PHONOLOGICAL ANALYSIS

All dissimilation is computationally subsequential

AMANDA PAYNE

University of Delaware

This article presents a computational analysis of the 185 dissimilation patterns in the typological surveys by Suzuki (1998) and Bennett (2013), and shows that dissimilation is computationally less complex than has been previously shown. Dissimilation patterns are grouped into three general types (basic, blocking, and polarity), each of which can be modeled with a subsequential finite-state transducer. This lends support to the claim that phonological patterns are not only regular, but in fact subsequential, which is a more restrictive class of patterns computationally and provides a stronger bound on the types of processes expected in natural language phonology.*

Keywords: dissimilation, Chomsky hierarchy, complexity, subsequentiality, subregular hypothesis

1. INTRODUCTION. Dissimilation, wherein two sounds in a word become less similar to one another (or the avoidance of similar sounds in a word is observed), is a well-attested phonological process both synchronically and diachronically (Suzuki 1998, Bennett 2013). Although it may be seen as the counterpart to assimilation, dissimilatory processes are often nonlocal and may involve segments that block the dissimilation. Thus, if one were to look for a computationally 'complex' pattern in phonology, long-distance dissimilation would be a good candidate, since complexity is often tied to locality.

This article establishes that the dissimilation patterns of the world's languages are computationally less complex than previously realized. It is well known that phonological grammars are REGULAR—that is, they can be expressed with finite-state transducers (Johnson 1972, Kaplan & Kay 1994). (See Beesley & Kartunnen 2003 for a linguistically oriented introduction to finite-state machines.) The analysis here reveals that each dissimilation pattern is SUBSEQUENTIAL, which means it can be expressed with a particular type of finite-state transducer, one that is either left- or right-deterministic. (The terms 'regular', 'subsequential', and 'deterministic' are all defined in more detail in §2.)

Because the subsequential class of patterns is a proper subclass of the regular class of patterns, it can be said that this article establishes a tighter computational characterization of dissimilation patterns than previously realized. In this way, this result is similar to other studies that have shown that other segmental processes in phonology are subsequential, such as metathesis and vowel harmony (Chandlee et al. 2012, Chandlee & Heinz 2012, Gainor et al. 2012, Heinz & Lai 2013). (See Jardine 2016 for an argument that tonal patterns are not.)

The empirical basis for the study comes from Bennett's (2013) 146-pattern typology of long-distance dissimilatory patterns, as well as fifty-seven segmental patterns from Suzuki's (1998) dissimilation typology. The study revealed only one potential counterexample from Yidiny (Dixon 1977). However, closer examination in §5.2 shows there is an analysis compatible with subsequentiality.

The main result of this article, that dissimilation patterns are subsequential, helps address one of the fundamental questions of generative linguistics, namely: What are the

* I am indebted to Jeffrey Heinz for his truly extensive feedback on this article, Eric Baković for his patient advice, and three anonymous referees who contributed detailed notes, all of whom have improved this paper significantly. Thanks as well to William Bennett and Keiichiro Suzuki for their in-depth dissimilatory descriptions, and to participants at NELS 44 and the University of Delaware computational linguistics group for their feedback. Any remaining errors are, of course, my own.

possible and impossible language patterns (Chomsky 1965)? The study here provides support for what could be called the SUBSEQUENTIAL HYPOTHESIS: segmental processes in phonology must be subsequential. This hypothesis predicts that certain phonological patterns (those that are nonsubsequential) cannot exist. An example of such a pattern—'unbounded circumambient assimilation'—is described in §2.2.

Beyond typology, the computational nature of linguistic patterns also has ramifications for processing, learning, and acquisition. If a pattern is regular, it means memory is restricted in an important way. Furthermore, with respect to subsequentiality, it is well known that deterministic finite-state machines process inputs more efficiently than nondeterministic ones. When it comes to learnability, Heinz and colleagues (2015:Ch. 3) argue that it is significantly easier to learn patterns that are describable with deterministic grammars. One illustrative example comes from Oncina and colleagues (1993), who provide the basis for the efficient, exact learning of subsequential patterns—the very kind shown here to include dissimilation patterns, including long-distance ones. This contrasts with negative results on learning the regular class of patterns for which provably no exact learning algorithm exists (Gold 1967). These learning results (see also Gildea & Jurafsky 1996), in conjunction with the evidence favoring the subsequential hypothesis, signal a way in which segmental processes can be plausibly acquired.

The rest of this article is organized as follows: a computational background, including a definition of subsequentiality and finite-state transducers, is provided first (§2), followed by a theoretical background for analyses of dissimilation and questions of representation (§3). Section 4 then describes the sources of typological data used in this article, as well as three subtypes of dissimilation patterns. The computational analysis of each subtype is given in §5, and the article concludes with a discussion of the results' significance (§6). An extensive appendix provides a categorization for each of the language patterns covered by this analysis, as well as specific subsequential finite-state transducers to generate each pattern.[1]

2. COMPUTATIONAL BACKGROUND. One of the most well-known systems for formalizing the complexity of language patterns is the CHOMSKY HIERARCHY (Chomsky 1956), a schema that divides grammars (that is, the sets of rules and symbols that generate languages) into different levels of complexity, as shown in Figure 1.

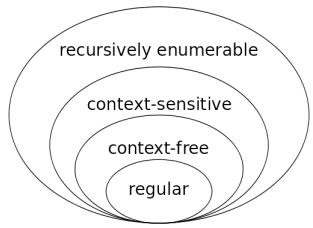

FIGURE 1. The Chomsky hierarchy.

The larger ovals can be seen to imply a greater level of complexity; that is, the grammars that generate CONTEXT-FREE languages can generate a larger set of languages than the grammars that generate REGULAR languages, and so on. The diagram represents a subset relation, so any language that is generated by a regular grammar could also be generated by a context-free grammar. Therefore the class of 'regular languages' is

[1] The appendix can be accessed as online supplementary material at http://muse.jhu.edu/resolve/30.

strictly smaller than the class of 'context-free languages' (languages generated by a context-free grammar). For further information on these measures of complexity, see Partee et al. 1990.

However, the Chomsky hierarchy classifies FORMAL LANGUAGES, which are sets of strings, and we are interested in the nature of phonological MAPS, which can be thought of as string-to-string functions. Maps take some input string and return some output string—for instance, they map an underlying form to a surface form. Phonological maps can be described with an ordered list of rewrite rules of the form A → B / C __ D (Chomsky & Halle 1968). They can also be described with OPTIMALITY-THEORETIC grammars (OT; Prince & Smolensky 1993, 2004). Rewrite-rule grammars have long been known to generate regular maps (Johnson 1972, Kaplan & Kay 1994), and OT grammars generate regular maps under certain conditions (Frank & Satta 1998, Riggle 2004).

Figure 2 shows a hierarchy for the class of regular maps, with finite maps being the least complex, and nonregular maps the most complex.

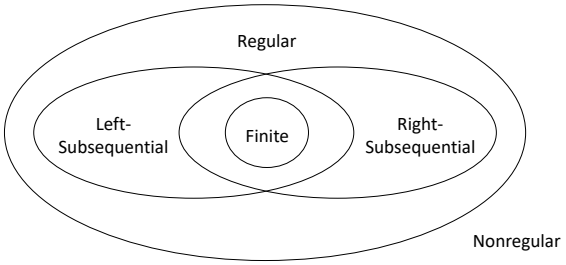

FIGURE 2. A hierarchy for string-to-string functions.

The relevant subset of the regular class for our purposes is the SUBSEQUENTIAL class, a class comprised of the union of the left-subsequential and right-subsequential patterns in Fig. 2. The subsequential class includes all patterns that can be described with a subsequential finite-state transducer (to be defined below). These subsequential patterns are strictly less complex than nonsubsequential regular patterns (Mohri 1997).

Before turning to the definition of subsequential finite-state transducers, we first consider what typifies FINITE-STATE TRANSDUCERS (FSTs) in general. An example FST for a simple rule changing A to B word-finally is shown in Figure 3.

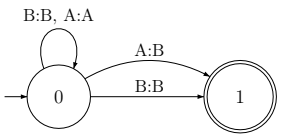

FIGURE 3. Nonsubsequential FST.

This FST describes the same map as the following rule: A → B / __ #. That is to say, A will change to B only at the end of a word. The FST describes this process in the following way: beginning in the state labeled '0', seeing a B as input (the left side of 'B:B') means that the FST will output a B (the right side of 'B:B') and return to state 0. Seeing an A could mean either that you loop back to state 0 and output an A, or that you transition to state 1 and output a B. This is a case of nondeterminism, since the single input A could follow either path. However, since only state 1 is a final state with no outgoing transition, only word-final A will end up being output as B, and the rest will re-

main As. A sample derivation for the string /BBAA/ is illustrated in 1. Here, the string is read from left to right (see §2.1 below for more on directionality).

(1)

Input		B		B		A		A	
State	0	→	0	→	0	→	0	→	1
Output		B		B		A		B	

First, starting in the initial state 0, the segment B is read, and looping to state 0 outputs an B. The second B is read, and again looping back to state 0, we output another B. Next we read an A, and so we follow the transition back to state 0 and output an A, since this is not a word-final segment. (If we had transitioned to state 1, then we would not be able to process the final input segment.) Finally, we read another A and transition to state 1, outputting a B. There are no more segments in the input, so we end in state 1.

Because this pattern is describable using an FST, it is regular, but this particular map is also subsequential, as represented in Figure 4. To be a subsequential FST (or sFST) requires two things. First, the FST is DETERMINISTIC on the input, meaning that for any given input in any state, there is only one possible path to take. Second, every final state in the FST also outputs a string when the derivation is finished. (It can also be the empty string ε.)

These restrictions mean that not every possible pattern can be described with a subsequential FST. In terms of phonology, the fact that the sFST must be deterministic means that from any point in the string, each output must be determinable from looking ahead and behind in the word only a bounded amount. At some point, you must choose a 'path' in the sFST. The outputs in the final state mean that eventually the string must have an ending point, and it can end by outputting some specific segment(s). An example of a nonsubsequential phonological pattern is given in §2.2 below. (Heinz and Lai (2013) and Jardine (2016) provide further examples of nonsubsequential functions relevant to phonology.)

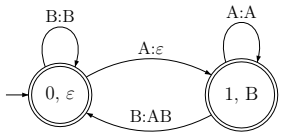

FIGURE 4. Left-to-right subsequential FST.

A derivation for the input /BBAA/ using the sFST in Fig. 4 is given in 2.

(2)

Input		B		B		A		A		
State	0	→	0	→	0	→	1	→	1	→
Output		B		B		ε		A		B

Here, the input begins in state 0, where the segment B is read, and looping to state 0 outputs a B. The second B is read, and again looping back to state 0, we output another B. Next we read an A, and so we follow the transition to state 1 and output the empty string. This empty string is a way to store the fact that an A was seen, so that upon seeing the next segment, if one exists, the A can be output, but if there is no other segment, a B can be output. Finally, we read another A, and loop to state 1, outputting an A, corresponding to the previous A from the input. There are no more segments in the input, so we end in state 1 and output a B, producing the output string *BBAB*.

2.1. DIRECTIONALITY. Another important distinction is whether these maps are left-to-right subsequential, or right-to-left ('reverse') subsequential. The previous example

was left-to-right subsequential, and it works well with patterns where the trigger for dissimilation is to the left of the target (progressive dissimilations). However, for regressive dissimilations, where the trigger is to the right of the target, a right-to-left subsequential FST is typically necessary.

A left-to-right subsequential function is recognized by a subsequential transducer. A right-to-left subsequential function is recognized by a subsequential transducer that processes the input string from right to left. In other words, if a pattern is right-to-left subsequential, the initial input string is REVERSED, then the sFST applies, and finally the result is reversed again to get the proper output.

The pattern described in Fig. 4, where A changes to B word-finally, is describable with both a left-to-right subsequential (as in 2) and right-to-left subsequential transducer. An example of a right-to-left subsequential map for this pattern is as follows.

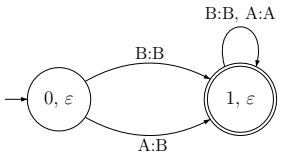

FIGURE 5. Right-to-left subsequential FST.

Consider again the pattern described by the rule A → B / __ #, now using the right-to-left subsequential FST in Figure 5. Here again /BBAA/ would surface as [BBAB]. To generate the output for such a map, if you had the input string /BBAA/, you would reverse it (*AABB*) before applying the sFST. The derivation would then proceed as shown in 3. The first segment in the input is now an A, so we move to state 1 and output a B. Reading the next A, we loop to state 1 and output an A. Next we read a B, looping to state 1, and output a B. Finally, we read the final B, looping back to state 1 and outputting a B. We end here in state 1 and output the empty string.

(3)	Input		A		A		B		B	
	State	0	→	1	→	1	→	1	→	1	→
	Output		B		A		B		B	ε

Finally the output of the sFST (*BABB*) would be reversed, yielding the correct output: *BBAB*.

2.2. SUBSEQUENTIAL PATTERNS ARE A PROPER SUBSET OF REGULAR PATTERNS. Finally, what about a regular pattern that is neither left-to-right nor right-to-left subsequential? Figure 6 shows a transducer for one such pattern, wherein a B will assimilate to an A if there is some A both preceding AND following it: B → A / AZ_0 __ Z_0A. (Here Z_0 stands for any number of segments of either type.) Note that the As may be unboundedly far from the B on both ends of the word. This cannot be classified as a progressive or regressive harmony pattern, since it is a combination of both. If this seems like a strange phonological pattern, that is in fact a good thing, because the pattern is not subsequential, and indeed none of the dissimilation patterns investigated here are of this nature. (Note that it fails because there is nondeterminism stemming from state 1, where a /B/ may be output as either an [A] or a [B] depending on what happens later.) Jardine (2016) calls these patterns UNBOUNDED CIRCUMAMBIENT and proves they cannot be subsequential.

A sample derivation for this FST is as follows. Consider the string /ABBA/, which maps to [AAAA], as shown in 4. First we read an A, and moving to state 1, output an A. Next, we read a B, and move to state 3, outputting an A. (Note that if we had chosen to

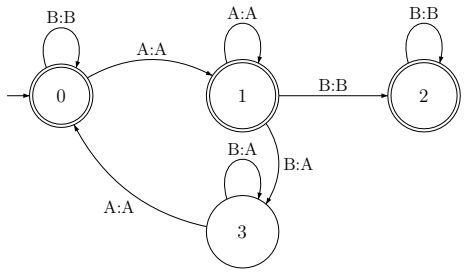

FIGURE 6. Unbounded circumambient assimilation.

move to state 2 instead, we would not be able to read any additional A segments, so we must move to state 3.) Reading the next B loops back to state 3, outputting another A. Finally we read an A, move to state 0, and output another A, ending in state 0.

(4)

Input		A		B		B		A	
State	0	→	1	→	3	→	3	→	0
Output		A		A		A		A	

3. THEORETICAL BACKGROUND. Dissimilation has been analyzed in a variety of ways in phonological theory. However, some of the more common approaches involve a tier-based autosegmental analysis using the OBLIGATORY CONTOUR PRINCIPLE (OCP) explicitly (Frisch et al. 2004), or an OT approach using local self-conjunction of constraints (Alderete 1997), which is essentially another way to encode OCP violations (Itô & Mester 1998). In an autosegmental approach, the dissimilating segments share a feature that is projected on a separate tier. In OT, constraints are used that ban a word from having multiple segments with the same feature value.[2]

One example of basic nonlocal dissimilation can be seen in Tashlhiyt Berber (5), where the first of two labial consonants dissimilates.[3]

(5) /**m**-kadda**b**/ → [**n**-kadda**b**] 'consider a liar (reciprocal)' (Jebbour 1985)

In OT, this can be analyzed with markedness constraints formed by local conjunction, following Alderete (1997). These constraints simply penalize multiple segments with the same feature value, as follows.

(6) a. LAB-2: Assign one violation for every word with two labial segments.[4]
b. COR-2: Assign one violation for every word with two coronal segments.
c. IDENT(LAB): Assign one violation for every segment with a different specification for [±labial] in the input and output.
d. IDENT(LAB)$_{stem}$: Assign one violation for every segment in the stem with a different specification for [±labial] in the input and output.

(7)

/m + kaddab/	*LAB-2	*COR-2	IDENT(LAB)	IDENT(LAB)$_{stem}$
☞ n-kaddab		*	*	
m-kaddab	*!			
m-kaddad		*	*	*!

[2] Note that these approaches are not incompatible, and it is possible to incorporate autosegmental representations in OT as well (Yip 2002).

[3] Tashlhiyt Berber dissimilation has many analyses, especially given that many of the changes are diachronic cooccurrence restrictions. The description presented here is just one possible option, and others are listed in the appendix, available at http://muse.jhu.edu/resolve/30.

[4] These 'local-conjunction' constraints are essentially STRICTLY PIECEWISE constraints with $k = 2$ (Rogers et al. 2010); they forbid two labials from appearing in the same domain.

An autosegmental account without the use of OT is equally successful in this case. Suppose that labials are projected to their own tier, and two labials with the same feature specification (in this case, any two labials within the same word) are forbidden by the OCP. This ill-formed structure is dealt with by a delinking and deletion of the [+labial] feature from the first segment, as well as an insertion and linking of a [+coronal] feature.

Each of the aforementioned analyses can give an account for dissimilation data, and each of them describes dissimilation in terms of an input-output map. In the OT analysis, the input comes in the form of a phonological string with morpheme boundaries encoded. This input is mapped to the phonological output string that least violates the highest-ranked constraints. In the autosegmental analysis, the input segments are strings of phonological segments that are linked to various features. This input maps to a linearized phonological output in which all OCP violations are removed. Though these accounts (as well as the computational account presented here) are not necessarily mutually exclusive, one might expect that different models could have differing explanatory power, or could account for different classes of patterns.

The computational analysis investigates the character of the map, regardless of whether it is represented with an OT grammar or autosegmental rules. It is worth noting that despite the various theoretical constructs (morphemes, features, etc.) deployed in each of the analyses of dissimilation, the linearized surface output from each analysis must turn out to be the same phonetically, regardless of the theory used to get there. In using a subsequential model to describe the input-output relationship, it becomes apparent that the complexity of the actual input-output MAP is constant, despite possible differences in the theoretical representation of the input string (see §4.1 for an example of this fact, regarding the representation of long vowels). The analysis presented here is only concerned with mapping string-to-string representations, so while it is not speaking to every aspect of the phonological literature, it relies on the common assumption that input and output representations are strings. However, that is not to say that alternative interpretations of the actual processes involved in phonological data are not relevant for computational complexity—in fact, the interpretation of the data can be crucial, as discussed in the Yidiny case in §5.2 below. This is very different from the alternative representational depictions of something like the long vowels discussed in §4.1, which do not change the computational complexity of the map.

It is worth noting that many, if not all, of the languages with dissimilatory patterns mentioned in this article contain lexical exceptions to the dissimilatory generalization for the language. However, what is important here is the nature of the actual dissimilation process (or rule, or constraint), rather than the fact that it does not apply strictly to every vocabulary item. Thus from this point forward, any exceptions are abstracted away from, but this does not affect the claims being made regarding the complexity of the dissimilation processes themselves.

Also of note is the fact that while dissimilation patterns from Bennett 2013 are long-distance and often bounded by the word, the patterns in Suzuki 1998 are occasionally local or bounded by syllables, moras, or other domain signifiers (indicated by DOM for each pattern in his appendix). Is it significant for the sFST analysis that the boundaries for the application of dissimilation rules are so variable? My assumption here is that each sFST applies to the relevant domain; however, it is possible to encode domain boundaries in an FST if necessary. For instance, we can easily output a word boundary symbol (rather than ε) at the end of the derivation, and the same goes for other boundary symbols, like a syllable or foot boundary. The corresponding outputs can then be concatenated into a single word (or sentence, or utterance). While issues of domain and

locality are important for many dissimilation analyses, because we are considering only the functional map of dissimilation and not its interaction with other phonological processes, bounded patterns are no more challenging for the analysis here than are unbounded patterns.

4. Dissimilation patterns. The two main sources for the dissimilation patterns analyzed in this article are Suzuki 1998 and Bennett 2013. Suzuki's (1998) appendix describes sixty-one dissimilation patterns from fifty-three languages (some languages have multiple unique dissimilation patterns). While Suzuki's appendix includes fifty-seven numbered patterns and four subpatterns, I make no distinction between patterns and subpatterns here and consider each as an individual pattern, even if many of them are quite similar. I have analyzed each of these sixty-one patterns, including the four tonal patterns (in Arusa, Bantu, Margi, and Peñoles Mixtec), which I consider nonsegmental. Although Suzuki's survey contains patterns that Bennett does not consider true cases of dissimilation (e.g. Finnish length dissimilation; Bennett 2013:586), I included them all, since the assertion that all dissimilation is subsequential is not weakened by their inclusion.[5] Bennett's (2013) appendix describes 146 dissimilation patterns from 131 languages, all of which are also included in this analysis. Forty-nine of the dissimilation patterns were described as morpheme structure constraints or cooccurrence restrictions; these are not processes, but can still be analyzed with either left-to-right or right-to-left subsequential transducers and are thus included in this study.

There are twenty-two patterns that appear in both Bennett 2013 and Suzuki 1998; in cases of overlap, the descriptions of Suzuki and Bennett occasionally conflicted. When there were conflicting pattern descriptions, additional sources listed in the appendix were used to clarify. The languages with significantly conflicting definitions were Salish: Moses-Columbia, where I ultimately used Suzuki's description based on Bessell & Czaykowska-Higgins 1993; Latin, where I include both a simplified version from Suzuki as well as the description from Bennett and from Cser 2010; Yidiny, where I use the analysis in Dixon 1977; Zulu, where I used Bennett's description based on Beckman 1993; and Yucatec Mayan, where I use Suzuki's description based on Yip 1989. Given the sixty-one patterns in Suzuki, plus the 146 patterns in Bennett, there are 207 total patterns; subtracting the twenty-two overlapping patterns yields the 185 dissimilation patterns included in this article's analysis.

These 185 patterns are listed in the separate appendix to this article, organized by type.[6] Each entry in the appendix provides the language name, pattern type, source, and an sFST for the dissimilation pattern. The nonbasic dissimilation types used here are based on the generalizations given in Bennett, for blocking, and Suzuki, for polarity. Many dissimilation patterns belong to more than one type. This can depend partly on the particular assumptions made. This is not unusual in formal language theory, where distinct classes do not necessarily imply disjoint groups of patterns. Most importantly, since each type will be shown to be subsequential, the fact that some patterns belong to more than one type will not challenge the subsequentiality hypothesis. For instance, only three patterns are categorized as 'blocking' patterns here, and they are what Bennett considers to be the three most empirically strong cases of segmental blocking patterns.

[5] In fact, since the subsequential hypothesis asserts that all segmental phonological patterns are subsequential, it is still support for the hypothesis that these patterns are subsequential, even if they are not necessarily dissimilation patterns.

[6] See the supplementary materials online at http://muse.jhu.edu/resolve/30.

In this section, I review the dissimilation patterns from both Suzuki and Bennett. These can be collapsed into the categories of basic dissimilation, blocking dissimilation, and polarity dissimilation.

4.1. PATTERN 1: BASIC DISSIMILATION. By far the most common dissimilation pattern is one wherein two underlying segments in some domain (typically a word) that share one or more features are mapped to two surface segments that no longer share the same feature(s). This BASIC DISSIMILATION subtype involves only one target segment and one trigger for the dissimilation; there is no segment that intervenes and prevents the dissimilation (see §4.2). Some languages that exhibit basic dissimilation are Arabic (Yip 1989), Cambodian (Yip 1989), Javanese (Padgett 1991), Russian (Padgett 1991), and Yucatec Mayan (Yip 1989) for [PLACE]; Ponapean (Goodman 1995) and Yao (Ohala 1981) for [LABIAL]; Akan (McCarthy & Prince 1995) and Dakota (Hong 1990) for [CORONAL]; and Moses-Columbia Salish (Bessell & Czaykowska-Higgins 1993) for [PHARYNGEAL].

This pattern is exemplified by Chukchi's restriction on cooccurring nasal segments (Krause 1979, Suzuki 1998). For instance, consider the Chukchi form /e.naw.rəŋ.nən/ 'he presented him', which surfaces as [e.naw.rəɣ.nən]. This dissimilatory map could be described as in 8.

(8) [+nasal] → [−nasal] / __ [+nasal]

Length dissimilation may also be classified as a particular case of basic dissimilation. Rather than a single segment dissimilating, length dissimilation involves a complex segment, but it follows the same basic template, much like *NC … NC dissimilation, another common example of basic dissimilation. Essentially, length dissimilation involves long consonants or vowels dissimilating to short segments when in the context of another long consonant or vowel. Oromo provides an example for vowel length dissimilation, as the plural suffix *-oota* dissimilates to *-ota* when it follows any other long vowel in the preceding syllable (Gragg 1976, Alderete 1997), as shown in 9.

(9)	a.	nam	'person'	→	nam-**oo**ta	'people'
		fardd	'horse'	→	fardd-**oo**ta	'horses'
	b.	g**aa**l	'camel'	→	gaal-**o**ta	'camels'
		l**oo**m	'lemon tree'	→	loom-**o**ta	'lemon trees'

Long-vowel dissimilation in Oromo presents the question of whether the way vowel length is represented matters for the computational analysis. As shown in §5.1 below, it does not. Long vowels can be represented as either complex segments or a sequence of two vowels in an sFST, and since the pattern of dissimilation relies only on searching in one direction (in this case, leftward) for an identically complex segment or sequence of two vowels, it is subsequential regardless. Although the prosodic structure conditioning such dissimilation may be hierarchical and of a higher level of representational complexity, the surface pattern itself is not. Of the 185 dissimilation patterns investigated, 178 of them are of the basic dissimilation type.

4.2. PATTERN 2: BLOCKING. Blocking patterns are those where one class of segments 'blocks' or prevents another segment from dissimilating, where the dissimilation would otherwise occur if the blocking segment were not present. Note that there is some ambiguity with regard to the meaning of the term 'blocking'. In a sense, dissimilation is 'blocked' whenever it does not occur, but this could happen either due to a lack of the proper environment for dissimilation, or because there is some particular segment that acts as a blocker. Bennett (2013) distinguishes between these cases with the terms 'non-

segmental blocking' and 'segmental blocking', respectively. The patterns Bennett describes as nonsegmental blocking are instances of basic dissimilation under the schema provided in §4.1, and I have classified them as such here. What Bennett (2013) calls segmental blocking is what I consider to be blocking in this analysis. Bennett focuses on exactly three cases of segmental blocking (Latin, Georgian, and Yidiny), and these are the three that I also classify as blocking.

In Latin liquid dissimilation, typically the second of two /l/s in a word will dissimilate to an /r/, as in 10a,b, UNLESS there is a blocking /r/ intervening between them, as seen in 10c.

(10) a. /so**l**-a**l**is/ 'solar' → [sol-a**r**is]
b. /**l**un-a**l**is/ 'lunar' → [lun-a**r**is]
c. /f**l**o**r**-a**l**is/ 'floral' → [flor-a**l**is] (Steriade 1987)

Cser (2010) notes an additional exception to the dissimilation process in 10: not only is /r/ a blocker, but so too are all noncoronal segments (e.g. /g/ in [le**g**-alis] 'legal'). The analysis in the exposition here has been simplified to include only /r/ as a blocking segment, but the complete pattern is still a blocking pattern and is available in Appendix A3 in the online supplementary materials.

The blocking dissimilation seen in Georgian is very similar to Latin's: the suffix /-uri/ is realized as [-uli] when following an /r/ (11b vs. 11a). However, this /r/ dissimilation is blocked by an intervening /l/, as seen in 11c.

(11) a. /polon-u**r**i/ 'Polish' → [polon-u**r**i]
b. /su**r**-u**r**i/ 'Assyrian' → [sur-u**l**i]
c. /ka**rtl**-u**r**i/ 'Kartvelian' → [kartl-u**r**i] (Odden 1994)

Yidiny dissimilation is also commonly analyzed as blocking involving liquids, though it is slightly more complicated. In this situation, /l/ dissimilates to [r] before another /l/ (12a), but the dissimilation is blocked when the first /l/ is preceded by an /r/ (12b,c), as described by Crowhurst and Hewitt (1995). There is more discussion about this suffix change in §5.2.

(12) a. /duŋga-ŋa**l**in-ŋa**l**/ 'went running with' → [duŋga-**r**i-ŋal]
b. /bu**r**wa-ŋa**l**in-ŋa**l**/ 'went jumping with' → [burwa-**l**i-ŋal]
c. /bu**r**gi-ŋa**l**in-ŋa**l**/ 'went walkabout with' → [burgi-**l**i-ŋal] (Dixon 1977)

At first glance, this Yidiny pattern appears similar to the unbounded circumambient assimilation in Fig. 5, since the target of the dissimilation is between, but arbitrarily far from, both the trigger and the blocker. However, an alternate account of the same Yidiny facts comes from Dixon 1977.

Dixon analyzes this apparent blocking as a double dissimilation, based on the fact that when the usual nonblocking /l/ → [r] dissimilation occurs, this is not just an /l/ → [r] before /l/ alternation, but the surface form also contains a different allomorph of the suffix. So, it is a two-step process: /bu**r**gi-ŋa**l**in-ŋa**l**/ → /bu**r**gi-**r**in-ŋa**l**/ (basic /l/ to [r] dissimilation, which includes suffix reduction) → [bu**r**gi-**l**in-ŋa**l**] (basic /r/ to [l] dissimilation). Or, as Dixon puts it:

> A case can be made out for saying that *-:li-n* dissimilates to *-:ri-n* under pressure from *-ŋa-l* (and the form *-:ri-n* is then generalised to all conjugations), and that, as a second stage, *-:ri-n* dissimilates to *-:li-n* in the presence of a rhotic (of either type) in the root. (1977:100)

Whether the pattern is a single dissimilation or a double one in this case really does matter for the subsequential hypothesis. The 'unbounded circumambient' description is not amenable to a subsequential analysis, but Dixon's analysis is, as explained in §5.2. However, although the interpretation of the surface data here is crucial to determining

its complexity, Dixon's description for the process under discussion removes the crucially nonsubsequential unbounded circumambient map.

4.3. PATTERN 3: POLARITY DISSIMILATION. Suzuki (1998) notes a special dissimilation pattern of POLARITY DISSIMILATION, where segments dissimilate from two 'opposing' directions—for instance, both high and low vowels dissimilating (high to low and low to high) or both high and low tone switching poles. This is essentially two dissimilation patterns described as one phenomenon, but since Suzuki's typology distinguishes it as a unique class, I maintain the distinction and show that this pattern is also subsequential.

An example of tone polarity dissimilation can be seen in Margi, where the second singular subject clitic *gu* appears with a high tone when it follows a low tone, as in 13a, and a low tone when it follows a high tone, as in 13b (Hoffmann 1963, Suzuki 1998).

(13) a. hègy**ì** g**ú** 'you are a Higi'
b. màrgy**í** g**ù** 'you are a Margi'

Polarity dissimilation with regard to voicing is seen in Chontal: the imperative suffix /laʔ/ begins with an /l/ after a voiceless segment (14a), and a /ɬ/ after a voiced segment (14b) (Kenstowicz & Kisseberth 1979, Suzuki 1998).

(14) a. fuš-**l**aʔ 'blow it!'
b. ko-**ɬ**aʔ 'say it!'

This type of dissimilation is further exemplified by a process called 'dissimilative jakan'e' in some southern dialects of Russian. A brief example of one dialect's dissimilative jakan'e, the 'Don subtype', is shown in 15. Here, nonhigh vowels dissimilate to low vowels when they are followed by a high vowel, as in 15a, and nonhigh vowels dissimilate to high vowels when they are followed by a low vowel, as in 15b (Davis 1970, Suzuki 1998).

(15) a. /rʲ**e**k**i**/ → [rʲ**a**k**i**] 'rivers'
b. /glʲ**a**d**a**t/ → [glʲ**i**d**a**t] 'they see'

Of the 185 dissimilation patterns investigated, four of them are of the polarity dissimilation type. They are Chontal, Dinka, Margi, and Russian.

5. COMPUTATIONAL ANALYSIS. This section demonstrates that each of the three classes of dissimilation patterns discussed in §4 can be defined explicitly and modeled using subsequential FSTs. Recall that this subsequential characterization is stronger than the existing claim that phonological patterns (like dissimilation) are regular. Therefore, this establishes that all attested long-distance dissimilations are subsequential processes, which places a tighter restriction on the possible dissimilation patterns expected in human languages.

5.1. PATTERN 1: BASIC DISSIMILATION. Recall that basic dissimilation patterns are of the following type: $AZ_0A \rightarrow AZ_0B$, applied left-to-right, where Z_0 represents any number of intervening non-A or non-B segments.

An sFST exemplifying a basic dissimilation pattern is shown in Figure 7. This sFST can be used for both left-to-right and right-to-left subsequential patterns.

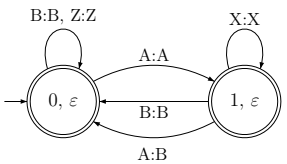

FIGURE 7. Basic dissimilation.

Recall the basic dissimilation pattern seen in Chukchi (§4.1), where the first of two nasal segments dissimilates to a nonnasal segment.

(16) /e.naw.rəŋ.nən/ → [e.naw.rəɣ.nən]

A more specific sFST to describe this type of pattern is shown in Figure 8, a right-subsequential transducer. In Fig. 8, [+nasal] stands for any nasal segment. Because the nasal feature partitions segments into two disjoint groups, there is no Z:Z loop nor X:X loop in Fig. 8 like the ones seen in Fig. 7.

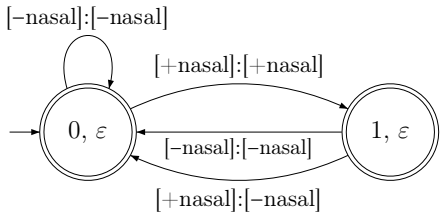

FIGURE 8. Chukchi basic regressive nasal dissimilation.

Consider how the Chukchi example, /enawrəŋnən/ → [enawrəɣnən], would be derived in this sFST. In this case we are concerned with the feature [+nasal]. Then, A = [+nasal], B = [−nasal], the nonnasal counterpart of the nasal, and Z = everything else. Because this is a right-to-left subsequential transduction, first the input string is reversed, yielding *nənŋərwane*. Beginning in the start state 0 (indicated by the arrow), the first segment seen is an /n/. This is [+nasal], so we follow the transition to state 1 while outputting it as the [+nasal] segment that it is: [n]. Next is the segment /ə/, which is [−nasal]. Therefore, the [−nasal] transition is followed back to state 0, and [ə] is output. Then, from state 0, the next segment seen is the [+nasal] /n/. Therefore, following the transition to state 1, this [+nasal] segment [n] is output. Next, we see the [+nasal] segment /ŋ/, and so we follow the transition to state 0, which outputs a [−nasal] counterpart, [ɣ]. Next, a [−nasal] /ə/ is seen and output as [ə], following the loop to state 0. The same occurs for the following [r], [w], and [a]. Then a nasal /n/ is seen, and following the transition to state 1, the [+nasal] string [n] is output. Finally, the nonnasal /e/ is read as we transition back to state 0, again outputting an [e]. Because there are no more remaining inputs, the process ends in state 0. We reverse the output string to obtain the desired result: [enawrəɣnən]. This derivation is shown in 17.

(17)

Input		n		ə		n		ŋ		ə		r		w		a		n		e	
State	0	→	1	→	0	→	1	→	0	→	0	→	0	→	0	→	0	→	1	→	0	→
Output		n		ə		n		ɣ		ə		r		w		a		n		e	ε

REPRESENTATIONAL ISSUES. Recall the Oromo vowel length dissimilation in 9, partially repeated in 18.

(18) fardd ‘horse’ → fardd-oota ‘horses’
gaal ‘camel’ → gaal-ota ‘camels’

If long vowels are considered complex single segments, this is clearly just a case of basic dissimilation. Here, then, the language-particular sFST is simplified to the following, where a nonspecified V represents a short vowel, and A = [+long]V, B = V, and Z = C.

A derivation for /gaːloːta/ using the sFST in Figure 9 would then proceed as follows.

(19)

Input		g		aː		l		oː		t		a	
State	0	→	0	→	1	→	1	→	0	→	0	→	0	→
Output		g		aː		l		o		t		a	ε

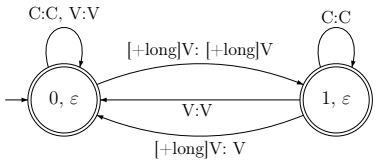

FIGURE 9. Vowel length dissimilation (complex segments).

However, one might prefer to represent each vowel segment individually. In this case, there is still a possible sFST for the pattern, shown in Figure 10.

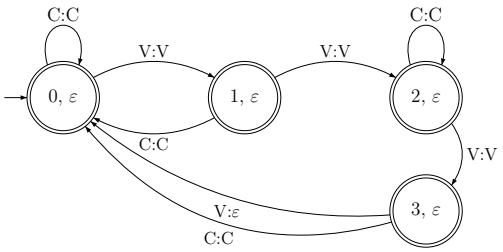

FIGURE 10. Vowel length dissimilation (individual segments).

A derivation using the sFST with individual segments in Fig. 10 for /gaaloota/ would proceed as in 20.

(20) Input g a a l o o t a
State 0 → 0 → 1 → 2 → 2 → 3 → 0 → 0 → 1 →
Output g a a l o ε t a ε

These two FSTs are functionally equivalent for Oromo—they produce identical outputs from the same input string; the only difference is the theoretical representation of long vowels. In some cases, if the long vowels in question must be two identical segments (e.g. *aa* but not *ai*), it might be more illustrative to represent long vowels as complex segments, though an sFST with separate states for each individual vowel could be used to produce the same pattern. While the specific choice of representation for long vowels (or consonants) may have ramifications for certain linguistic theories, such a choice has no impact either on the final output of the process phonetically or on the complexity of the pattern (it is subsequential regardless), provided the sFST is written correctly.

Similarly, while many dissimilation analyses (especially OCP-based ones) appeal to dissimilation operating only on specific 'featural' tiers, the resulting maps should be the same, regardless of tiers. The input/output map is identical. This is also the case for the polarity patterns of §5.3—regardless of the representation of segments used, the underlying dissimilation processes turn out to be subsequential.

5.2. PATTERN 2: BLOCKING. Blocking dissimilation patterns are of the following type: AZ*A → AZ*B, where Z* represents any number of intervening non-A, -B, or -X segments. X is a blocking segment, for example, AZ*XZ*A → AZ*XZ*A. In other words, dissimilation proceeds as in basic dissimilation, EXCEPT that when segments of type X appear, the dissimilation does not occur. It is true that blocking dissimilation and basic dissimilation are nearly identical, and in fact, a great many of the basic dissimilation patterns investigated here could fit into a blocking dissimilation template. However, only those patterns that have been particularly noted to be so-called 'segmental blocking dissimilation' patterns in Bennett 2013 are classified as such here, as discussed in §4.2.

An sFST for this pattern is in Figure 11. Note that unlike the basic dissimilation sFST of Fig. 7, this sFST has an additional transition from state 1 to state 0, allowing for the dissimilation trigger to 'reset' and essentially become inactive if a blocking segment (X) occurs.

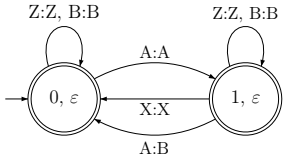

FIGURE 11. Blocking dissimilation.

Recall the /r/ blocking found in Latin liquid dissimilation in 10, with additional examples given in 21.

(21) /nav-a**l**is/ 'naval' → [nav-a**l**is]
/mi**l**it-a**l**is/ 'military' → [milit-a**r**is]
/**l**itor-a**l**is/ 'of the shore' → [litor-a**l**is] (the /r/ blocks dissimilation)

This Latin blocking pattern can be modeled as in Figure 12, where A = /l/, B = /r/, Z = nonliquid consonants and vowels, and X = /r/.

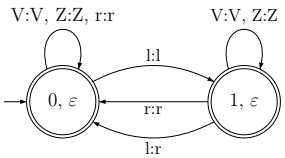

FIGURE 12. Latin blocking dissimilation (simplified).

Consider the derivation for /litoralis/. Starting in state 0, we read an /l/ and move to state 1, outputting the [l]. Then we read /i/, which is a V, and output this [i] and loop back to state 1. Next we read a /t/, which is a nonliquid C, so we output this [t] and remain in state 1. Similarly we read an /o/, loop back to state 1, and output this [o]. Next we read an /r/, the blocking segment, and transition to state 0, outputting the [r]. We read an /a/ (another V), loop back to state 0, and output this [a]. Now when we see this /l/, we again transition to state 1 and output the [l]—the dissimilation does not occur. Finally we read an /i/, looping to state 1 and outputting the [i], and we read an /s/, looping to state 1 and outputting the [s]. We finish the derivation in state 1 and output the empty string, *ε*.

(22)

Input	l		i		t		o		r		a		l		i		s	
State	0	→	1	→	1	→	1	→	1	→	0	→	0	→	1	→	1	→	1	→
Output	l		i		t		o		r		a		l		i		s		*ε*	

The more nuanced picture of Latin dissimilation shown in Cser 2010, where all noncoronals are blockers, can be represented easily as a blocking dissimilation, as seen in Figure 13. A derivation for /legalis/ using Fig. 13 is shown in 23.

(23)

Input	l		e		g		a		l		i		s	
State	0	→	1	→	1	→	0	→	0	→	1	→	1	→	1	→
Output	l		e		g		a		l		i		s		*ε*	

YIDINY DOUBLE DISSIMILATION. As noted in §4.2, dissimilation in Yidiny APPEARS to be a complex case of blocking dissimilation, where the blocker is to the left of the target

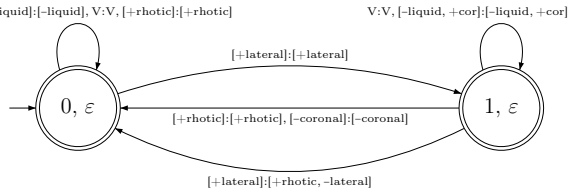

FIGURE 13. Latin blocking dissimilation (complete).

and the trigger is to the right. Such a pattern, if unbounded, would not be subsequential. However, per Dixon's description in §4.2, this blocking dissimilation is really just two distinct basic dissimilation processes, both of which are subsequential, as per §5.1.

If one wanted to think of their combination as a single process, that process would be WEAKLY DETERMINISTIC, a concept explained in Heinz & Lai 2013. A weakly deterministic pattern is a composition of a left-to-right subsequential (here, the left-to-right dissimilation) and right-to-left subsequential (the right-to-left dissimilation) process, which does not require any additional symbols in the intermediate representation. While this weakens the subsequential hypothesis to some degree, Heinz and Lai argue that it is still significantly more restricted than entirely nonsubsequential patterns like the unbounded circumambient assimilation seen in §2.2.

5.3. PATTERN 3: POLARITY DISSIMILATION. Polarity dissimilation patterns are of the following type: AY*B → AY*Z and ZY*B → ZY*A, where A and Z are opposite 'poles'—something like high- and low-toned segments. B is a segment of the same type as A and Z, but not at either 'pole' (e.g. a mid vowel, or a segment with neutral tone). Y* represents any number of intervening non-A, -B, or -Z segments. This pattern may look complicated, but in fact, it is really two basic dissimilations considered in union. An sFST schema is as in Figure 14.

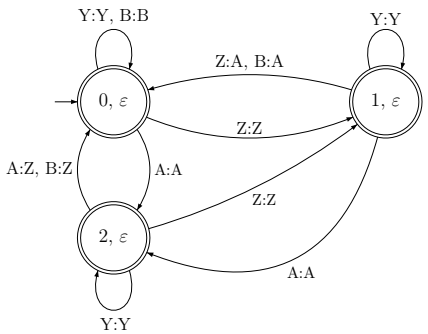

FIGURE 14. Polarity dissimilation.

An example of polarity dissimilation is seen in Russian (as in 15, repeated below as 24), where only nonhigh vowels undergo polarity dissimilation—becoming low before a high vowel, and high before a low vowel.

(24) /rʲ**e**k**i**/ → [rʲ**a**k**i**] 'rivers'
/glʲ**a**d**a**t/ → [glʲ**i**d**a**t] 'they see'

The sFST for this pattern is right-subsequential, and would look like Figure 15, with A = [+high, −low], B = [−high, −low], Y = consonants, Z = [−high, +low].

Using this sFST, a derivation for [rʲaki] is as follows. First, reverse the input: /rʲeki/ → *ikerʲ*. Then follow the sFST to obtain the map in 25. Starting in state 0, we read an /i/,

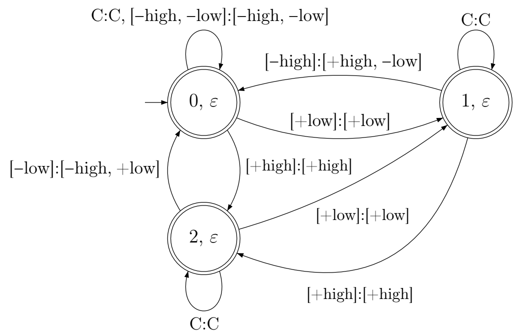

FIGURE 15. Russian polarity dissimilation.

which is a high vowel, and go to state 2, outputting the high vowel [i]. Next we read the consonant /k/, loop back to state 2, and output [k]. Reading an /e/, which is a nonhigh vowel, we go to state 0, and output a low vowel [a]. Next we read the consonant /r^j/, staying in state 0, and output a [r^j], ending in state 0.

(25)

Input		i		k		e		r^j		
State	0	→	2	→	2	→	0	→	0	→
Output		i		k		a		r^j		ε

Finally, reverse the result to obtain the surface form: *ikar^j* → [r^jaki].

The process proceeds similarly for a word like [gl^jidat], which dissimilates to a high vowel rather than a low one. Again, we reverse the input: /gl^jadat/ → *tadal^{j}g*. Following the sFST in Fig. 15 to get the derivation in 26, we read and output a /t/, remaining in state 0. Next we read the low vowel /a/ and go to state 1, outputting the [a]. Next we read the consonant /d/, looping back to state 1, and output [d]. Reading an /a/, which is a nonhigh vowel, we go to state 0 and output a high vowel [i]. We read the consonant /l^j/, staying in state 0, and output a [l^j], and then another consonant /g/, looping to state 0 and outputting the [g], before ending in state 0.

(26)

Input		t		a		d		a		l^j		g		
State	0	→	0	→	1	→	1	→	0	→	0	→	0	→
Output		t		a		d		i		l^j		g		ε

Finally, reverse the result to obtain the surface form: *tadil^{j}g* → [gl^jidat].

6. CONCLUSIONS. An investigation of the dissimilation patterns in Bennett 2013 and Suzuki 1998 reveals that each of these patterns can be described by a subsequential FST. This result has significance for many reasons. If every dissimilation pattern is subsequential, it is further evidence that segmental phonology in general is subsequential. If true, this would have ramifications for typology, learnability, and acquisition. Typologically, the subsequentiality hypothesis provides a strong, falsifiable claim about whether a certain pattern could exist in natural language. For instance, a dissimilation pattern similar to the circumambient unbounded example in §2.2 is predicted to be impossible. Consider the hypothetical example in 27 below, where an /l/ dissimilates to an /r/ only if there is another /l/ both preceding and following it somewhere in the word.

(27) /karo**l**inia**l**/ → [karo**l**inia**l**]
/**l**amako**l**inia**l**/ → [**l**amako**r**inia**l**]
/**l**amakoakoakoakoakoakoakoakoako**l**inia**l**/ → [**l**amakoakoakoakoakoakoakoako**r**inia**l**]

Patterns like those in 27 may seem like plausible ones, but if the distance between target and trigger is truly unbounded, such a pattern is not subsequential and thus not predicted to be possible in segmental phonology.

The learning results from Oncina and colleagues (1993) show that the class of total left-to-right subsequential patterns is learnable, and Gildea and Jurafsky (1996) adapt their algorithm for phonology. This means that all dissimilation patterns are learnable as well with these techniques, which lends more credence to the subsequentiality hypothesis.

Future research avenues include investigating the possibility of an even more restrictive characterization than subsequentiality for phonological patterns, as well as building on learning results. Some work in this avenue for bounded patterns includes Chandlee 2014 and Chandlee et al. 2014. Related work has investigated the psycholinguistic predictions for learning artificial language patterns in various classes of complexity (Lai 2015). This too is a ripe area for future investigation.

Though some details of individual language patterns may have been simplified in this account, such simplifications do not endanger the conclusion that each of the patterns is subsequential in nature. This means that dissimilation processes in general, insofar as they have been described (by Suzuki and Bennett), are less complex and more restrictive than regular relations. It may be the case, therefore, that all segmental phonological patterns are restricted in this way, placing a more narrow bound on the possible phonological patterns one could expect to find in the world's languages. In other words, this article has shown that 'being subsequential' is a universal property of dissimilation patterns, and this is the strongest known computational universal property of these patterns to date.

REFERENCES

Alderete, John. 1997. Dissimilation as local conjunction. *North East Linguistic Society (NELS)* 27.17–32.

Beckman, Jill N. 1993. Feature organization and the strong domain hypothesis in Zulu [labial] phonology. *Phonological representations* (University of Massachusetts occasional papers in linguistics 16), ed. by T. Sherer, 1–26. Amherst, MA: GLSA Publications.

Beesley, Kenneth R., and Lauri Kartunnen. 2003. *Finite state morphology*. Stanford, CA: CSLI Publications.

Bennett, William G. 2013. *Dissimilation, consonant harmony, and surface correspondence*. New Brunswick, NJ: Rutgers University dissertation.

Bessell, N. J., and Ewa Czaykowska-Higgins. 1993. The phonetics and phonology of postvelar consonants in Moses-Columbia Salish (Nxaamxcín). Technical report.

Chandlee, Jane. 2014. *Strictly local phonological processes*. Newark: University of Delaware dissertation.

Chandlee, Jane; Angeliki Athanasopoulou; and Jeffrey Heinz. 2012. Evidence for classifying metathesis patterns as subsequential. *West Coast Conference on Formal Linguistics (WCCFL)* 29.303–9. Online: http://www.lingref.com/cpp/wccfl/29/paper2715.pdf.

Chandlee, Jane; Rémi Eyraud; and Jeffrey Heinz. 2014. Learning strictly local subsequential functions. *Transactions of the Association for Computational Linguistics* 2.491–503. Online: https://transacl.org/ojs/index.php/tacl/article/view/429.

Chandlee, Jane, and Jeffrey Heinz. 2012. Bounded copying is subsequential: Implications for metathesis and reduplication. *Proceedings of the 12th meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, 42–51. Online: https://dl.acm.org/citation.cfm?id=2390936.

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 2.113–24. DOI: 10.1109/TIT.1956.1056813.

Chomsky, Noam. 1965. *Aspects of the theory of syntax*. Cambridge, MA: MIT Press.

Chomsky, Noam, and Morris Halle. 1968. *The sound pattern of English*. New York: Harper & Row.

Crowhurst, Megan, and Mark Hewitt. 1995. Prosodic overlay and headless feet in Yidiɲ. *Phonology* 12.39–84. DOI: 10.1017/S0952675700002384.

Cser, András. 2010. The *-alis*/*-aris* allomorphy revisited. *Variation and change in morphology: Selected papers from the 13th International Morphology Meeting, Vienna*, ed.

by Franz Rainer, Wolfgang U. Dressler, Dieter Kastovsky, and Hans Christian Luschützky, 33–51. Amsterdam: John Benjamins.
Davis, P. W. 1970. A classification of the dissimilative jakané dialects of Russian. *Orbis* 19(2).360–76.
Dixon, R. M. W. 1977. *A grammar of Yidiñ*. Cambridge: Cambridge University Press.
Frank, Robert, and Giorgio Satta. 1998. Optimality theory and the generative complexity of constraint violability. *Computational Linguistics* 24(2).307–15. Online: https://dl.acm.org/citation.cfm?id=972739.
Frisch, Stefan A.; Janet B. Pierrehumbert; and Michael B. Broe. 2004. Similarity avoidance and the OCP. *Natural Language and Language Theory* 22.179–228. DOI: 10.1023/B:NALA.0000005557.78535.3c.
Gainor, Brian; Regine Lai; and Jeffrey Heinz. 2012. Computational characterizations of vowel harmony patterns and pathologies. *West Coast Conference on Formal Linguistics (WCCFL)* 29.63–71. Online: http://www.lingref.com/cpp/wccfl/29/paper2688.pdf.
Gildea, Daniel, and Dan Jurafsky. 1996. Learning bias and phonological-rule induction. *Computational Linguistics* 22(4).497–530. Online: https://dl.acm.org/citation.cfm?id=256329.256335.
Gold, E. Mark. 1967. Language identification in the limit. *Information and Control* 10.447–74. DOI: 10.1016/S0019-9958(67)91165-5.
Goodman, Beverly D. 1995. *Features in Ponapean phonology*. Ithaca, NY: Cornell University dissertation.
Gragg, Gene. 1976. Oromo of Wellegga. *The non-Semitic languages of Ethiopia*, ed. by M. Lionel Bender, 166–95. East Lansing: Michigan State University, African Studies Center.
Heinz, Jeffrey; Colin de la Higuera; and Menno van Zaanen. 2015. Grammatical inference for computational linguistics. *Synthesis Lectures on Human Language Technologies* 8.1–139. DOI: 10.2200/S00643ED1V01Y201504HLT028.
Heinz, Jeffrey, and Regine Lai. 2013. Vowel harmony and subsequentiality. *Proceedings of the 13th Meeting on Mathematics of Language*, 52–63. Online: https://aclanthology.info/pdf/W/W13/W13-3006.pdf.
Hoffmann, Carl. 1963. *A grammar of the Margi language*. Oxford: Oxford University Press.
Hong, Sung-Hoon. 1990. OCP-triggered rules in Dakota and their interaction with morphology. Tucson: University of Arizona, ms.
Itô, Junko, and Armin Mester. 1998. Markedness and word structure: OCP effects in Japanese. Santa Cruz: University of California, Santa Cruz, ms. Online: http://roa.rutgers.edu/article/view/265.
Jardine, Adam. 2016. Computationally, tone is different. *Phonology* 33.247–83. DOI: 10.1017/S0952675716000129.
Jebbour, Abdelkrim. 1985. La labio-vélarization en Berbère dialecte Tachelhit (parler de tiznit). *Memoire de phonologie*. Rabat: Université Mohammed V.
Johnson, C. Douglas. 1972. *Formal aspects of phonological description*. The Hague: Mouton.
Kaplan, Ronald M., and Martin Kay. 1994. Regular models of phonological rules systems. *Computational Linguistics* 20.331–78. Online: https://dl.acm.org/citation.cfm?id=204917.
Kenstowicz, Michael, and Charles Kisseberth. 1979. *Generative phonology: Description and theory*. New York: Academic Press.
Krause, Scott Russell. 1979. *Topics in Chukchee phonology and morphology*. Urbana-Champaign: University of Illinois dissertation.
Lai, Regine. 2015. Learnable vs. unlearnable harmony patterns. *Linguistic Inquiry* 46(3).425–51. DOI: 10.1162/LING_a_00188.
McCarthy, John J., and Alan Prince. 1995. Faithfulness and reduplicative identity. *Papers in optimality theory* (University of Massachusetts occasional papers in linguistics 18), ed. by Jill N. Beckman, Laura Walsh Dickey, and Suzanne Urbanczyk, 249–384. Amherst, MA: GLSA Publications.
Mohri, Mehryar. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics* 23.269–311. Online: https://dl.acm.org/citation.cfm?id=972698.

Odden, David. 1994. Adjacency parameters in phonology. *Language* 70.289–330. DOI: 10.2307/415830.
Ohala, John J. 1981. The listener as a source of sound change. *Chicago Linguistic Society (Parasession on language and behavior)* 17(2).178–203.
Oncina, J.; P. Garcia; and E. Vidal. 1993. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15(5).448–58. DOI: 10.1109/34.211465.
Padgett, Jaye E. 1991. *Stricture in feature geometry*. Amherst: University of Massachusetts, Amherst dissertation.
Partee, Barbara; Alice ter Meulen; and R. Wall. 1990. *Mathematical methods in linguistics*. Dordrecht: Kluwer.
Prince, Alan, and Paul Smolensky. 1993. Optimality theory: Constraint interaction in generative grammar. Technical report 2. New Brunswick, NJ: Rutgers University Center for Cognitive Science.
Prince, Alan, and Paul Smolensky. 2004. *Optimality theory: Constraint interaction in generative grammar*. Malden, MA: Blackwell.
Riggle, Jason. 2004. *Generation, recognition, and learning in finite state optimality theory*. Los Angeles: University of California, Los Angeles dissertation.
Rogers, James; Jeffrey Heinz; Gil Bailey; Matt Edlefsen; Molly Visscher; David Wellcome; and Sean Wibel. 2010. On languages piecewise testable in the strict sense. *The mathematics of language*, ed. by Christian Ebert, Gerhard Jäger, and Jens Michaelis, 255–65. Berlin: Springer. DOI: 10.1007/978-3-642-14322-9_19.
Steriade, Donca. 1987. Redundant values. *Chicago Linguistic Society* 23(2).339–62.
Suzuki, Keiichiro. 1998. *A typological investigation of dissimilation*. Tucson: University of Arizona dissertation. Online: http://hdl.handle.net/10150/288816.
Yip, Moira. 1989. Feature geometry and cooccurrence restrictions. *Phonology* 6(2).349–74. DOI: 10.1017/S0952675700001068.
Yip, Moira. 2002. *Tone*. Cambridge: Cambridge University Press.

[amandapa@udel.edu]

[Received 12 March 2016;
revision invited 28 July 2016;
revision received 6 November 2016;
accepted with revisions 2 May 2017;
revision received 16 May 2017;
accepted 28 June 2017]