L}}\text{LL}\ltimes$	LHL	L	$\text{L} \mid \text{L}$	4	λ
5.	4	$\rtimes\text{LHL}\boxed{\ltimes}$	$\rtimes\text{LH}\boxed{\text{L}}\text{LL}\ltimes$	LHL	L	$\ltimes \mid \text{L}$	5	L
6.	5	$\rtimes\text{LHL}\boxed{\ltimes}$	$\rtimes\text{LHL}\boxed{\text{L}}\text{L}\ltimes$	LHL	L	$\lambda \mid \text{L}$	5	L
7.	5	$\rtimes\text{LHL}\boxed{\ltimes}$	$\rtimes\text{LHLL}\boxed{\text{L}}\ltimes$	LHL	L	$\lambda \mid \text{L}$	5	L
8.	5	$\rtimes\text{LHL}\boxed{\ltimes}$	$\rtimes\text{LHLLL}\boxed{\ltimes}$	$\text{HL}\ltimes$	λ	$\lambda \mid \ltimes$	7	λ
End	7							LHLLL

Table 2: Derivation of $\langle$LHL, LHLLL$\rangle \rightarrow$ LHLLL.

5. Brief Empirical Survey of IML functions

IML functions can compute long distance processes like the UTP as well as local phonological processes. In this section, we show how IML captures local and long distance processes in tone, giving analyses of bounded tone shift in Rimi to represent the former and that of an unbounded tone spread in Ndebele to exemplify the latter. IML functions cover most tone processes, but for reasons of space, we limit ourselves to these two. Also, we do not give the full DM-FSTs for these processes and limit the exposition to the derivation tables that show the fragment of each DM-FST necessary for performing the computation.

5.1 Bounded Tone Shift in Rimi

In Rimi, H tone shifts one-step to the right (Meyers 1997) as shown in the examples in (8) below where the underlined vowel represents the position the tone shifted from. The Rimi function is shown in (9).

(8) Tone Shift in Rimi (Meyers 1997)

a. /rá-mu-ntu/ → [ra̱-mú-ntu] ‘of a person’
b. /u-púm-a/ → [u-pu̱m-á] ‘to go away’

(9) Tone Shift in Rimi

a. $f(\langle \mathrm{HL,HLLL} \rangle) = \mathrm{LHLL}$
b. $f(\langle \mathrm{LHL,LHL} \rangle) = \mathrm{LLH}$

The Rimi function is computed by a DM-FST with the properties in 6. A sample derivation is given in Table 3. The input HLLL is surrounded by domain delimiters (boundary symbols) as $\rtimes$HLLL$\ltimes$ for the timing tier. In a first step, $\rtimes$HLLL$\ltimes$ will be fed to the melody function, which will return $\rtimes$HL$\ltimes$. The derivation in Table 3 shows how the 2-tape DM-FST of Rimi takes $\langle \mathrm{HL}, \mathrm{HLLL} \rangle$ and outputs LHLL.

Step	q	Melody	Timing	j-win.	k-win.	$X\|Y$	r	Z
1.	0	$\boxed{\rtimes}\mathrm{HL}\ltimes$	$\boxed{\rtimes}\mathrm{HLLL}\ltimes$	$\rtimes$	$\rtimes$	$\rtimes\|\rtimes$	1	λ
2.	1	$\rtimes\boxed{\mathrm{H}}\mathrm{L}\ltimes$	$\rtimes\boxed{\mathrm{H}}\mathrm{LLL}\ltimes$	H	$\rtimes$H	$\lambda\|$H	3	λ
3.	3	$\rtimes\boxed{\mathrm{H}}\mathrm{L}\ltimes$	$\rtimes\mathrm{H}\boxed{\mathrm{L}}\mathrm{LL}\ltimes$	H	HL	$\lambda\|$L	4	LH
4.	4	$\rtimes\boxed{\mathrm{H}}\mathrm{L}\ltimes$	$\rtimes\mathrm{HL}\boxed{\mathrm{L}}\mathrm{L}\ltimes$	H	LL	$\lambda\|$L	4	L
5.	4	$\rtimes\boxed{\mathrm{H}}\mathrm{L}\ltimes$	$\rtimes\mathrm{HLL}\boxed{\mathrm{L}}\ltimes$	H	LL	$\lambda\|$L	4	L
6.	4	$\rtimes\boxed{\mathrm{H}}\mathrm{L}\ltimes$	$\rtimes\mathrm{HLLL}\boxed{\ltimes}$	H	L$\ltimes$	$\lambda\|\ltimes$	5	λ
7.	5	$\rtimes\mathrm{H}\boxed{\mathrm{L}}\ltimes$	$\rtimes\mathrm{HLLL}\boxed{\ltimes}$	L	L$\ltimes$	L$\|\lambda$	5	λ
8.	5	$\rtimes\mathrm{HL}\boxed{\ltimes}$	$\rtimes\mathrm{HLLL}\boxed{\ltimes}$	$\ltimes$	L$\ltimes$	$\ltimes\|\lambda$	5	λ
End	5							LHLL

Table 3: Rimi derivation of $\langle \mathrm{HL}, \mathrm{HLLL} \rangle \to \mathrm{LHLL}$.

As can be seen in the derivation, the primary job of the machine is to output an input H on the timing tier to the subsequent TBU. This can be seen in steps 2 and 3 of the derivation, in which an input H is initially output as λ to ‘wait’ to see if there is a subsequent TBU to shift to and then, in step 3 upon seeing a following L, outputting these two TBUs are output as a LH sequence, ‘shifting’ the input H one TBU to the right. Note that the machine only moves on the timing tier at this point. This is because the Rimi tone shift being a local process, it does not need the look-ahead option provided by a scan of the melody. (The last steps of the derivation simply move the melody read-head to the end of the tape.) Note that Rimi is IML because this machine satisfies all the properties in (6), notably that the machine computes deterministically with j- and k-windows of size 1 and 2, respectively.

5.2 Unbounded Tone Spread in Ndebele

In Ndebele (Bantu, Zimbabwe) H tone spreads unboundedly to any number of TBUs until the antepenultimate syllable (Sibanda 2004, Hyman 2011) as shown in (10); (11) gives the pattern’s function. Note that the unbounded spreading targets tonally unspecified TBUs, but for simplicity purposes, those tonally unspecified TBUs are represented with Ls. Furthermore, the domain of spreading is the phrase, meaning the boundary symbols mark the edges of the phrase. The Ndebele function examplified in (11) is only defined for inputs with a single H tone, for the sake of focusing on the unbounded spreading nature of the function.

(10) Ndebele Tone Spread (adapted from Hyman 2011)

a. /ú-ku-hlek-a/ $\rightarrow$ [ú-kú-hlek-a] 'to laugh'
b. /ú-ku-hlek-is-a/ $\rightarrow$ [ú-kú-hlé k-is-a] 'to amuse (make laugh)'
c. /ú-ku-hlek-is-an-a/ $\rightarrow$ [ú-kú-hlé k-ís-an-a] 'to amuse each other'

(11) Ndebele function examples

a. $f(\langle \text{HL,HLLLLL} \rangle) = \text{HHHHLL}$
b. $f(\langle \text{L,LLLLLL} \rangle) = \text{LLLLLL}$

This function is IML, as shown by the derivation in Table 4. In intuitive terms, following a high tone the Ndebele DM-FST outputs every low tone as high except for the last two, which it keeps track of by 'waiting' two inputs to the right before it outputs a symbol for an input L.

Step	q	Melody	Timing	j-win.	k-win.	$X\|Y$	r	Z
1.	0	$\boxed{\rtimes}\text{HL}\ltimes$	$\boxed{\rtimes}\text{HLLLL}\ltimes$	$\rtimes$	$\rtimes$	$\rtimes\|\rtimes$	1	λ
2.	1	$\rtimes\boxed{\text{H}}\text{L}\ltimes$	$\rtimes\boxed{\text{H}}\text{LLLL}\ltimes$	$\rtimes$H	$\rtimes$H	H|H	3	H
3.	3	$\rtimes\text{H}\boxed{\text{L}}\ltimes$	$\rtimes\text{H}\boxed{\text{L}}\text{LLL}\ltimes$	HL	HL	L|L	4	λ
4.	4	$\rtimes\text{H}\boxed{\text{L}}\ltimes$	$\rtimes\text{HL}\boxed{\text{L}}\text{LL}\ltimes$	HL	LL	λ|L	5	λ
5.	5	$\rtimes\text{H}\boxed{\text{L}}\ltimes$	$\rtimes\text{HLL}\boxed{\text{L}}\text{L}\ltimes$	HL	LL	λ|L	5	H
6.	5	$\rtimes\text{H}\boxed{\text{L}}\ltimes$	$\rtimes\text{HLLL}\boxed{\text{L}}\ltimes$	HL	LL	λ|L	5	H
7.	5	$\rtimes\text{HL}\boxed{\ltimes}$	$\rtimes\text{HLLLL}\boxed{\ltimes}$	HL	LL	$\ltimes\|\ltimes$	6	LL
End	6							HHHLL

Table 4: Ndebele derivation of $\langle \text{HL,HLLLL} \rangle \rightarrow \text{HHHLL}$.

This 'waiting' occurs in steps 4 and 5, in which the machine reads in L TBUs without outputting anything. Now in step 5, the machine has transitioned to state 5, meaning it has not only seen an H but also has waited enough (in two counts) to make sure the two symbols following the H on the timing tape are not the penult and the ultima, respectively. When the machine reads an L on the timing tape while staying put on the melody tape, When it sees the end of the string in step 7, it outputs two Ls and transitions to the accepting state 7. These final two Ls compensate for the waiting (λ outputs) in steps 3 and 4, which guarantee that the penult and final TBUs in the domain will never be spread onto. In this way, unbounded spreading can be computed by keeping track of a local window on both the melody and the timing tier, and is thus IML.

6. Discussion and Future Research

In this section, we discuss how IML functions and autosegmental phonology compare on one hand and how IML functions distinguish between attested long distance processes and the logically possible yet unattested processes.

As they are inspired by autosegmental phonology, IML functions share many properties with the latter. Firstly, IML functions employ the idea of independent timing and melody tiers. However, instead of representing associations between units on these tiers, the relationship between the melody and timing tiers in IML functions is indirect, as the former is derived from the latter through the melody function.

As a result, the well-formedness proposed in autosegmental phonology (Goldsmith 1976:p.48) are derived in IML functions through the operation of the melody function. For example, the no-crossing constraint is emergent as a byproduct of the melody function and the determinism of the IML functions. Because subsequentiality requires that symbols be read from one end going in one direction toward the opposite end, only two movement instructions are available to the read-head on each of the tapes of the corresponding DM-FST: *no movement/stay put* and *move to the subsequent symbol*. With these two movement instructions, a read-head can not go back in the opposite direction to read the symbol that precedes an already read symbol. As a result, there can never be cases (say, in a left subsequential function) where a symbol following the current symbol, say on the timing tier, is read at the same time as the symbol that precedes the last read symbol on the melody tier.

Given IML functions' ability to characterize complex phonological processes in tone, it is only logical to ask the question as to whether they also predict the existence of patterns we don't otherwise see in phonology. We make the following conjectures about the nature of IML functions. First, they do not compute *non-regular* functions, such as the majority rules (Baković 2000) and midpoint pathology (Eisner 1997) that can be constructed in OTP. We argue that this is likely true due to the fact that DM-FSTs are finite-state, and furthermore are *deterministic*. Additionally, while even multi-tape FSTs can capture a range of patterns (Rawski and Dolatian 2020), we conjecture that the IML functions are still sub-*regular*, meaning that the restriction to deterministic machines is still meaningful. This is due to the restricted nature of the melody tape, namely that it must be derived via the melody function from the timing tape. However, we leave it to future work to formalize with proofs the expressivity of the machines we have outlined here. Another avenue for such research would be the logical approach to comparing representations outlined in Strother-Garcia (2019) and Oakden (2020).

We have shown that the subsequentiality hypothesis for phonology is tenable given an enriched representation, which highlights the fact that representation matters in phonology. However, there is much further work to be done. First, while this paper has given some examples of tone processes that can be modeled with IML functions, a more comprehensive survey is the subject of ongoing work (Mamadou ms.).

Second, this paper has focused on patterns in which tones are associated to TBUs in the underlying representation. Futher work should reconcile the representations used here and those used for processes like those seen in the Manding languages (e.g., Mende), which are usually analyzed as not having an underlying association and there is a basic tone-TBU algorithm that link the symbols on the two tiers. Another representational question is one of underspecification. Like Jardine (2020), we have abstracted away from analyses in which, for example, surface L-toned TBUs are underlyingly unspecified for tone (as in Luganda). As Jardine shows, there are various ways to modify the `mel` function to accommodate representations with unspecified TBUs. Interested readers are referred to that paper for further details.

Finally, the subsequentiality hypothesis is not limited to tone, raising the question of whether the class of IML functions includes non-tonal phonological processes. IML functions are trivially adaptable to segmental processes, one only needs to replace tone symbols in the current alphabet with the symbols relevant to the segmental process in question, say a feature. However, a challenge with using features comes from cases of vowel harmony where the harmonizing feature only spreads onto vowels with specific (other) features. While giving a definite analysis for vowel harmony is beyond the scope of this paper, a tentative solution would be to allow interacting features to have a shared melody tier. This will also help keep the number of input tiers for the IML functions to two.

7. Conclusion

In this paper, we proposed and defined a new class of functions, the IML functions, which rely on an enriched representation made of a timing tier and the derived melody tier. We gave evidence that the class is descriptively adequate for tonal phonology by analyzing several tone patterns representing a range of processes with varying levels of complexity, including but not limited to unbounded circumambient processes. This class allows for the strong and testable phonological hypothesis that a phonological process must be subsequential over a string of TBUs and/or its derived melody. Future work will focus on the abstract characterization of IML functions and on how to apply them to a wider range of empirical examples.

References

Baković, Eric. 2000. Harmony, dominance and control. Doctoral dissertation, Rutgers University.

Chandlee, Jane. 2014. Strictly local phonological processes. Doctoral dissertation, University of Delaware.

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. Strictly locality and phonological maps. *LI* 49:23–60.

Chandlee, Jane, Adam Jardine, and Jeffrey Heinz. 2015. Learning repairs for marked structures. In *Proceedings of the 2015 Annual Meeting on Phonology*. LSA.

Eisner, Jason. 1997. Efficient generation in primitive Optimality Theory. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, 313–320. Madrid. URL `http://cs.jhu.edu/~jason/papers/#eisner-1997-acl`.

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

Goldsmith, John. 1976. Autosegmental phonology. Doctoral dissertation, Massachussets Institute of Technology.

Heinz, Jeffrey. 2007. The inductive learning of phonotactic patterns. Doctoral dissertation, UCLA.

Heinz, Jeffrey. 2009. On the role of locality in learning stress patterns. *Phonology* 26:303–351.

Heinz, Jeffrey. 2010. Learning long-distance phonotactics. *LI* 41:623–661.

Heinz, Jeffrey. 2018. The computational nature of phonological generalizations. In *Phonological typology*, ed. by 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 Mathematics of Language*, ed. by Andras Kornai and Marco Kuhlmann. Sofia, Bulgaria.

Heinz, Jeffrey, Chetan Rawal, and Herbert G Tanner. 2011. Tier-based strictly local constraints for phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, 58–64. Association for Computational Linguistics.

Hyman, Larry M. 2011. Tone: Is it different. *The handbook of phonological theory* 2.

Hyman, Larry M, and Francis X Katamba. 2010. Tone, syntax, and prosodic domains in luganda. *UC Berkeley PhonLab Annual Report* 6.

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

Jardine, Adam. 2016b. Learning tiers for long-distance phonotactics. In *Proceedings of the 6th Conference on Generative Approaches to Language Acquisition North America (GALANA 2015)*, 60–72.

Jardine, Adam. 2020. Melody learning and long-distance phonotactics in tone. *Natural Language and Linguistic Theory* 1–51.

Leben, William R. 1978. The representation of tone. In *Tone*, 177–219. Elsevier.

Mamadou, Tajudeen. ms. Long distance processes in tone and subsequentiality. 2^{nd} Qualifying Paper, Rutgers University, New Brunswick. Available at https://dinemamadou.wixsite.com/dinemamadou/papers/.

McCarthy, John J. 1986. OCP effects: gemination and antigemination. *LI* 17:207–263.

McCollum, Adam G., Eric Baković, Anna Mai, and Eric Meinhardt. 2020. Unbounded circumambient patterns in segmental phonology. *Phonology* 37:215–255.

McMullin, Kevin, and Gunnar Ólafur Hansson. 2019. Inductive learning of locality relations in segmental phonology. *Laboratory Phonology* 10.

McNaughton, Robert, and Seymour Papert. 1971. *Counter-free automata*. MIT Press.

Meyers, Scott. 1997. Ocp effects in optimality theory. *Natural Language & Linguistic Theory* 15:847–892.

Mohri, Mehryar. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics* 23:269–311.

Oakden, Chris. 2020. Notational equivalence in tonal geometry. *Phonology* 37:257–296.

Odden, David. 1986. On the role of the Obligatory Contour Principle in phonological theory. *Language* 62:353–383.

Odden, David. 1988. Anti antigemination and the ocp. *Linguistic inquiry* 19:451–475.

Odden, David. 1994. Adjacency parameters in phonology. *Language* 70:289–330.

Oncina, José, Pedro García, and Enrique Vidal. 1993. Learning subsequential transducers for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15:448–458.

Rawski, Jonathan, and Hossep Dolatian. 2020. Multi-input strict local functions for tonal phonology. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, ed. by Allyson Ettinger, Gaja Jarosz, and Max Nelson, volume 3. URL `https://scholarworks.umass.edu/scil/vol3/iss1/25`.

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

Schützenberger, Marcel Paul. 1977. Sur une variante des fonctions séquentielles. *Theoretical Computer Science* 4:47–57.

Sibanda, Galen. 2004. Verbal phonology and morphology of ndebele. berkeley, ca: University of california dissertation.

Strother-Garcia, Kristina. 2019. *Using model theory in phonology: a novel characterization of syllable structure and syllabification*. University of Delaware.

Wilson, Colin. 2003. Analyzing unbounded spreading with constraints: marks, targets, and derivations. Unpublished ms.

Wilson, Colin. 2006. Unbounded spreading is myopic. Paper presented at the workshop Current Perspectives on Phonology.

Tajudeen Mamadou, Adam Jardine
dine.mamadou@rutgers.edu, adam.jardine@rutgers.edu