Strict Locality and Phonological Maps

Jane Chandlee and Jeffrey Heinz

(to appear in *Linguistic Inquiry*, Winter 2018)

Abstract

In this paper we identify *Strict Locality* as a strong computational property of a certain class of phonological maps from underlying to surface forms. We show that these maps can be modeled with Input Strictly Local (ISL) functions, a previously undefined class of subregular relations. These functions extend the conception of locality from the Strictly Local formal languages (recognizers/acceptors) (McNaughton and Papert, 1971; Rogers and Pullum, 2011; Rogers *et al.*, 2013) to maps (transducers/functions) and therefore formalize the notion of phonological locality. We discuss the insights such computational properties provide for phonological theory, typology, and learning.

Keywords: locality, phonological maps, computational properties, typology, learning

1 Introduction

This paper proposes that locality in phonology is best characterized with the computational property known as Strict Locality. We will show that the maps from underlying to surface forms that correspond to local phonological processes can be modeled with *Input Strictly Local (ISL) functions*, which we define by generalizing the Strictly Local formal languages (McNaughton and Papert, 1971; Rogers and Pullum, 2011; Rogers *et al.*, 2013) to string-to-string functions. Equally important is the fact that many logically possible but phonologically bizarre processes do *not* have this property. In addition to this strong typological prediction, we explain how Strict Locality also naturally provides an inductive principle for learners to generalize correctly from finitely many examples of the map (Chandlee *et al.*, 2014).

This research extends the work of Heinz (2007, 2009, 2010) and Heinz *et al.* (2011), which identified restrictive computational properties of the sets of strings that obey natural language phonotactic patterns. These properties not only non-trivially approximate the range of attested phonotactic patterns, they also reveal the structure of these patterns in a way that makes learning them more feasible. Grounded in formal language theory, the properties are drawn from a hierarchy of subregular language families, each of which has several independently-motivated mathematical characterizations (see Figure 1). Such mathematical descriptions provide a classification system for natural language patterns. Surveys of segmental phonotactic patterns indicate that virtually all can be modeled with NonCounting formal languages (Graf, 2010), and the vast majority with Strictly Local and Strictly Piecewise ones (Heinz, 2010; Heinz *et al.*, 2011; Rogers *et al.*, 2013; Heinz, 2014). Additionally, these classes have well-defined cognitive parallels that suggest exactly the minimum information a cognitive mechanism needs to be mindful of in order to

correctly compute whether strings belong to the language (Rogers and Pullum, 2011; Berwick *et al.*, 2011; Rogers *et al.*, 2013). This collective work on the computational nature of phonotactics amounts to a theory of possible markedness constraints, when these are conceived of as sets of valid strings. The current work takes the next step and proposes a theory of valid input-output maps.

This line of research that aims to characterize natural language patterns based on their computational properties originated with early work in generative grammar and is typically associated with studies in syntax (Chomsky, 1956; Berwick, 1982, 1985; Joshi, 1985; Schieber, 1985), though phonological processes have been studied in this manner as well. Since Johnson (1972), Koskenniemi (1983), and Kaplan and Kay (1994) it has been known that phonological rewrite rules of the form $A \rightarrow B \ / \ C \_\_ D$ are *regular* relations provided the rule cannot re-apply to the locus of its structural change. Regular relations can be defined in various ways, one being a string-to-string relation that can be described with a finite state transducer (FST), a formalism we will introduce in some detail in §3. Though stated in terms of phonological rules, these previous results do not in fact depend on a rule-based conception of phonological processes, as they hold of the input-output map itself, not the rule.

In this paper we propose that local phonological maps (i.e., those maps that involve a contiguous substring of the input of bounded length) are not just regular relations but in fact belong to a proper *subregular* class of relations we call Input Strictly Local functions.[1] Though the theory of subregular maps is not as well developed as the one for formal languages shown in Figure 1, the insights the latter have provided for phonotactic patterns suggest that developing a comparable hierarchy of relations/functions with which to study phonological maps will be an equally fruitful endeavor. This research then represents the first step towards that goal, starting with the most restrictive class of maps as the base of the hierarchy.

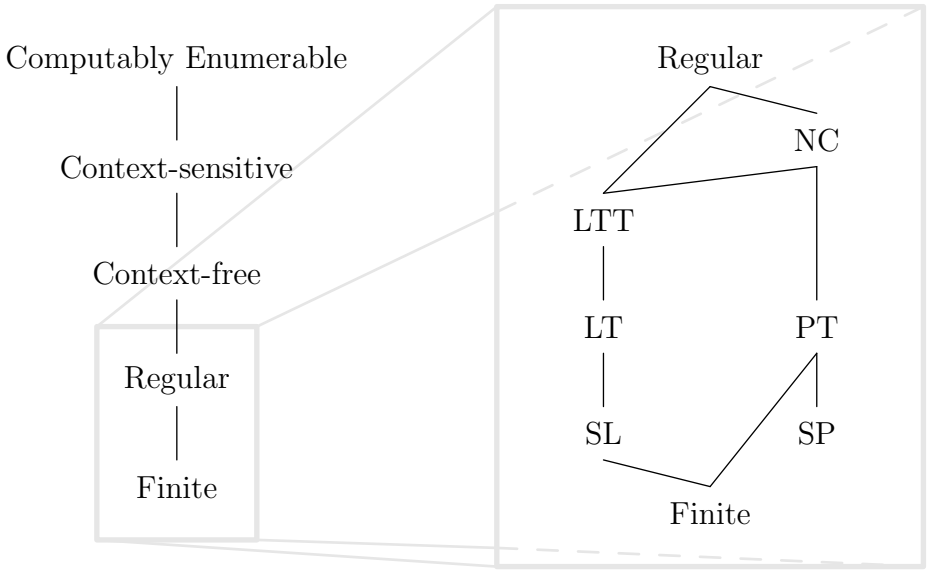

Figure 1: Chomsky Hierarchy of Languages and the Subregular Hierarchy of Languages. Lines indicate proper inclusion from top to bottom. NC = NonCounting, LTT = Locally Threshold Testable, LT = Locally Testable, SL = Strictly Local, PT = Piecewise Testable, SP = Strictly Piecewise.

As mentioned above, the criterion for a phonological map to belong to this most restrictive class of the hierarchy is that its target and triggering context form a contiguous substring of bounded length. In rule-based terms, this is a process describable with a rule of the form A $\rightarrow$ B / C __ D in which CAD is a finite set of strings (though again the characterization does not depend on a rule-based framework). We present several examples of how phonological maps that meet this criterion can be modeled with ISL functions. In addition, we show that at least some maps that comprise multiple, potentially interacting processes can also be modeled with ISL functions. We then discuss several categories of maps that do not meet this condition and are therefore not describable with ISL functions. Such maps include vowel harmony with transparent vowels (Nevins, 2010; Walker, 2011) and long-distance consonant agreement (Hansson, 2001, 2010; Rose and Walker, 2004) and dissimilation (Suzuki, 1998; Bennett, 2013). This divide between local and long-distance phonological patterns is natural and was recognized in Heinz (2010). We propose that such long-distance maps will be characterized by different subregular classes of functions, though the precise characterization of such classes remains for future work. As mentioned above, the underlying concepts of the subregular hierarchy provide an explicit scale for investigating the nature of complexity in phonology, and this paper will focus on its contributions to local phonology.

Having established the range of phonological maps that are ISL, we ask the question of why this property should characterize local phonology. Our answer comes from the fact that phonological grammars are learned. In other work we have proven that the ISL functions are learnable from positive data (Chandlee, 2014; Chandlee *et al.*, 2014; Jardine *et al.*, 2014), and we describe one such learning algorithm in this paper. Importantly, these algorithms can *only* learn ISL functions, even if the data they receive is consistent with a more complex hypothesis for the target map. In

other words, learners generalize a map from data in a particular way. In turn, this means that the resulting grammar is ISL *because of the way it was learned*. ena in the discussion section.

The remainder of the paper is organized as follows. Section 2 elaborates on the perspective of phonological processes as maps, and section 3 introduces the formalism of finite state transducers that will be used throughout the analyses. Section 4 defines the ISL functions. First we present the class of formal languages called Strictly Local languages and demonstrate how these can be used to model natural language phonotactics. Then we extend the notion of locality provided by the SL languages to maps by defining ISL functions and the particular type of finite state transducer that represents them. Section 5 demonstrates the exact range of phonological processes that have the property of Input Strict Locality, which includes substitution processes, as well as deletion, insertion, and synchronic metathesis. This section also quantifies the empirical coverage of ISL functions by presenting the results of a survey of the P-Base database of phonological patterns (v1.95 Mielke, 2008). Section 6 describes the utility of the ISL classification in the learning of such processes by presenting a learning algorithm that uses the defining property of ISL as an inductive principle or learning bias. Section 7 discusses the potential for extending the current analysis to non-ISL, long-distance phonological phenomena, as well as the implications of our results for phonological theory. Section 8 concludes.

2 Phonological Maps

In this section we reiterate the view, widely held in modern generative phonology, that posits a map (i.e., an infinite set of (input, output) ordered pairs) between abstract underlying mental representations (URs) and surface phonetic

representations (SRs). Such a map is posited by both rule-based and constraint-based theories, and this point of agreement allows us to study the properties of those maps independently of the particular grammar used to describe them.

To illustrate, consider the phonological phenomenon found in German, Russian, and many other languages of word-final voiced obstruents being pronounced as their voiceless counterparts. In a rule-based formalism (Harris, 1951; Chomsky and Halle, 1968), this fact might be represented with a rule like (1).

(1) $[-\text{sonorant}] \rightarrow [-\text{voice}] \; / \; \_\_$

This rule represents a phonological process by which word-final voiced obstruents are devoiced.[2] This exact same map can also be described in a constraint-based formalism (Legendre *et al.*, 1990; McCarthy, 2000; Prince and Smolensky, 2004; Pater, 2012), for example with the constraint ranking shown in (2).

(2) $^*[+\text{voice}, -\text{sonorant}]\# \gg \textsc{Ident}(\text{voice})$

The effect of this constraint ranking is that a candidate with a word-final voiceless obstruent will be preferred to one with a word-final voiced obstruent, even if the voiced obstruent is underlying. Other faithfulness constraints that could be violated to prevent $^*[+\text{voice}, -\text{sonorant}]\#$ sequences would also have to outrank $\textsc{Ident}(\text{voice})$ in order for devoicing (and not deletion, epenthesis, metathesis, etc.) to be the preferred repair.

The phenomenon captured by both (1) and (2) can also be represented as a set of URs paired with SRs, as in (3). If the UR has a word-final voiced obstruent, the SR ends with the corresponding voiceless obstruent. Otherwise, the strings are identical.

(3) {(baːd, baːt), (saːg, saːk), (alt, alt), (wald, walt), . . . }

The set in (3) is infinite in size, since the final devoicing generalization is defined for more than just the words in a single speaker's lexicon. It is defined for any *possible* word. With no principled upper bound on the length of words, the set of possible words for a given language is infinite.[3]

Our approach then is to identify the formal properties of these maps themselves, which are thus meaningful for, and compatible with, any theory that aims to describe the object represented in (3).[4] In particular, these properties can shed light on the kinds of such transformations that a given theory should permit. While most of our examples will thus correspond to singular processes, in §5.3 we do consider maps that are traditionally conceived of as containing more than a single process to emphasize that ISL is a property of an input-output map in general, not necessarily one that corresponds to an isolated process.

This approach of identifying properties of the map that are independent of the grammatical formalism does have precedent in the phonological literature. For example, Tesar (2008, 2012, 2014) designates as *output-driven* those phonological maps that meet the following condition: if some input A is mapped to some output B, then every input A′ which is more similar to B than A is also maps to B, assuming a similarity metric over the space of inputs and outputs. As Tesar makes clear, this property is independent of whether the map is described with an OT grammar or some other formalism.

To sum up, the object of inquiry at hand is the phonological map from UR to SR and its properties. In the next section, we show how tools from theoretical computer science allow us to examine these maps directly and identify properties of them that are independent of any particular grammatical formalism.

3 Computational Background

Here we introduce a standard formalism from computer science – finite state automata – which will be used in the computational analyses throughout the paper. We will assume familiarity with strings, concatenation, and sets, and will make use of notation like a^k to indicate a concatenated with itself k times (so $a^3 = aaa$). We provide only a brief introduction to finite state automata; for more details on them and their linguistic applications, see Roche and Schabes (1997), Hopcroft *et al.* (2000), Beesley and Karttunen (2003), Roark and Sproat (2007) and Hulden (2009a). Readers familiar with automata may skip this section.

Finite state automata include finite state acceptors (FSAs), which represent formal languages or stringsets, and finite state transducers (FSTs), which represent relations or maps. An example of an FSA is shown in Figure 2. FSAs include a set of states, represented in the figure with circles, and a set of transitions, represented in the figure with labeled arrows. Starting from the *initial state*, marked with a small incoming arrow (e.g., state 0 in Figure 2), a given input string is read one symbol at a time, and the FSA follows the transition whose label matches the current symbol being read. When the end of the input is reached, the string is *accepted* (i.e., it is in the language) if the state the FSA is in is a *final state*, marked with a double circle. If the state is not a final state, the string is rejected (i.e., it is not in the language).

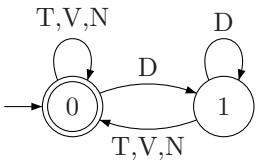

Figure 2: FSA for a language without word-final voiced obstruents (V=vowel, N=sonorant consonant, T=voiceless obstruent, D=voiced obstruent)

A given FSA is defined for a particular *alphabet* that includes the possible labels for its transitions. The alphabet of the FSA in Figure 2 is the set {D, T, V, N}. It is easy to see that any string that ends in a D will be rejected from the language this FSA represents. All other strings will be accepted. If we interpret D as a voiced obstruent, T as a voiceless obstruent, V as a vowel, and N as a sonorant consonant, then this FSA describes a language that bans voiced obstruents from word-final position.

FSTs, on the other hand, describe string-to-string relations. An example of an FST is shown in Figure 3. Like FSAs, FSTs read an input string one symbol at a time and proceed through the states according to the labeled transitions. The difference between an FSA and an FST is that the FST also produces an output string as it proceeds through the transitions. Thus the transition labels are of the form $a : x$, where a is the input symbol read in and x is the portion of the output string produced. Note that x is a string, so it contains zero or more symbols. Single-symbol transition labels indicate that the input and output are identical (e.g., $a = a : a$ in an FST).

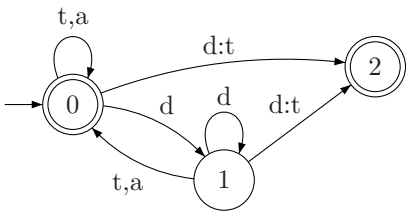

Figure 3: FST for final devoicing

As an example, consider the input string *tad*. Given this input, the FST in Figure 3 begins in state 0 and loops back to state 0 on the first two input symbols t and a, outputting the input unchanged. When it next reads in d, it has a choice. It

can either move to state 2 and output t, or it can move to state 1 and output d. For the map $tad \mapsto tat$ to be valid, there must be at least one path through the FST that reads *tad*, outputs *tat*, and ends in a final state. Such a path does exist in this case, and it is shown in (4). In (4), the top row shows the input symbols, the bottom row the corresponding output, and the center row the series of visited states.

(4)
	t		a		d	
0	⇒	0	⇒	0	⇒	2
	t		a		t	

Notice what happens with an input like *tada*. For the first two input symbols, t and a, the FST proceeds exactly as it did in the previous example. This time, however, when it faces the choice between moving to state 1 or 2 on d, it must choose state 1. This is because there are no outgoing transitions from state 2, so if it moves there it has no where to go when it reads the final a. Not being able to read the entire input would result in the map being rejected. So it moves to state 1, then back to 0 on a and correctly maps *tada* to *tada*. This derivation is shown in (5). These two examples show how the FST—just like the generative analyses in (1) and (2)—will only accept the mapping of d to t when that d is in final position.

(5)
	t		a		d		a	
0	⇒	0	⇒	0	⇒	1	⇒	0
	t		a		d		a	

Note that the FST in Figure 3 can map any possible string to an output, including strings like *ddd*, *aaaaaat*, and *dtdtdt*. As mentioned in the previous section, a given phonological map can be defined for all strings of any length over a given alphabet. The fact that no language is likely to have a UR for a lexical item like *ddd* is

independent of the linguistic generalization known as 'final devoicing'. The way the rule in (1) is defined it will apply to *ddd* just as it does to *dad*.[5] The same is true for the OT analysis of this generalization. In a sense, this idea has been axiomatized in OT: under the assumption of Richness of the Base (Prince and Smolensky, 2004) there are no constraints on possible input forms.

As stated in the introduction, SPE rules that cannot re-apply to the locus of their structural change can be modeled with FSTs and therefore describe regular relations (Johnson, 1972; Koskenniemi, 1983; Kaplan and Kay, 1994). And because regular relations are closed under composition (i.e., if the maps A $\mapsto$ B and B $\mapsto$ C are regular relations, then so is A $\mapsto$ C), an important consequence of these previous results is that the UR $\mapsto$ SR map of the entire phonological grammar is also regular. This is important because it has been widely recognized that language-specific, ordered SPE-style rewrite rules are sufficiently expressive to describe the phonology of any language. This does not mean that the grammar is the most elegant, or compact, or that it has any other desirable trait. It just means that the phonology of any language can be described in this manner. If this is correct, then 'regular' is a *universal* property of the UR $\mapsto$ SR map, and this fact holds true even if the grammar is instead described with a constraint-based formalism.[6]

Requiring phonological maps to be regular, however, does not appear to be a strong enough restriction, since many maps that can be described with regular relations are unattested and typologically odd. A further restriction to a *subregular* class known as the *subsequential functions* was apparently first suggested by Mohri (1997).[7] The subsequential functions are a proper subset of the regular relations; their respective FSTs, called subsequential FSTs (SFSTs), are restricted to being *deterministic*. This means that at any given state there is only one outgoing transition per input symbol in the alphabet. While less expressive than regular

relations, recent work suggests that subsequential functions are indeed sufficiently expressive to model a wide range of phonological processes (both local and long-distance) (see Chandlee *et al.* (2012); Gainor *et al.* (2012); Chandlee and Heinz (2012); Heinz and Lai (2013); Luo (2013); Payne (2013); Jardine (to appear)).

We take these previous results one step further by establishing a more restrictive class of functions that is a proper subset of both the regular relations and the subsequential functions: the Input Strictly Local functions.

4 Strict Locality

In this section we define Input Strictly Local functions. The formal definition of this class of functions (and the particular FSTs that describe them) will be presented below in §4.2. As a preliminary, however, in §4.1 we first present the well-studied class of formal languages called the Strictly Local languages (McNaughton and Papert, 1971; Rogers and Pullum, 2011; Rogers *et al.*, 2013), on which the ISL functions are based.

4.1 Strictly Local Languages

As discussed above, string-to-string relations (like phonological maps) can be represented with sets of string pairs. Formal languages, on the other hand, are simply sets of strings, and as such they can model markedness (or phonotactic) constraints, which dictate sets of well-formed words. Consider an example based on Hawaiian, which does not allow words that end in consonants or words with consonant clusters (Alexander, 1920).[8] These constraints can be expressed as follows: words cannot contain substrings of the form CC or C#, where C is a consonant and # is the word boundary. These constraints can also be represented by the infinite set of words that

do *not* violate them, some of which are shown in (6).

(6) {honalulu, hilo, maui, kauwai... }

Since the banned substrings needed to model this infinite set are both of length 2, the language is in fact an SL-2 language. We will elaborate on what this means throughout the section.

The SL languages have been argued to model all local phonotactic constraints, meaning those that ban finitely many contiguous substrings of bounded length (thus excluding action-at-a-distance effects) (Heinz, 2007, 2009, 2010). We review two different characterizations of the SL languages (McNaughton and Papert, 1971; Rogers and Pullum, 2011; Rogers *et al.*, 2013): one language-theoretic and one automata-theoretic. These characterizations will pave the way for our characterizations of the ISL functions, which will also be presented in both language- and automata-theoretic terms.

The language-theoretic characterization, shown in (7), is a property that Rogers and Pullum (2011) call the Suffix Substitution Closure (SSC):[9]

(7) An SL language is one in which there exists an integer k such that strings which share a suffix of length $k-1$ are extendable in exactly the same way. So if u_1x and v_1x are strings such that x contains $k-1$ symbols, and it is the case that suffixing a string u_2 onto u_1x results in a string in the language, then suffixing u_2 onto v_1x also results in a string in the language.

Essentially, this property says that if two strings in the language share a substring of length $k-1$, then they also have the same options for what comes after that substring. To see how this works, consider again the case of Hawaiian. We know that

both of the words CVCV and CVVV are acceptable. These words share several substrings of length $k-1$ (recall in this example $k=2$). For example, they share the substring V shown in boldface: C**V**CV, CV**V**V. We can parse each of these strings into three pieces, with the second piece being that common substring: C·**V**·CV, CV·**V**·V. The suffixes of these two strings following the common substring are CV and V, respectively. According to the SSC, if we 'flip' or substitute these two suffixes, the resulting strings will also be valid words in Hawaiian. This prediction holds: the resulting strings are C**V**V and CV**V**CV.

The strength of the SSC property is that we can repeat this experiment with *any* two strings in the language that share a substring of length $k-1$ and the results are *guaranteed* to also be in the language. If a single counter-example exists, then the language is not SL. The SSC thus provides an inference law which supports learning. If a learner knows that the target language is SL for some k and observes u_1xv_1 and u_2xv_2 with x being of length $k-1$, then the learner can safely conclude that u_1xv_2 and u_2xv_1 are also in the language.

For an example of a language that is not SL for any k, consider a language in which all words must have an even number of consonants. Let k be any integer greater than zero. It is easy to see that the strings CV^{k-1}C and CCV^{k-1} are both in this language. These strings share the substring V^{k-1}, so we can again parse them as follows: $\text{C}\cdot\text{V}^{k-1}\cdot\text{C}$ and $\text{CC}\cdot\text{V}^{k-1}\cdot\lambda$. (Note λ is the 'empty' string that contains no symbols.) Switching the suffixes gives us the strings CCV^{k-1}C and CV^{k-1}, which both contain an odd number of consonants and are therefore not in the language. Since the value of k in this example is arbitrary, we can conclude that for no possible value of k does the property in (7) hold for this language. It is therefore not an SL language.

The SL languages also have a finite state characterization, meaning they are the languages that can be described with a particular class of FSAs. Specifically, the SL

languages are those that can be described with FSAs in which the states correspond to all possible sequences from the alphabet of length $k-1$, and the transitions ensure that being in a given state is only possible if that state represents the most recently read $k-1$ symbols.

For example, an SL FSA describing the Hawaiian pattern is shown in Figure 4. Since in this case $k=2$, the states in Figure 4 represent all possible sequences from the alphabet {C, V} up to length $2-1$ or 1. Also note that whenever this FSA reads a C it moves to state C, and whenever it reads a V it goes to state V.

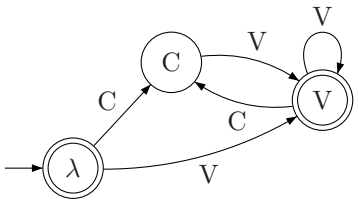

Figure 4: FSA for the Hawaiian SL-2 grammar

The available transitions between states represent the allowable sequences of length k; thus there is a transition from state C to state V because CV is permitted, and there is a loop at state V because VV is also permissible. There is no loop at state C, however, because CC is not permitted. And the C state is not a final state, because C# is also not allowed.

FSAs in general are not subject to these restrictions on the state set and transitions, which therefore delimit the set of languages SL FSAs can describe. To see this, consider the FSA in Figure 5, which describes the language in which all words must contain an even number of consonants. As we have already proven using the SSC, this language is not SL. Therefore it cannot be described with an FSA that has the properties of the Hawaiian FSA.

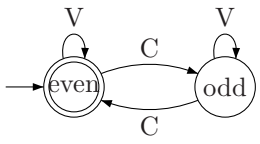

Figure 5: Non-SL FSA for the language in which all words have an even number of consonants

To see this, consider two strings, CC^{k-1} and C^{k-1}, this time for some arbitrary even value of k. We know the first string is in the language but the second is not. If we constructed an SL FSA for this language, both strings would end in state C^{k-1}, because by definition the FSA is always in the state that represents the last $k-1$ segments. Therefore, the FSA must either accept or reject both CC^{k-1} and C^{k-1} (depending on whether or not the state C^{k-1} is an accepting state). In other words, the FSA cannot distinguish between the well-formedness of the first string and the ill-formedness of the second. Again, since k is arbitrary, we conclude that the same problem will hold no matter what even value we assign to k. This example also proves that the SL languages are a *proper* subset of the regular languages (i.e., they are *subregular*), because this language can be described with an FSA (i.e., it is a regular language), just not an SL one.

The notion of Strict Locality is actually quite intuitive from a phonological standpoint: string well-formedness depends on the well-formedness of each of its substrings of some bounded length. The whole string is well-formed only if each of its substrings are. Therefore, when examining a particular substring, it is not necessary to remember information about the substrings that come before or after it. This is the essential 'local' quality of such phonological dependencies, and it has a clear interpretation as a short-term memory model. The k-value essentially parameterizes

the amount of available memory.

With the SL languages as a foundation, we are now ready to define the ISL functions by adapting the language- and automata-theoretic characterizations presented in this section from stringsets to string-to-string relations.

4.2 Input Strictly Local Functions

One of the main contributions of this paper is that Strict Locality is a computational property that not only characterizes markedness constraints which ban contiguous substrings; it also characterizes the maps that prevent those substrings from surfacing. What we mean by this is the UR↦SR maps that such processes represent are in fact Input Strictly Local functions. In this section we first define these functions in language-theoretic terms and then provide an automata-theoretic characterization that can be used to model those processes that have the ISL property. Readers are referred to Chandlee (2014) and Chandlee *et al.* (2014) for additional mathematical details of the definition and a proof that ISL FSTs correspond exactly to the ISL function class.

The defining property of ISL functions is similar in spirit to the SSC property of SL languages that was discussed in the previous section. Before presenting the actual definition, however, we must first introduce the concept of *tails*. The basic idea is that given a string x that is a prefix[10] of one of the strings in the domain of a function f, the tails of x are pairs of strings that correspond to the possible extensions of x and the contribution of that extension to the output string.

The formal definition of the tails of an input x with respect to a function f is shown in (8). To unpack this definition a bit: let $\texttt{lcp}(S)$ be the *longest common prefix* shared by a set of strings S, and let Σ^* be the set of all possible strings of finite length that can be formed using symbols from the alphabet Σ. So if we take all of the

input strings with x as a prefix (i.e., the set $x\Sigma^*$), and apply f to them all, the result will be another set of strings. Let u be the longest common prefix of this set. This `lcp`, u, thus represents the contribution to the output of x, regardless of what happens next. The set of tails, then, is the set of options for 'what happens next'.

(8) $\texttt{tails}_f(x) = \{(y, v) \mid f(xy) = uv \wedge u = \texttt{lcp}(f(x\Sigma^*))\}$

As an example, consider the alphabet $\Sigma = \{\text{D, T, V, N}\}$ and a function f that maps strings in which a T follows an N to strings in which that T is a D (all other strings are mapped to themselves). So f(VNT) = VND, f(NTD) = NDD, f(DVD) = DVD, etc. Now consider the input x = VN, which the function maps to the output VN. It is easy to see that any input that has VN as a prefix will be mapped to an output with VN as a prefix (e.g., f(VNN)=VNN, f(VNV)=VNV, etc.). In other words, VN is the longest common prefix of all outputs that have inputs that begin with VN.[11]

The remainder of the output string will differ depending on the next input symbol. Extending VN with D, V, or N will extend the output by D, V, and N, respectively. But extending it with T will extend the output by D. These pairings of 'input extension' and 'output extension' are members of the infinite set of tails (with respect to f) for VN, as shown in (9).

(9) $\texttt{tails}_f$(VN) = { (D,D), (V, V), (N, N), (T, D), (TV, DV), (VN, VN), (TN, DN) ...}

The set of tails for a given string is not necessarily unique to that string, meaning more than one string can have the same set of tails. We can group those strings that share a set of tails together into equivalence classes. In an important sense, the function treats all of the strings within an equivalence class in the same way. In other

words, the function disregards the actual differences among the strings within a class and just views them in terms of the class itself. In the present example, it should be clear that the set of tails of VN is also the set of tails of VNVN, TVN, DVN, NNN, DVDVDVN, etc. In fact, any string that ends in N has the set of tails in (9).

We are now ready to define an Input Strictly Local function. The essential property of these functions is that input strings that share a suffix of length $k-1$ also have the same set of tails. This is what we saw with the example above: strings that 'end in N' (or, share the suffix N) have the same tails. This function in fact corresponds to the process of post-nasal obstruent voicing, attested in languages like Quechua (Pater, 2004):

(10) Quechua Post-nasal Obstruent Voicing

a. /kampa/$\mapsto$[kamba], 'yours'

To correctly model this process, the function crucially needs to distinguish voiceless obstruents (or T) that follow nasals from those that do not. In other words, it needs to be 'on the lookout' for the sequence NT. Since this sequence is of length 2, the function is ISL for $k = 2$. As it reads an input like *kampa*, at the point it reads the nasal it will recognize that this string is in the class of strings with the set of tails in (9). Therefore when it next reads the voiceless obstruent it will know to map it to the voiced counterpart.

The definition of an ISL function is formalized below (note that $\texttt{Suff}^{k-1}(x)$ represents the unique suffix of x of length $k-1$ if the length of x is at least k; otherwise it is just x itself).

Definition 1 (Input Strictly Local Function)**.** *A function f is Input Strictly Local iff there is a k such that for all $u_1, u_2 \in \Sigma^*$, it is the case that if*

$\texttt{Suff}^{k-1}(u_1) = \texttt{Suff}^{k-1}(u_2)$ *then* $\texttt{tails}_f(u_1) = \texttt{tails}_f(u_2)$.

Like the SSC, this is a language-theoretic property in the sense that it is independent of any particular grammatical formalism.

The relevance of strings that share a set of tails is also evident in the FST representation of the function, shown in Figure 6.[12] It is clear in the figure that all input strings (of any length) that end in N will end in state N. The paths available from state N are therefore clearly the same for all of those strings. These 'available paths' are exactly the set of tails. Likewise, input strings that end in D will all lead to state D and therefore share a set of tails, input strings that end in T will lead to state T, and input strings that end in V will lead to state V.

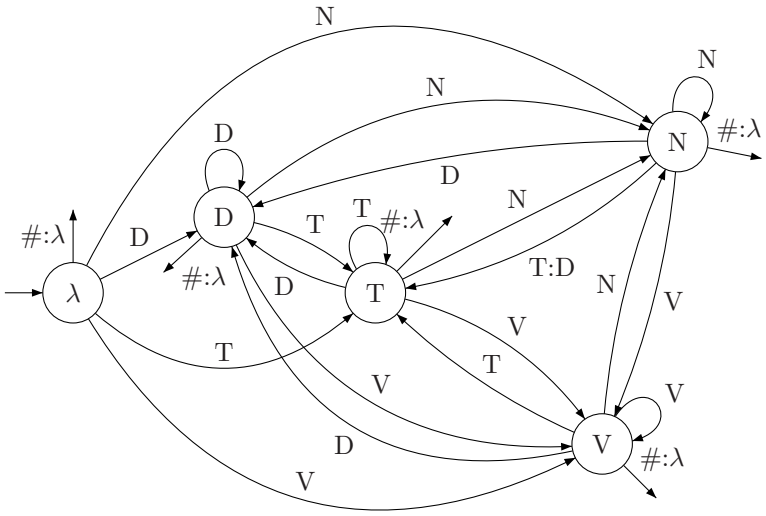

Figure 6: 2-ISL FST for the function that maps T to D if it follows N, with Σ = {D, T, N, V}

Like the SL FSAs described in the previous section, ISL FSTs have certain properties that restrict the kinds of string-to-string maps they can represent. What are these properties? The first is that they are *subsequential* FSTs. As mentioned in

§3, SFSTs describe subsequential functions, which are a proper subclass of the regular relations (Mohri, 1997). SFSTs are necessarily deterministic, and in addition they include a *final output function* that maps states to strings (Mohri, 1997). The (possibly empty) string mapped to a state is appended to the output when an input terminates in that state.[13] In Figure 6 and all subsequent FSTs in the paper, this final output function is represented with a transition leaving the state on the input # (the end of word symbol). Since this symbol is assumed to only appear at the end of a string, its transition does not lead to any other state (i.e., there is necessarily no more input).

The fact that ISL FSTs are SFSTs means that all ISL functions are also subsequential functions, but they are importantly a *proper* subclass of subsequential. ISL FSTs have two additional properties that restrict the range of mappings they can describe beyond just the subsequential ones: 1) the state set is the set of all possible sequences up to length $k-1$ for the given alphabet[14], and 2) the transitions are defined such that being in a state is only possible if that state represents the most recently read $k-1$ input symbols. These two properties should sound familiar, as they echo the properties of the SL FSAs that we introduced in the previous section. In this way, ISL FSTs combine the properties of subsequential transduction with the notion of Strict Locality from SL languages. The result is a type of SFST that can only describe a restrictive class of maps, specifically the ISL functions. As before, the k value can be interpreted as the amount of available short-term memory.

It follows, then, that a phonological map is ISL provided that the output that is written at any given point *only depends* on the current input symbol and the preceding $k-1$ symbols of the input string. Compare the ISL post-nasal obstruent voicing example above to a hypothetical version in which T's become D's only if they follow an even number of N's. So $f(\text{NNT}) = \text{NND}$ but $f(\text{NNNT}) = \text{NNNT}$. This

function is not ISL for any k (i.e., no matter what value we assign to k, the defining property of ISL functions will fail to hold). A simple argument proves this. For any even value assigned to k, the strings NN^k and N^k share a suffix of length $k-1$. But (T, T) is in the tails of NN^k and (T, D) is in the tails of N^k. Thus these strings share a suffix of length $k-1$ but do not have the same tails, contra to Definition 1. A similar argument would show the property doesn't hold for any odd k. This proves that such a function is not ISL for any k.

A (non-ISL) FST for this 'even-N' post-nasal obstruent voicing function is shown in Figure 7. Notice that any contiguous span of Ns of odd length will lead to state 1, while any contiguous span of Ns of even length will lead to state 2. Thus for an even k, NN^k would end in state 1 and N^k would end in state 2, and so the voicing of a following T would correctly occur in the latter but not the former case. An ISL FST, however, would force both NN^k and N^k to end in state N^{k-1}, such that the voicing must apply to either both strings or neither of them. The needed distinction would be lost.

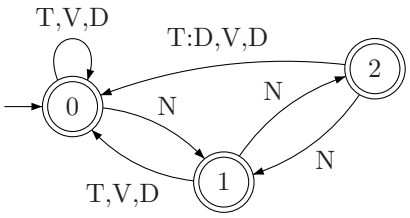

Figure 7: FST for the function that maps T to D if it follows an even number of Ns

To our knowledge, the process that the function in Figure 7 describes is unattested and phonologically odd. In this way, as a hypothesis of possible phonological maps that involve contiguous substrings, the ISL property draws a clear boundary with post-nasal obstruent voicing on one side and 'even-N' post-nasal obstruent voicing on

the other. In the next section we evaluate this hypothesis to study how well the ISL functions characterizes the range of attested phonological processes.

5 ISL Phonological Maps

In this section, we present several categories of phonological maps that can be modeled with ISL functions. As stated above, the unifying trait of such maps is that their target and triggering contexts form a contiguous substring of bounded length. We begin with the simple case of maps that correspond to an isolated process or rule, giving examples of substitution, insertion, deletion, and metathesis maps. We then give an example of a map that corresponds to multiple (potentially interacting) processes and show that it too is ISL. Lastly, we explain that iterative spreading processes are in fact not ISL, but they are amenable to an ISL-like analysis, with one important difference.

5.1 Maps corresponding to single rules

Similar to the approach of Kaplan and Kay (1994), we will begin by referring to a process with its rule-based representation, with the reminder that our result is not dependent on a rule-based formalism. Consider a rule of the form in (11).

(11) $A \rightarrow B \ / \ C \_\_ D$

When earlier phonologists thought about the map that such a rule represents, they realized that it very much matters how the rule applies. In Chomsky and Halle (1968), rules applied simultaneously; this means that all possible targets of the rule are identified in the input and then the output is generated by applying the rule to *all* of those targets. We return to directional application of rules later. For now, we

assume something like simultaneous application as the mode of application.

We see that input strings that contain a CAD sequence are going to be subject to the rule. This same format can and has been used to write rules for both local and long-distance processes. The difference between the two is the nature of the set of strings represented by CAD. Consider two examples of assimilation, one long-distance (from Kikongo (Meinof, 1932; Dereau, 1955; Webb, 1965; Ao, 1991; Odden, 1994; Hansson, 2001, 2010; Rose and Walker, 2004), shown in (12-a)) and one local (from Quechua, in (10) and repeated in(12-b)).

(12) a. Kikongo: /tunik-idi/ $\mapsto$ [tunik-ini], 'we ground'

b. Quechua: /kampa/ $\mapsto$ [kamba], 'yours'

The long-distance consonant agreement process in Kikongo nasalizes the voiced stop in the suffix if a nasal consonant precedes it anywhere in the word. In the rule for such a process, C needs to include strings of any length that start with a nasal. This in turn means that CAD is in fact an infinite set of strings. But in Quechua, the set of strings in C that trigger voicing of a following stop is just the set of nasals, and so CAD is finite. This crucial difference – whether the structural description of a rule is finite or infinite – leads us to the following statement of our typological claim: a phonological map described with a rule A $\rightarrow$ B / C $\_\_$ D that can be said to apply in a simultaneous fashion is an ISL function if CAD is a finite language. In the appendix we provide a procedure for constructing an ISL FST for a rule that meets this condition. The basic insight into the construction follows from the fact that k can be set to the length of the longest string in CAD.

We now illustrate this construction and what it means to be an ISL process by providing several example processes along with the ISL FSTs that describe them. We

begin by returning to the example of German final devoicing as an example of a substitution process. Figure 8 presents the ISL FST for the final devoicing map, for which $k = 2$.[15] Unlike the nasal assimilation FST in Figure 6, in this example the final output function comes into play. To illustrate its role in the computation of this map, a sample derivation for the input string *datad* is shown in (13).

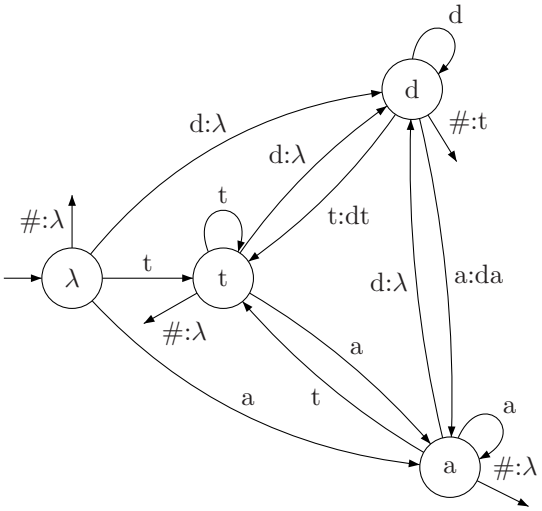

Figure 8: 2-ISL FST for final devoicing, with $\Sigma = \{\text{d, t, a}\}$

(13)

		d		a		t		a		d		#
λ	⇒	d	⇒	a	⇒	t	⇒	a	⇒	d		
		λ		da		t		a		λ		t

As shown in (13), upon reading the first d, the FST outputs the empty string λ and moves to state d. The output is empty because this d may or may not be output as t; that decision cannot be made until the FST determines whether this d is word-final. This option to 'postpone' output is only available if the information needed to decide on the output is a bounded number of symbols away. Since in this case $k = 2$, the needed information to decide on word-finalness is only $k - 1$ or 1 symbol away: if

there is a next symbol (other than #), then the obstruent is clearly not word-final.

The output cannot be postponed in this way in cases where the needed information is an unbounded number of symbols away. This is because the FST has to remember all the input it has seen since it began to postpone output; it does this by advancing to another state, one per input, and that state now serves as a record of the input that has been seen. If this has to go on for an unbounded amount of input, there is no way to know how many states are needed (i.e., the transducer could not be *finite* state).

In this way the machine makes concrete the notion that the output written at each step only depends on the current input symbol and the previous $k-1$ input symbols. The information needed to decide the output at each step is contained within this window.

Getting back to the example, once it is determined that the d is not word-final, the transition to state a outputs the string da. When it reads the second d, it again outputs λ, and this time, since it has reached the end of the input it takes the # transition and appends t to the output. Thus the $datad \mapsto datat$ map is correctly described.

It is easy to see that this process meets the criterion of CAD being a finite language. In the final devoicing rule in (14), the left context C is empty, the target A is always a single obstruent, and the right context D is just the end of word symbol.

(14) $[-\text{sonorant}] \rightarrow [-\text{voice}] \; / \; \_\_\#$

Thus there is a upper bound on the length of the strings in CAD, and that bound (which is also the k value of the function) is 2. As long as such a bound exists, the set of strings represented by CAD is finite.

Next we consider an example of insertion. As shown in (15), in Dutch ə is inserted between /l/ or /r/ and a [−coronal] consonant (Warner *et al.*, 2001):

(15) Dutch

a. /mɛlk/ ↦ [mɛləlk] 'milk'

b. /ʊɪlx/ ↦ [ʊɪləx] 'willow'

c. /hʏlp/ ↦ [hʏləp] 'help'

Figure 9 presents the ISL FST for this process. The alphabet includes $\Sigma = \{\text{L, K, ?}\}$, where L is a liquid, K is any [−coronal] consonant and (following Beesley and Karttunen (2003)) the symbol '?' represents any other segment in the Dutch inventory aside from L and K. From the L state, if the next symbol is K then it is outputted along with the epenthesized [ə].

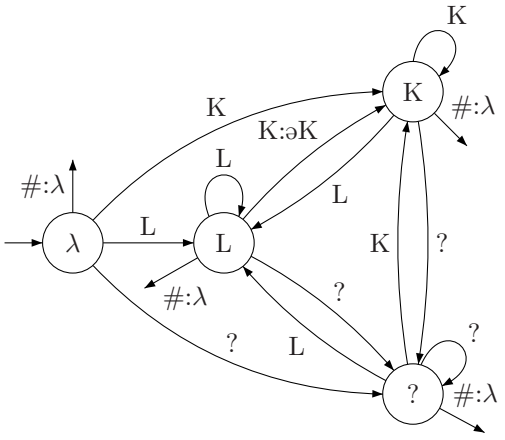

Figure 9: 2-ISL FST for Dutch ə-epenthesis, with $\Sigma = \{\text{L, K, ?}\}$

As for deletion, in Greek an interdental fricative is deleted before either /s/ or /θ/. This process is ISL for $k = 2$, and its ISL FST is shown in Figure 10. (Again, in the alphabet '?' is an abbreviation for all segments except for those that already have

transitions.)

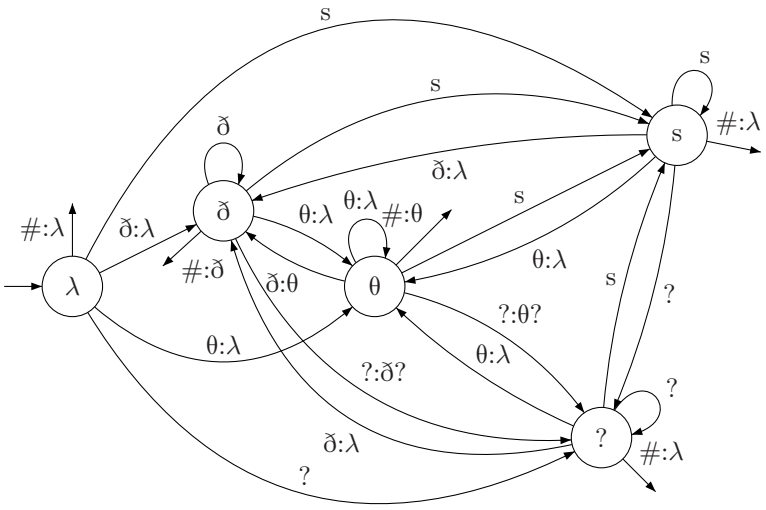

Figure 10: 2-ISL FST for Greek fricative deletion, with $\Sigma = \{s, ð, θ, ?\}$

Beyond substitution, insertion, and deletion, metathesis processes are also ISL (or at least all the synchronic ones surveyed, see below). Consider the example from Rotuman of word-final CV metathesis (Churchward, 1940), as in (16).

(16) Rotuman

hosa $\mapsto$ hoas 'flower'

This process is ISL for $k = 4$; an ISL FST that describes it is shown in Figure 11. Note that this FST has been minimized for readability, with the result that only the states corresponding to prefixes of VCV# are shown. Rotuman is a classic case of CV-metathesis that affects adjacent segments, but many metathesis patterns that have been labeled long-distance in the literature are also ISL. An example from Cuzco Quechua is shown in (17) (Davidson, 1977).

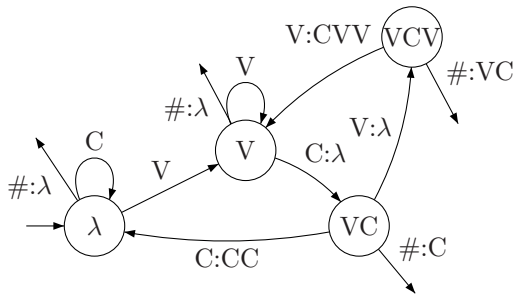

Figure 11: 4-ISL FST for Rotuman metathesis, with $\Sigma = \{V,C\}$

(17) Cuzco Quechua

yuraq $\mapsto$ ruyaq 'white'

Unlike Rotuman, in which the metathesizing segments are adjacent, in (17) *y* and *r* metathesize around an intervening vowel. Though this process is long-distance in the sense that it involves non-adjacent segments, it is still ISL because the number of intervening segments is limited to one and therefore CAD is a finite set of strings. Based on typological surveys of metathesis (e.g., Ultan, 1978; Blevins and Garrett, 1998, 2004; Buckley, 2011), it appears that all cases of synchronic metathesis are bounded in this way and are therefore ISL.[16]

In contrast, surveys of metathesis in the diachronic domain (e.g., Ultan, 1978; Blevins and Garrett, 1998, 2004; Buckley, 2011) reveal purported cases of unbounded phonological movement, such as the liquid displacement shown in (18), from a South Italian dialect of Greek (Rohlfs, 1950).[17] Similar processes are reported diachronically in Spanish (Lipski, 1992), Luchonnais Gascon (Grammont, 1905-1906; Dumenil, 1987), and Italian (Vennemann, 1988), among other languages.

(18) Classical > South Italian Greek

a. kopros > kropo 'dung'

b. gambros > grambo 'son-in-law'

c. kapistrion > krapisti 'halter'

In (18), a liquid in a non-initial syllable moves to the initial onset. If this movement is truly unbounded, meaning it takes place no matter how far into the word the liquid appears, then it is not ISL. However, in many cases the description of the process and/or the provided data suggest that the movement only spans one or two syllables, in which case it is in fact bounded and therefore ISL. Whether or not displacement is ISL, and, if not, why long-distance movement is limited to the diachronic domain are certainly interesting questions. But answering them is not crucial for our purposes. As a diachronic phenomenon, displacement falls beyond the scope of our goal of identifying the computational properties of synchronic phonological grammars.

5.2 Survey of P-base

Now that we have established what it means for a phonological map to be ISL, it is natural to question how much of phonology has this property. As one piece of evidence, we manually reviewed the approximately 5500 phonological patterns (from over 500 languages) reported in the P-Base database (v1.95 Mielke, 2008) and determined that over 95% of these patterns are ISL.

We of course acknowledge that the P-Base database is not (and was not intended to be) representative of the cross-linguistic distribution of local versus long-distance processes. Nonetheless, as it is the most comprehensive collection of processes of which we are aware, we felt it necessary to survey as evidence that the ISL property extends beyond the handful of examples presented in this paper. We also believe this result lends weight to our claim that computational Strict Locality is the right way to

think about locality in phonology.

Our impression is that the structural description (and hence the k-value) for most of these maps is 2 or 3, though there are many examples where the k value is 1, 4 and 5 and even a few cases where it appears to be as large as 7 or 8. These numbers reinforce the idea that the k-value ought to be interpreted as a kind of short-term memory. They remind us of Miller's Law, which asserts that the number of objects an average person can hold in working memory is about seven "plus or minus two" (Miller, 1956). It follows that this memory limitation sets a maximum value on k, with smaller numbers being more common simply because they place a smaller load on working memory.

The few cases that are not ISL for any k are primarily examples of long-distance harmony and disharmony, which we return to in the discussion section.

5.3 Maps corresponding to multiple rules

While all of the examples given thus far are of ISL functions that correspond to a single rule/process, ISL functions in general are not limited in this way. To illustrate, consider the following two phonological processes: post-nasal obstruent voicing and nasal place assimilation. These two generalizations are schematized in (19) and (20), respectively, with SPE-style rules.

(19) $[-\text{sonorant}] \rightarrow [+\text{voice}] \;/\; [+\text{nasal}] \_\_$

(20) $[+\text{nasal}] \rightarrow [\alpha \text{ place}] \;/\; \_\_[\alpha \text{ place}, -\text{sonorant}]$

Figure 12 presents a 2-ISL FST that describes both processes. For brevity, the figure assumes a single underlying nasal /n/ and obstruent /p/. All other sounds are subsumed under the ? symbol, whose states are collapsed in the figure along with the

start state for readability. This example shows that a single ISL map can represent multiple processes.

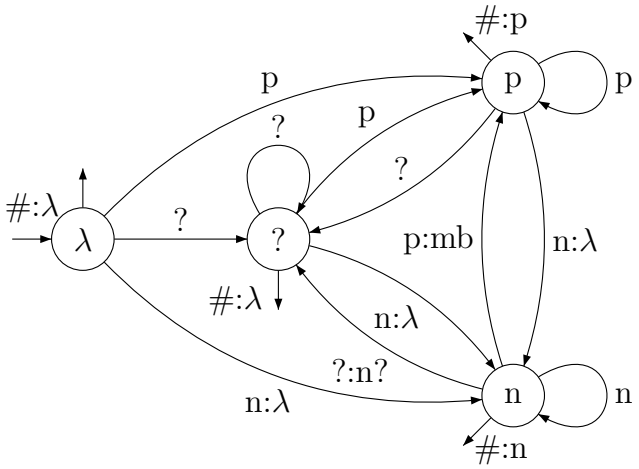

Figure 12: 2-ISL FST representing both post-nasal obstruent voicing and nasal place assimilation, with $\Sigma = \{p, n, ?\}$

The kinds of maps that are describable as the interaction of two or more processes is not limited to non-interacting processes like the example above. Chandlee *et al.* (2015b) demonstrates that the ISL property can also hold of interacting maps, particularly those traditionally described as opaque. Here we give one example of a simple counterfeeding chain: front vowel lowering before rhotics in Danish (Trubetzkoy, 1969; Hyman, 1975; Lundskaer-Nielsen and Holmes, 2011). Traditionally, this is described as two rules in a counterfeeding order: the first rule lowers /e/ to [ɛ] before /r/ and the second lowers /i/ to [e] before /r/. Figure 13 shows that this map is also 2-ISL.

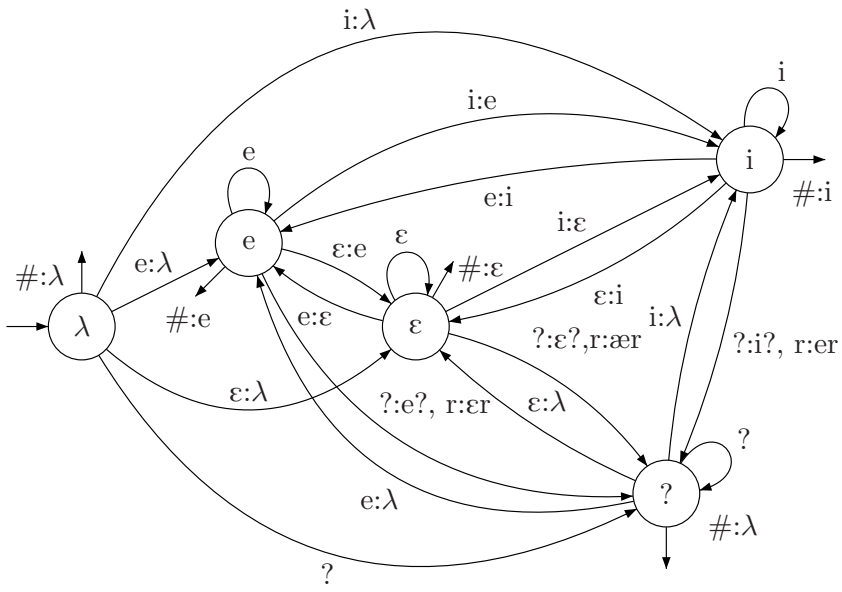

Figure 13: 2-ISL FST for Danish front vowel lowering (commonly understood as a case of counterfeeding), with $\Sigma = \{e, \varepsilon, i, ?\}$

These examples make clear that ISL maps should not be thought of as corresponding to singular processes, but in fact ought to be thought of as a property holding (or not) of an entire phonological map. This is not especially surprising. Each state in these FSTs represent an environment, and the transitions that leave each state can be thought of as an input/output relation w.r.t. that environment. Since there are multiple states in the machine, multiple environments can be represented, and hence multiple processes can be described.

It is beyond the scope of the current work to assess whether every known case of interacting processes is ISL or not, though we do acknowledge the importance and interest of this question, especially with respect to maps that have been called opaque (Baković, 2007).

An important related question is whether the ISL class is closed under composition. If so, then the functional composition of two ISL maps is necessarily ISL too. However, we currently cannot prove this result, nor do we possess a convincing counterexample at present, and so this aspect of the ISL class remains an open question.

5.4 Spreading

It was mentioned that an additional criterion for a rule to describe an ISL map is that the rule apply in a simultaneous fashion. The alternative is iterative application, in which the input is scanned from left-to-right (or from right-to-left) and the rule applies to each target as it is identified, with future applications being assessed based on the output of previous applications. The difference between these modes of application is revealed in cases where one application of the rule creates or removes what would have been 1) an additional target or 2) the context for an additional target.

In all of the examples we gave previously, whether the rule was applied simultaneously or iteratively the output would be the same. But in other cases, the different modes of application crucially create different maps (see Kenstowicz and Kisseberth, 1977; Kaplan and Kay, 1994; Hulden, 2009a).

Consider the progressive and regressive nasal harmony rules in (21) and (22), respectively.

(21) $[+\text{son}] \rightarrow [+\text{nasal}] \;/\; [+\text{nasal}]\ \_\_$

(22) $[+\text{son}] \rightarrow [+\text{nasal}] \;/\; \_\_\ [+\text{nasal}]$

An example of progressive spreading, from Johore Malay (Onn, 1980), is shown in

(23).

(23) Johore Malay

/pəŋawasan/ ↦ [pəŋãw̃ãsan] ‘supervision’

Assuming an alphabet of $\Sigma = \{\text{N, D, V}\}$, where N is a nasalized segment, D is an obstruent, and V is an oral vowel or non-nasal sonorant consonant, consider the input DVVNVV. Applying (21) and (22) to this string simultaneously would give the outputs DVVN$\tilde{\text{V}}$V and DV$\tilde{\text{V}}$NVV, respectively. If those outputs are correct, then the process is 2-ISL. But if rules (21) and (22) apply iteratively rather than simultaneously, leading to the outputs DVVN$\tilde{\text{V}}\tilde{\text{V}}$ and D$\tilde{\text{V}}\tilde{\text{V}}$NVV, respectively, then the maps are not 2-ISL. This is because by definition an ISL function can only pay attention to the input string, which is insufficient for iterative rules because they create triggers in the *output* and those triggers will be ignored by an ISL function, which only pays attention to the *input*.

More generally, it can be shown that the map described with the rule in (21) applying left-to-right is not ISL for any k. The argument follows from the rule in (24), in which a [+son] segment is nasalized after a [+nasal] segment and a sequence of as few as zero V’s and as many as k V’s.

(24) $[\text{+son}] \rightarrow [\text{+nasal}] \;/\; [\text{+nasal}]\; \text{V}_0^k \;\_\_$

This rule would apply to the final vowel of the input NV^{k}V but not to the final vowel of the input NV^{k+1}V, which means the prefixes NV^k and NV^{k+1} have different tails even though they share the same $k-1$ suffix (i.e., (V, $\tilde{\text{V}}$) is in the tails of the former while (V, V) is in the tails of the latter). Thus, such iterative spreading processes are not ISL for any k (since once again k in the example is arbitrary). A similar

argument shows (22) is not ISL for any k.

Though spreading is not ISL, we share the intuition with most phonologists that maps for iterative spreading ought to be construed as local. Indeed, in rule-based representations of these maps, the structural descriptions are finite, even if the rules do not apply simultaneously. This intuition is captured by a comparable class of functions we call *Output Strictly Local* (OSL) functions. The difference between the ISL and OSL classes lies in what strings share a suffix of length $k-1$. In an ISL function, two *inputs* with the same suffix of length $k-1$ have the same set of tails. In an OSL function, two inputs have the same set of tails if their respective *output strings* share a suffix of length $k-1$. Because the OSL functions pay attention to the recent output instead of the input, they are able to model spreading processes like in Johore Malay. Related discussion of the role of output in directional application can be found in Kaplan and Kay (1994) and Hulden (2009a).

OSL functions have been studied in detail by Chandlee *et al.* (2015a). We summarize their main results here. Like ISL functions, there is a class of subsequential FSTs that correspond exactly to OSL functions, which we call OSL FSTs. Figure 14 shows the OSL FST for progressive nasal spreading, like that found in Johore Malay. The most important difference to note between ISL and OSL FSTs is that the states represent the last $k-1$ *output* symbols written (as opposed to input symbols read). Hence in Figure 14, the transition leaving N and reading V goes to state $\tilde{\text{V}}$, not V. Similarly, from state $\tilde{\text{V}}$, there is a self-loop reading V and writing $\tilde{\text{V}}$.[18]

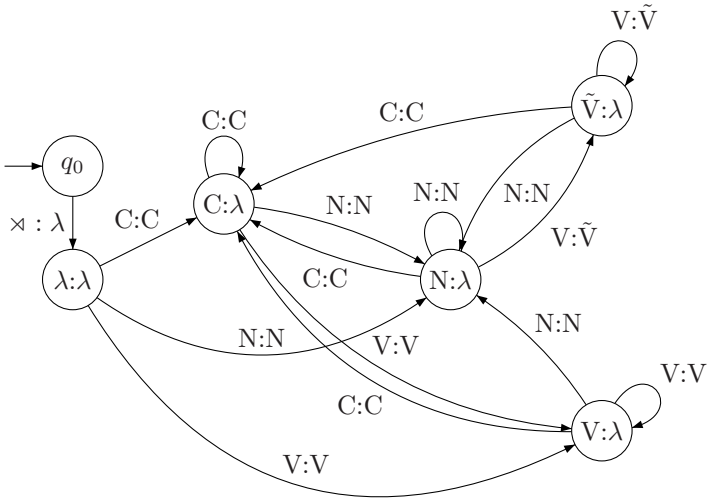

Figure 14: 2-OSL FST for progressive nasal spreading (as in Johore Malay), with $\Gamma = \{\text{C, N, V, } \tilde{\text{V}}\}$

Unlike ISL functions, OSL functions come in two varieties: left and right. Progressive spreading like in (21) can be characterized by OSL FSTs that read input strings left-to-right and regressive spreading like in (22) can be characterized by OSL FSTs that read input strings right-to-left. So the OSL class is in fact two intersecting but distinct subclasses. (This distinction is also true of the subsequential functions, which likewise consist of two classes called left subsequential and right subsequential.[19]) Figure 15 summarizes the relationships among the regular relations, the left and right subsequential functions, the ISL functions, and the left and right OSL functions. See Chandlee *et al.* (2014) and Chandlee *et al.* (2015a) for formal proofs of these relationships among the classes. While the focus of the current paper is on the ISL functions, Chandlee *et al.* (2015a) provide a formal definition of the OSL classes, their FST characterization, and a learning algorithm for them.

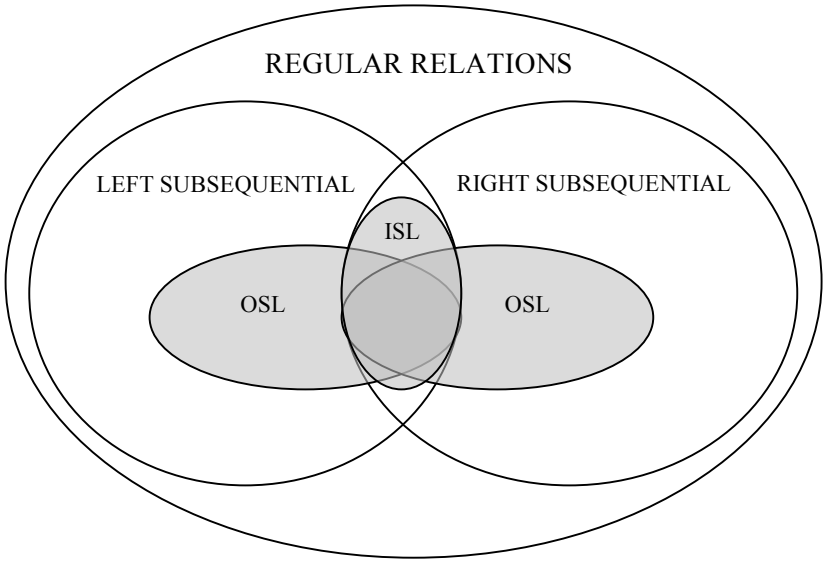

Figure 15: Relationships among ISL, OSL, subsequential, and regular

To summarize, we have established that processes describable with rules of the form A $\rightarrow$ B / C __ D (where CAD is finite) that apply in a simultaneous fashion are ISL, and processes describable with rules of the form A $\rightarrow$ B / C __ (where CA is finite) or A $\rightarrow$ B / __ D (where AD is finite) that apply in a left-to-right or right-to-left fashion are Left OSL or Right OSL, respectively. Interestingly, iterative rules with two-sided contexts are neither Left nor Right OSL. We discuss such maps in §7.1.

6 ISL and Phonological Learning

Having now established the range of phonological maps that have the ISL property, we now turn to the question of *why* that should be the case. What factors and/or considerations lead to local phonological maps being restricted in this way?

We propose an answer from the perspective of how phonology is learned. We assume a modular approach to phonological learning, in which different classes of

patterns (ISL, OSL, long-distance) are potentially learned by different mechanisms (cf. Heinz (2010)). Thinking for now just about the class of ISL phonological maps we propose the following account of how learning influences typology.

When faced with a finite amount of positive data, the phonological learner must make decisions in order to generalize from that data to a grammar. If those decisions are based on the memory-limiting property of Input Strict Locality, then the resulting grammars will be Input Strictly Local. Put another way, if the learner were free to posit any logically possible map that is consistent with the data it has seen so far, we would likewise expect to see a greater range of processes, some ISL, some not. The fact that we do *not* see such a range suggests the alternative: that the learner is *not* free to consider any map but rather restricts its hypotheses to those with the ISL property. Those without the property will therefore not find their way into the grammar.

The following background helps make this position clearer. The class of regular relations is not identifiable in the limit from positive data (Gold, 1967), but certain *subclasses* of the regular relations are. The ISL functions are one such class. The subsequential functions are another (Mohri, 1997). The latter class is learnable by the Onward Subsequential Transducer Inference Algorithm (OSTIA) (Oncina *et al.*, 1993).[20] As a proper subset of the subsequential functions, the ISL functions can be learned by OSTIA too. But as a hypothesis of how humans learn phonology, OSTIA is wanting. Empirical investigations have suggested that it is insufficient for learning phonology in practice (Gildea and Jurafsky, 1996), though modifications based on established phonological principles improve the situation (we say more about this below). But an important reason OSTIA is insufficient for phonology is that if humans learned phonology in the manner it suggests then we expect phonological maps to be found all across the subsequential region. This appears contrary to the

facts (recall that the 'even-N' post-nasal obstruent voicing function shown in Figure 7 is subsequential). Instead what we find are well-structured subclasses of subsequential, which so far include ISL and Left and Right OSL. The class for long-distance phenomena remains to be defined, but we have reason to believe it too will be a subclass of subsequential (see §7.1 below).

A reviewer proposes an alternative explanation for the prevalence of ISL maps in phonology, suggesting that phonetic factors may more strongly favor local dependencies over long-distance ones and therefore the fact that so much of phonology is ISL may have nothing to do with learning. We offer two responses to this proposal. One, we are not in fact arguing that learnability considerations explain why local processes are more common than long-distance ones. Our goal is to show that many phonological maps are local in the mathematical sense we have introduced and advocated for here, and that this has potentially significant implications for understanding the range of typological variation and how phonological grammars are acquired.

Second, we not only acknowledge the possibility that a phonetic-based explanation can equally explain the property of Strict Locality, but we would welcome such a theory. There is no reason the Strictly Local account offered here cannot be unified with a phonetic theory, provided it offers at least the same level of precision offered here. We see phonetic and computational approaches as working in tandem, not in opposition, towards the common goal of delimiting the set of possible phonological maps. It may be that phonetic accounts can and will ultimately subsume the insights offered by computational theories, but at this time we believe both are contributing valuable pieces of a much larger puzzle.

6.1 Learning Algorithm: ISLFLA

In the remainder of this section we present a learning algorithm for ISL functions that uses the class's defining property (i.e., inputs that have the same suffix of length $k-1$ also have the same tails) as an inductive principle to generalize the function from positive examples. The learner first creates a finite state representation of its given data and ultimately outputs an ISL FST that describes the function that the data samples. It generalizes using the technique of state merging, which is commonly used in grammatical inference; see Angluin (1982), Oncina *et al.* (1993), Heinz (2009), and de la Higuera (2010) for other examples and Heinz *et al.* (2015) for an overview of state-merging.

To demonstrate how state merging works, first we consider a learner of languages (i.e., sets of strings) that is given a single example from the target language, the word *aa*. A finite state representation of this single-word data set is shown in Figure 16. This representation is called a *prefix tree*, because each state corresponds to a prefix of a string in the data set. Note that the prefix tree is an FSA that accepts exactly one word: the word *aa* (i.e., it accepts the finite language $\{aa\}$). If the learner merges states 1 and 2 in this prefix tree while preserving their transitions, the result is the FSA shown on the right of Figure 16. This FSA accepts more than $\{aa\}$; it accepts all strings of at least one *a* (i.e., aa^*). In this way, state merging generalizes an infinite language from a finite data set.

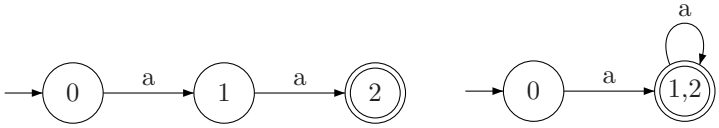

Figure 16: Prefix tree for *aa* (left) and FSA after merging states (right)

Of course, we want a learner to do more than just generalize: we want it to converge on the *correct* generalization. So if in the previous example the target language is not in fact aa^*, then whatever state merging criterion led to the merging of states 1 and 2 was incorrect. In this way, the choice of which states to merge in the prefix tree determines the kind of generalizations that can be learned. We want the learner to employ the state merging strategy that leads it to the intended target.

The learner for ISL functions, called the ISL Function Learning Algorithm (or ISLFLA), is given k and a set of (UR, SR) string pairs that sample the target ISL function. Similar to the toy example above, the ISLFLA proceeds through the two stages of 1) building a prefix tree and 2) merging states. Since the data set consists of string pairs rather than single strings, the prefix tree is an FST, not an FSA like in Figure 16. The states of this prefix tree transducer are the prefixes of the input strings of the data set, and it can only provide the correct output for the input strings in that set. To generalize, the learner merges states in the prefix tree transducer until it converges on an FST that describes the function that the data set samples.

To learn k-ISL functions – and, importantly, *only* k-ISL functions – the learner employs a state merging criterion that directly reflects the character of k-ISL FSTs. Specifically, it merges all and only those states that share a suffix of length $k-1$. As a result, when all of the state merging is complete the FST that remains will have states corresponding to all possible sequences from the alphabet of length $k-1$ (i.e., it will have a state set that meets the definition of an ISL FST). This in turn means that only those maps that can be described with FSTs of this form fall within the hypothesis space of this learner. In this way, the defining property of the target map both *limits* and *structures* the hypothesis space. The learner can converge on its target by zeroing in on the crucial information: states with the same suffix of length $k-1$ *must* have the same tails and therefore are merged.

A formal definition of the ISLFLA and a proof that it successfully learns any k-ISL function can be found in Chandlee *et al.* (2014). It is also proven there that the algorithm has polynomial time and data complexity. The bounds on complexity are in fact quadratic. A different learning algorithm with even tighter bounds on time and data complexity that can also learn the class of k-ISL functions is presented in Jardine *et al.* (2014). For an algorithm that learns k-OSL functions and proofs of its correctness and efficiency, see Chandlee *et al.* (2015a).

We acknowledge some limitations to the ISLFLA as a model of phonological learning. As already mentioned, the data required by the ISLFLA is a set of (UR, SR) pairs, but ultimately we would like to demonstrate how the computational properties we are defining can be further used to learn the URs from the SRs. Also, the ISLFLA only addresses the contribution ISL makes to learning in isolation of other relevant factors, such as phonological features (Albright, 2009) or phonetic factors (Wilson, 2006). In our estimate, the ISL functions presented here speak to – in Gildea and Jurafsky (1996)'s terms – the learning biases of 'context' and 'faithfulness', but not 'community'. Incorporating 'community' into the present algorithms would equip learners with knowledge of what types of changes are likely to affect certain segments as well as the tendency for segments within a natural class to be treated in like manner.

Lastly, readers may question the learning proposal given that 1) the learner is given the value of k rather than learning it, and 2) the learner only targets isolated processes rather than a complete phonological grammar. We will address these concerns in turn. First, the given k-value can be understood simply from the perspective of working memory, as mentioned earlier. So it does not need to be learned; it is given as part of the general cognitive architecture. With respect to finding the minimum k for a *given* phonological map, there are existing procedures

that can be used to decide this k based on the facts that subsequential transducers have canonical forms. This allows one to decide whether two subsequential functions such as two ISL functions with different k values are equivalent or not.

As for the second concern, the ISLFLA does not target single processes. It targets k-ISL functions, which as explained in §5 can represent maps consisting of multiple phonological processes. As the ISLFLA is proven to learn any k-ISL function, it will also learn such maps, including ones thought of as opaque. It will learn any phonological map that is k-ISL, regardless of how many component generalizations it contains. Of course, as already acknowledged, it is not the case that all complete grammars will amount to ISL maps – for example, those that include iterative spreading, as discussed above, or those with long-distance maps, as we discuss in the next section. Nonetheless, we argue that the theoretical learning results made possible by the ISL property warrant further exploration for the role it plays in both phonological description and phonological learning.

7 Discussion

In this section we address long-distance phonological maps, as well as the larger implications of the contributions of this paper and several important directions for future work.

7.1 Non-SL Processes

Non-ISL/OSL maps fall into two main categories: iterative/spreading processes with two-sided contexts and long-distance processes.

Consider first the (optional) ə-deletion process in French shown as a rule in (25) and exemplified in (26) (Dell, 1973, 1980, 1985; Noske, 1993).

(25) ə → ∅ / VC__CV

(26) tu devenais, 'you became'

a. /ty dəvənɛ/ ↦ [ty dvənɛ]

b. /ty dəvənɛ/ ↦ [ty dəvnɛ]

c. /ty dəvənɛ/ ↦ *[ty dvnɛ]

Applying (25) to the input /ty dəvənɛ/ in a left-to-right manner gives the map in (26-a), while applying it right-to-left gives the map in (26-b). Either of these maps are possible. The third map in (26-c), which corresponds to simultaneous application, is unattested. Thus the process is not ISL. Our discussion in §5.4 above suggests that the rule is iterative and therefore OSL, but in fact maps described by iterative rules with two-sided contexts are not OSL (Chandlee *et al.*, 2015a).

To model a map such as this one requires an additional class that combines knowledge of the recent input *and* output when deciding what to output. Such maps would allow a limited amount of 'look ahead' on the input side (like ISL) but also allow information on what has already been output (like OSL) to determine how to write the next portion of output. The technical details of this class, and how such functions can be learned, are the focus of current work. In addition, the need for such a class raises the interesting empirical question of how many processes of this kind exist and what (if any) additional factors distinguish them.

The other category of phonological maps that we leave for future work are long-distance processes in which an unbounded amount of material potentially intervenes between the target and trigger. One example of such a map was already discussed: the Kikongo nasal assimilation shown in (12-a) above. Another category that meets this criterion is vowel harmony. In a language without transparent vowels,

vowel harmony can be described with one of the rules in (27) and (28), where F is the harmonizing feature. The rules in (27-a) and (28-a) are for progressive harmony and the rules in (27-b) and (28-b) are for regressive harmony.

(27) a. $[+\text{voc}] \rightarrow [+\text{F}] \ / \ [+\text{voc}, +\text{F}]\ (\text{C})(\text{C})\_\_$

b. $[+\text{voc}] \rightarrow [+\text{F}] \ / \ \_\_(\text{C})(\text{C})[+\text{voc}, +\text{F}]$

(28) a. $[+\text{voc}] \rightarrow [+\text{F}] \ / \ [+\text{voc}, +\text{F}]\ \text{C}_0\ \_\_$

b. $[+\text{voc}] \rightarrow [+\text{F}] \ / \ \_\_\text{C}_0\ [+\text{voc}, +\text{F}]$

The rules in (27) assume a language that only allows single consonant codas and onsets, though complex codas and onsets could be accommodated by specifying additional optional consonants in the context. Importantly, though, there is some upper bound on the number of segments that intervene between the two harmonizing vowels. In contrast, the rules in (28) include an unspecified number of intervening consonants, represented with C_0. This difference is crucial, since the processes represented by (27) are OSL while those in (28) are not.

So the question is which description is the correct analysis of vowel harmony, meaning which better reflects the generalization learned by speakers of a language with vowel harmony? This question is not easily answered, but future experimental work may provide evidence. For example, it could be determined experimentally whether speakers of a language with vowel harmony apply the harmony process productively in nonce words which include more intervening consonants between vowels than is allowed by the syllable structure of the language. So if their language includes (27-a), they would be expected to apply harmony to a $\text{V}_{+\alpha}\text{CCV}_{-\alpha}$ sequence but not to a $\text{V}_{+\alpha}\text{CCCV}_{-\alpha}$ sequence.

What about a language with transparent vowels? In this case the distinction

between (27) and (28) is not relevant, since regardless of how many consonants intervene there would be no limit on how many transparent vowels can appear between the two harmonizing vowels. This type of vowel harmony is therefore not Strictly Local under any interpretation.

Our expectation for these non-SL phonological maps is that they instead belong to additional, as yet undefined, classes of subregular relations. Many processes that are not ISL or OSL are still subsequential (see Chandlee *et al.* (2012); Gainor *et al.* (2012); Chandlee and Heinz (2012); Heinz and Lai (2013); Luo (2013); Payne (2013); Jardine (to appear)). As mentioned earlier, the SL languages that form the basis for the ISL functions are just one region of the subregular hierarchy of formal languages (McNaughton and Papert, 1971; Rogers and Pullum, 2011; Rogers *et al.*, 2013) (see Figure 1). Other regions of this hierarchy have been shown to model long-distance phonotactics (Heinz, 2010; Heinz *et al.*, 2011), which suggests that the corresponding long-distance processes that enforce such phonotactic patterns (e.g., harmony, agreement, dissimilation) will likewise be model-able with functional counterparts to the subregular languages that describe their respective phonotactic constraints. Thus the primary goals of our current and ongoing research program include 1) defining these functional classes for long-distance processes, 2) establishing their respective learning algorithms, and 3) figuring out how these distinct subregular classes interact in the phonological grammar and in phonological learning.

7.2 Implications for Phonological Theories

Finally, in this section we address the relevance of the results presented in this paper for contemporary theories of generative phonology. We have provided a significant amount of evidence for the strong hypothesis that phonological processes with local triggers are Input Strictly Local functions. The question then becomes: for a given

phonological theory, what mechanism exists that guarantees that individual phonological processes with local triggers are Input Strictly Local?

For SPE-style rules that are used for segmental phonology, one answer was provided in Section 5: if the structural description of a rule is of bounded length and the rule applies simultaneously, then the rule is Input Strictly Local. If it were the case that *all* attested processes met this criterion, then the effect would be to ban certain notational devices from SPE-style rules, particularly those that allow for an unbounded amount of material in the structural description of the rule, such as the '*' notation (see Anderson, 1974), the C_0 notation, and variables like X.

However – as discussed above – there do exist processes in which a potentially unbounded amount of material intervenes between the target and trigger, and so a complete ban on such notations is unrealistic. Instead, the goal is to restrict the featural content and/or amount of material that such variables can refer to. Previous attempts at identifying these precise restrictions include Jensen (1974) and Halle (1975), who define the *relevant* segments of a process as a way to limit what essential variables like X can refer to (and therefore do away with subscript notation).[21] Odden (1994) also eliminates essential variables by restating rules in terms of feature geometry representations to identify the featural content of the segments participating in the process and parameterizing the amount of intervening material in terms of the level at which those segments must be adjacent (root, syllables, etc.).

Empirically, the question is which of these proposals (or others) is sufficient for the attested range of long-distance processes. In the context of the current research program, the question is, once a certain restriction on a variable like X is assumed, what class of (subregular) functions does the rule correspond to? In other words, once we restrict a variable in a particular way, what is the exact consequence for computational complexity? Identifying those restrictions on variables that allow for

both desirable computational restrictions while still being descriptively adequate could lead to valuable insights into how such processes are learned as well as the nature of long-distance phonological phenomena.

Turning now to the case of classic Optimality Theory, we see at present no simple way to account for the fact that phonological processes with local triggers can be modeled with ISL functions. Both Riggle (2004) and Gerdemann and Hulden (2012) provide illustrative examples of how *non-regular* mappings can be obtained through the interaction of simple markedness constraints and standard faithfulness constraints. In Gerdemann and Hulden (2012)'s case, the markedness constraints in fact are Strictly 2-Local, and the faithfulness constraints are canonical Dep and Max constraints. They show that the ranking Ident, Dep $\gg$ *ab $\gg$ Max leads to the map in (29).

(29) a. $a^n b^m \mapsto a^n$, if $m < n$

b. $a^n b^m \mapsto b^m$, if $n < m$

Such a map is non-regular, and therefore neither ISL nor OSL. It is easy to see that the information needed to determine the output cannot fit inside a window of size k. This shows concretely how optimization can lead to complex mappings and cannot easily be constrained by limiting the constraints to simple ones.

One reason for this is that constraint violations are counted in OT and there is no limit to the counting. Karttunen (1998) and Frank and Satta (1998) discuss how, in order to be finite state (i.e., regular), an OT grammar can only count constraint violations up to a certain point. In other words, for a given n, a finite state OT grammar can correctly select a candidate with n violations over $n + 1$, but it cannot distinguish a candidate with $n + 1$ violations from one with $n + 2$ violations. The

other reason is that optimization is global in the sense that it works to select candidates with fewer constraint violations regardless of where the violations occur within candidates.

It may be possible to identify a constraint set which has the effect of removing the non-regular maps from the factorial typology. Such a solution misses the point in two ways. First, this simple example demonstrates that optimization itself makes non-regularity possible. The particular mixture of constraints would be blunting the power of the core component of the theory. Second, such a constraint set misses the important generalizaton that the maps belong to a particular subregular class.

To put the point more strongly: there seems to be a conspiracy among phonological processes with local triggers that they all have the property of being an ISL or OSL function, but OT has no clear way to account for this. In fact, the above cited work shows that classic OT cannot even account for the weaker fact that phonological maps can be modeled with regular relations.

A reviewer challenges this critique of OT, based on that 1) in OT there are no individual processes, and 2) unlike regular relations, the ISL functions are not known to be closed under composition, so that a grammar comprised of individual ISL processes may result in a total map that is not ISL, in which case it is subject to the same limitation we claim for OT. We address these critiques in turn.

First, as noted in §2, a complete OT grammar includes a set of 'core rankings' like *CAD $\gg$ Faith(A$\mapsto$B) (Baković, 2013), each of which can be thought of as generating an individual process. The question thus remains: why are those maps that involve local triggers restricted to being ISL? Our proposal is that local phonological maps are Input Strictly Local because that is how they are learned.

Second, we are not claiming that phonological grammars are the composition of ISL functions. It is not the case that a given ISL function can only represent a single

process (see §5.3). Our claim is that a map which was traditionally viewed as the composition of multiple rules, each with a bounded structural description applying simultaneously, is a single ISL map. It is true that more work is needed to understand how a set of ISL (and OSL) functions interact when combined into a single grammar (be it by composition or some other operation such as parallel intersection or priority union, etc.). It is also true we want to better understand how these classes interact with the additional class(es) needed for unbounded patterns. We are eagerly pursuing answers to these questions, but that does not change the fact that our findings with respect to local maps are hard to understand in the context of OT.

8 Conclusion

This paper has shown that phonological UR-SR maps involving local triggers (whether expressed in SPE or OT) belong to a more restricted computational complexity class than has been previously established. Specifically, we defined a properly subregular class of string-to-string functions called Input Strictly Local functions and argued that they provide the right computational notion of phonological locality. We also briefly introduced OSL functions, which can model iterative spreading processes with one-sided contexts, or more generally processes where the context of the rule is met in its output. Both ISL and OSL exemplify a short-term memory model, which can only use information in this 'memory buffer' to determine what to output. In this way, the well-recognized idea that locality is relevant to phonology was made precise under a computational analysis, leading to a better understanding of what constitutes a humanly possible phonological map.

These findings also have significant (and related) implications for learning. It was explained how the ISL class is learnable by the provably correct and efficient learning

algorithm ISLFLA. Faced with linguistic input, the predetermined biases of this learner cause it to focus on certain information in the input and to ignore other information. If humans are biased to generalize on the basis of Input Strict Locality, then this learning bias helps to explain *why* local phonological maps are in fact ISL functions.

References

Albright, Adam. 2009. Feature-based generalisation as a source of gradient acceptability. *Phonology* 26:9–41.

Alexander, William D. 1920. *A Short Synopsis of the Most Essential Points in Hawaiian Grammar*. Honolulu, Thomas G. Thrum.

Anderson, Stephen R. 1974. *The Organization of Phonology*. New York: Academic Press.

Angluin, Dana. 1982. Inference of reversible languages. *Journal for the Association of Computing Machinery* 29:741–765.

Ao, Benjamin. 1991. Kikongo nasal harmony and context-sensitive underspecification. *Linguistic Inquiry* 22:193–96.

Baković, Eric. 2007. A revised typology of opaque generalisations. *Phonology* 24:217–259.

Baković, Eric. 2013. *Blocking and Complementarity in Phonological Theory*. Bristol, CT: Equinox.

Beesley, Kenneth R., and Lauri Karttunen. 2003. *Finite State Morphology*. Center for the Study of Language and Information.

Bennett, William. 2013. Dissimilation, consonant harmony, and surface correspondence. Doctoral dissertation, Rutgers.

Berwick, Robert. 1982. Locality principles and the acquisition of syntactic knowledge. Doctoral dissertation, MIT.

Berwick, Robert. 1985. *The Acquisition of Syntactic Knowledge*. Cambridge, MA: MIT Press.

Berwick, Robert C., Kazuo Okanoya, Gabriel J.L. Beckers, and Johan J. Bolhuis. 2011. Songs to syntax: the linguistics of birdsong. *Trends in Cognitive Sciences* 15:113 – 121.

Blevins, Juliette, and Andrew Garrett. 1998. The origins of consonant-vowel metathesis. *Language* 74:508–556.

Blevins, Juliette, and Andrew Garrett. 2004. The evolution of metathesis. In *Phonetically Based Phonology*, edited by B. Hayes, R. Kirchner, and D. Steriade, 117–156. Cambridge, Cambridge University Press.

Buckley, Eugene. 2011. Metathesis. In *The Blackwell Companion to Phonology Volume 3*, edited by Marc van Oostendorp, Colin J. Ewen, Elizabeth Hume, and Keren Rice. Wiley-Blackwell.

Castellanos, Antonio, Enrique Vidal, Miguel A. Varó, and José Oncina. 1998. Language understanding and subsequential transducer learning. *Computer Speech and Language* 12:193–228.

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

Chandlee, Jane, Angeliki Athanasopoulou, and Jeffrey Heinz. 2012. Evidence for classifying metathesis patterns as subsequential. In *WCCFL 29: Proceedings of the 29th West Coast Conference on Formal Linguistics*, edited by Jaehoon Choi, E. Alan Hogue, Jeffrey Punske, Deniz Tat, Jessamyn Schertz, and Alex Trueman, 303–309. Somerville, MA: Cascadilla.

Chandlee, Jane, Rémi Eyraud, and Jeffrey Heinz. 2015a. Output strictly local functions. In *Proceedings of the 14th Meeting on the Mathematics of Language (MOL 2015)*.

Chandlee, Jane, and Jeffrey Heinz. 2012. Bounded copying is subsequential: implications for metathesis and reduplication. In *Proceedings of SIGMORPHON 12*.

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

Chandlee, Jane, Adam Jardine, and Jeffrey Heinz. 2015b. Learning repairs for marked structures. In *Proceedings of the 2014 Annual Meeting of Phonology*.

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory 113124* .

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

Churchward, Clerk Maxwell. 1940. *Rotuman grammar and dictionary*. Sydney: Methodist Church of Australasia, Department of Overseas Missions.

Davidson, Joseph Orville, Jr. 1977. A contrastive study of the grammatical structures of Aymara and Cuzco Quechua. Doctoral dissertation, University of California, Berkeley.

Dell, François. 1973. *Les Régles et Les Sons*. Paris: Hermann.

Dell, François. 1980. *Generative Phonology and French Phonology*. Cambridge: Cambridge University Press.

Dell, François. 1985. *Les Régles et Les Sons*. 2 ed. Paris: Hermann.

Dereau, Léon. 1955. *Cours de Kikongo*. Namur: A. Wesmael-Charlier.

Dumenil, Annie. 1987. A rule-account of metathesis in Gascon. *Lingvisticae Investigationes* XI:81–113.

Ernestus, Mirjam, and R. Harald Baayen. 2003. Predicting the unpredictable: Interpreting neutralized segments in Dutch. *Language* 79:5–38.

Frank, Robert, and Giorgio Satta. 1998. Optimality Theory and the generative complexity of constraint violability. *Computational Linguistics* 24:307–315.

Gainor, Brian, Regine Lai, and Jeffrey Heinz. 2012. Computational characterizations of vowel harmony patterns and pathologies. In *WCCFL 29: Proceedings of the 29th West Coast Conference on Formal Linguistics*, edited by Jaehoon Choi, E. Alan Hogue, Jeffrey Punske, Deniz Tat, Jessamyn Schertz, and Alex Trueman, 63–71. Somerville, MA: Cascadilla.

Gerdemann, Dale, and Mans Hulden. 2012. Practical finite state Optimality Theory. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, 10–19. Donostia–San Sebastin: Association for Computational Linguistics.

Gildea, Daniel, and Daniel Jurafsky. 1996. Learning bias and phonological-rule induction. *Computational Linguistics* 22:497–530.

Gold, E. Mark. 1967. Language identification in the limit. *Information and Control* 447–474.

Graf, Thomas. 2010. Comparing incomparable frameworks: A model theoretic approach to phonology. In *University of Pennsylvania Working Papers in Linguistics*, vol. 16(1).

Grammont, Maurice. 1905-1906. La métathèse dans le parler de Bagnéres-de-Luchon. *Mémoires de la Société de Linguistique de Paris* 13:73–90.

Halle, Morris. 1975. Confessio grammatici. *Language* 51:525–535.

Hansson, Gunnar Ólafur. 2001. Theoretical and typological issues in consonant harmony. Doctoral dissertation, University of California, Berkeley.

Hansson, Gunnar Ólafur. 2010. *Consonant Harmony: Long-Distance Interaction in Phonology*. University of California Publications in Linguistics, 145. Berkeley, CA: University of California Press.

Harris, Zelig. 1951. *Methods in Structural Linguistics*. Chicago: University of Chicago Press.

Heinz, Jeffrey. 2007. The inductive learning of phonotactic patterns. Doctoral dissertation, University of California, Los Angeles.

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

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

Heinz, Jeffrey. 2014. Culminativity times harmony equals unbounded stress. In *Word Stress: Theoretical and Typological Issues*, edited by Harry van der Hulst, chap. 8. Cambridge, UK: Cambridge University Press.

Heinz, Jeffrey, Colin de la Higuera, and Menno van Zaanen. 2015. *Grammatical Inference for Computational Linguistics*. Morgan and Claypool.

Heinz, Jeffrey, and Regine Lai. 2013. Vowel harmony and subsequentiality. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, edited by Andras Kornai and Marco Kuhlmann, 52–63.

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*, 58–64. Portland, Oregon, USA: Association for Computational Linguistics.

de la Higuera, Colin. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

Hopcroft, John E., Rajeev Motwani, and Jeffrey D. Ullman. 2000. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley.

Hulden, Mans. 2009a. Finite-state machine construction methods and algorithms for phonology and morphology. Doctoral dissertation, University of Arizona.

Hulden, Mans. 2009b. Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, 29–32. Association for Computational Linguistics.

Hyman, Larry. 1975. *Phonology: Theory and Analysis*. Holt, Rinehart and Winston.

Jardine, Adam. to appear. Computationally, tone is different. *Phonology* .

Jardine, Adam, Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. 2014. Very efficient learning of structured classes of subsequential functions from positive data. In *Proceedings of the Twelfth International Conference on Grammatical Inference (ICGI 2014)*, edited by Alexander Clark, Makoto Kanazawa, and Ryo Yoshinaka, vol. 34, 94–108. JMLR: Workshop and Conference Proceedings.

Jensen, John T. 1974. A constraint on variables in phonology. *Language* 50:675–686.

Johnson, C. Douglas. 1972. *Formal Aspects of Phonological Description*. The Hague: Mouton.

Joshi, Aravind K. 1985. Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions? In *Natural Language Parsing*, edited by David Dowty, Lauri Karttunen, and Arnold Zwicky, 206–250. Cambridge University Press.

Kaplan, Ronald M., and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 371–387.

Karttunen, Lauri. 1998. The proper treatment of Optimality Theory in computational phonology. *Finite-state Methods in Natural Language Processing* .

Kenstowicz, Michael, and Charles Kisseberth. 1977. *Topics in Phonological Theory*. New York: Academic Press.

Koskenniemi, Kimmo. 1983. *Two-Level Morphology: A general computational model for word-form recognition and production*. University of Helsinki, Department of General Linguistics.

Legendre, Géraldine, Yoshiro Miyata, and Paul Smolensky. 1990. Harmonic grammar: A formal multi level connectionist theory of linguistic well formedness: Theoretical foundations. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, 388–395. Cambridge, MA.

Lipski, John M. 1992. Metathesis as template-matching: A case study from Spanish. *Folia Linguistica Historica* XI:89–104.

Lundskaer-Nielsen, Tom, and Philip Holmes. 2011. *Danish: An Essential Grammar*. Routledge Essential Grammars. Routledge.

Luo, Huan. 2013. Long-distance consonant harmony and subsequentiality. Unpublished manuscript, University of Delaware.

McCarthy, John J. 2000. Harmonic serialism and parallelism. In *NELS 30: Proceedings of the 30th Annual Meeting of the North East Linguistic Society*, edited by Masako Hirotani, Andries Coetzee, Nancy Hall, and Jiyung Kim, 501–524. GLSA, University of Massachusetts Amherst.

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

Meinof, Carl. 1932. *Introduction to the phonology of the Bantu languages*. Berlin: Dietrich Reimer/Ernst Vohsen. Trans. by N. J. van Warmelo.

Mielke, Jeff. 2008. *The Emergence of Distinctive Features*. Oxford: Oxford University Press.

Mielke, Jeff, and Elizabeth Hume. 2001. Consequences of word recognition for metathesis. In *Surface Syllable Structure and Segment Sequencing*, edited by Elizabeth Hume, Norval Smith, and Jeroen van de Weijer, 135–158. Leiden: HIL.

Miller, George Armitage. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review* 63:81–97.

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

Nevins, Andrew. 2010. *Locality in Vowel Harmony*. MIT Press.

Noske, Roland. 1993. *A Theory of Syllabification and Segmental Alternation*. Niemeyer, Tübingen.

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

Oncina, Jose, Pedro García, and Enrique Vidal. 1993. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 448–457.

Oncina, José, and Miguel A. Varò. 1996. Using domain information during the learning of a subsequential transducer. *Lecture Notes in Computer Science - Lecture Notes in Artificial Intelligence* 313–325.

Onn, Farid M. 1980. *Aspects of Malay Phonology and Morphology: A Generative Approach*. Kuala Lumpur: Universiti Kebangsaan Malaysia.

Pater, Joe. 2004. Austronesian nasal substitution and other NC effects. In *Optimality Theory in Phonology: A Reader*, edited by John McCarthy, 271–289. Oxford and Malden, MA: Blackwell.

Pater, Joe. 2012. Serial Harmonic Grammar and Berber syllabification. In *Prosody Matters: Essays in Honor of Elisabeth O. Selkirk*, edited by Toni Borowsky, Shigeto Kawahara, Takahito Shinya, and Mariko Sugahara, 43–72. London, Equinox Press.

Payne, Amanda. 2013. Restricting phonology: Dissimilation as a subsequential process. *NELS 44* .

Prince, Alan, and Paul Smolensky. 2004. *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell.

Riggle, Jason. 2004. Generation, recognition, and learning in finite state Optimality Theory. Doctoral dissertation, UCLA.

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

Roche, Emmanuel, and Yves Schabes. 1997. *Finite-State Language Processing*. Cambridge, MA and London, English: Bradford, MIT Press.

Rogers, James, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. 2013. Cognitive and sub-regular complexity. In *Formal Grammar, Lecture Notes in Computer Science*, edited by Glyn Morrill and Mark-Jan Nederhof, vol. 8036, 90–108. Springer.

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

Rohlfs, Gerhard. 1950. *Historische grammatik der unteritalienischen Gräzität*. München: Verlag der Bayerischen Akademie der Wissenchaften.

Rose, Sharon, and Rachel Walker. 2004. A typology of consonant agreement as correspondence. *Language* 80:475–531.

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

Schieber, Stuart M. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy* 333–343.

Suzuki, Keiichiro. 1998. A typological investigation of dissimilation. Doctoral dissertation, University of Arizona.

Tesar, Bruce. 2008. Output-driven maps. ROA-956.

Tesar, Bruce. 2012. Learning phonological grammars for output-driven maps. In *NELS 39: Proceedings of the 39th Annual Meeting of the North Eastern Linguistic Society*, edited by Suzi Lima, Kevin Mullin, and Brian Smith, 785–798. University of Massachusetts Amherst: GLSA.

Tesar, Bruce. 2014. *Output-Driven Phonology: Theory and Learning*. Cambridge, Cambridge University Press.

Trubetzkoy, Nikolai S. 1969. *Principles of phonology*. Berkeley & Los Angeles: University of California Press. Originally published 1939 as Grundzüge der Phonologie. Göttingen: van der Hoeck & Ruprecht.

Ultan, Russell. 1978. A typological view of metathesis. In *Universals of human language, Vol. 2, Phonology*, edited by Joseph H. Greenberg, 367–402. Stanford, CA, Stanford University Press.

Vennemann, gen. Nierfeld Theo. 1988. *Preference Laws for Syllable Structure and the Explanation of Sound Change*. Berlin, Mouton de Gruyter.

Walker, Rachel. 2011. *Vowel Patterns in Language*. Cambridge: Cambridge University Press.

Warner, Natasha, Allard Jongman, Anne Cutler, and Doris Mücke. 2001. The phonological status of Dutch epenthetic schwa. *Phonology* 18:387–420.

Webb, Nancy. 1965. Phonology and noun morphology of the Kindibu dialect of Kikongo. Master's thesis, UCLA.

Wilson, Colin. 2006. Learning phonology with substantive bias: An experimental and computational study of velar palatalization. *Cognitive Science* 30:945–982.

Jane Chandlee
Department of Computer Science
Haverford College
370 Lancaster Avenue
Haverford, PA 19041
jchandlee@haverford.edu

Jeffrey Heinz
Department of Linguistics and Cognitive Science
University of Delaware
125 E. Main Street
Newark, DE 19716
heinz@udel.edu

Appendix: Constructing an ISL FST for a phonological rule

This appendix explains constructively how any SPE-style rewrite rule with a finite structural description describes an ISL function. One way to think about this construction is that by explicitly linking the syntactic form of the rule to an infinitely-sized extension (set of UR-SR pairs), it provides a semantic interpretation of the rewrite rule. Other interpretations of SPE-stye rules were given by Johnson (1972) and Kaplan and Kay (1994). It remains an objective of future work to determine whether these interpretations coincide for the class of rules considered here.

It will be useful to introduce some notation for strings. If p and s are strings then $ps \cdot s^{-1} = p$. We will also need to know the degree to which the end of one string may overlap with the beginning of another string. An ordered pair of strings (u, v) overlap with s if there exists $s, u', v' \in \Sigma^*$ such that $u = u's$ and $v = sv'$. Note that every pair of strings always overlap with the empty string. We will make use of the *longest* overlapping string of a pair (u, v), denoted $\texttt{los}(u, v)$.

Given a rule of the form in (30),

(30) $A \rightarrow B \ / \ C\_\_D$

we interpret A, B, C, and D as follows. Recall that Σ is the alphabet of symbols that input strings can include; we more precisely call this the input alphabet. We use Γ for the output alphabet, or the alphabet of symbols the output can include, which we assume to include Σ.

A is the set of possible targets and B is the set of possible structural changes, but these are in a particular correspondence. For example, in final devoicing such pairs would only include (b, p), (d, t), (g, k), etc., since it is not the case that a given target in A can be replaced with *any* member of B. We formalize this as follows. Let $A \rightarrow B$ be a subset of $\Sigma^+ \times \Gamma^*$.[22] We define A as the set $\{a \mid (a, b) \in \mathrm{A} \rightarrow \mathrm{B}\}$ and B as the set $\{b \mid (a, b) \in \mathrm{A} \rightarrow \mathrm{B}\}$. We require for each $a \in A$ that if $(a, b), (a, b') \in \mathrm{A} \rightarrow \mathrm{B}$ then $b = b'$ (so $A \rightarrow B$ is functional). We also require, following Johnson (1972) and Kaplan and Kay (1994), that for all $a, a' \in A$, $\texttt{los}(a, a') = \lambda$ (so no two elements of A overlap with a non-empty string).

C represents the set of possible left contexts of the rule, with the option of the first (and only the first) symbol of a string in C being the word-initial boundary, #. So $C \subseteq \{\#, \lambda\} \cdot \Sigma^*$. Likewise, D is the set of possible right contexts, with the option of the last (and only the last) symbol being the word-final boundary, #. So $D \subseteq \Sigma^* \cdot \{\#, \lambda\}$.

Subsequential FSTs include a set of states, Q, an initial state, a set of transitions, δ, and the output function σ. We consider transitions to be elements of $Q \times (\{\#\} \cup \Sigma) \times \Gamma^* \times Q$ so the transition (q, a, u, q') indicates a transition from q to q' reading a and writing u. The output function maps states to output strings (so

$\sigma : Q \to \Gamma^*$).

Provided C, A, and D are finite sets, we can interpret the rule as an ISL function f by constructing an ISL FST as follows. We represent with CAD the set of all strings consisting of a string from C followed by a string from A followed by a string from D. Set the value of k to be the length of the longest string in CAD.

We will now define the construction of a k-ISL FST for a given rule. The set of states is a stringset which includes the empty string, all possible strings of length $k-1$ and all strings beginning with # that are of length less than $k-1$. So $Q = \{\#\}\Sigma^{\leq k-2} \cup \Sigma^{k-1} \cup \{\lambda\}$. The single start state, q_0, is the empty string λ.

To facilitate the definition of the transitions, we first define two helpful auxiliary concepts. First, let H be the set of all possible strings formed by concatenating a string from A (a target) with a string from D (a right context). So $H = \{ad \mid a \in A \text{ and } d \in D\}$. We will actually be interested in the proper prefixes of this set, denoted $PP(H)$. (Formally $PP(H) = \{w \mid \texttt{Pref}(H) - H\}$, where $\texttt{Pref}(H)$ is the set of all prefixes of all strings in H.) Note that since distinct targets do not overlap with a non-empty string, $PP(H) \cap AD = \emptyset$.

If the FST reads in a left context string (a string c from C) and then begins to read a string h from $PP(H)$, it will 'hold' h, meaning it will not produce any output until it has verified whether or not the entire context for the structural change has been met (i.e., it reads an entire *cad* string). Until it verifies this, it doesn't know whether to output *ad* unchanged (if the context turns out to not be met) or *bd* (if the context is met). In either case it can go ahead and output the c portion because this portion is always unchanged.

Second, the following auxiliary function $\pi : Q \to \Gamma^*$ determines, for any state q, the (possibly empty) string being 'held' in q. Let $\pi(q)$ be defined as follows. If there exists $s \in \Sigma^*, c \in C$ and $z \in PP(H)$ such that $q = scz$ then $\pi(q) = z$; otherwise,

$\pi(q) = \lambda$.

Next we define the transition function. There are five cases that determine the transition from every state $q \in Q$ upon reading every input symbol $i \in \Sigma$ as follows.

1. (Start state) If $q = q_0$, then $(q, \#, \lambda, \#) \in \delta$.

2. (Holding) If there exists $s \in \Sigma^*$, $c \in C$, and $h \in PP(H)$ such that $q = sch$ and $hi \in PP(H)$, then letting $q' = \texttt{Suff}^{k-1}(qi)$, we have $(q, i, \lambda, q') \in \delta$.

3. (Releasing, match) If there exists $s \in \Sigma^*$, $c \in C$, $d \in D$, $(a, b) \in$ A$\rightarrow$ B, and $h \in PP(H)$ such that $q = sch$, and $hi = ad$, then letting $o = bd$, $q' = \texttt{Suff}^{k-1}(qi)$, $t = \pi(q')$, and $r = \texttt{los}(o, t)$, we have $(q, i, o \cdot r^{-1}, q') \in \delta$.

4. (Releasing, no match) If there exists $s \in \Sigma^*$, $c \in C$, and $h \in PP(H)$ such that $q = sch$ but neither $hi \in PP(H)$ nor $hi \in AD$, then letting $o = \pi(q)i$, $q' = \texttt{Suff}^{k-1}(qi)$, $t = \pi(q')$, and $r = \texttt{los}(o, t)$, we have $(q, i, o \cdot r^{-1}, q') \in \delta$.

5. (Elsewhere) If none of the above conditions hold then $(q, i, i, \texttt{Suff}^{k-1}(qi)) \in \delta$.

Next, we define the output function σ for every $q \in Q$. If there exists $c \in C$, $(a, b) \in A \rightarrow B$, $d \in D$, such that $q\# = cad$ then $\sigma(q) = bd$; otherwise let $\sigma(q) = \pi(q)$.

Finally, we define the map that the ISL FST generates as follows. Let δ^* be the extended transition function so that δ^* is a function which maps elements of $Q \times \{\#, \lambda\} \cdot \Sigma^* \mapsto Q$ (this is possible since δ is deterministic). Then the map f that the ISL FST T generates, denoted f_T, is defined to be

$$f_T = \{(x, y) \mid (\exists q \in Q, u \in \Gamma^*)[(q_0, \#x, u, q) \in \delta^* \wedge y = u \cdot \sigma(q)]\} \ . \tag{1}$$

That f_T is an ISL function follows from Theorem 3 in Chandlee *et al.* (2014) and the above construction.

Note that while the construction presented here yields SFSTs with many states, since these SFSTs are deterministic efficient algorithms exist for minimizing them (Mohri, 1997).

We now present an example of this construction using the rule of intervocalic voicing shown in (31):

(31) T → D / V __V

We assume $\Sigma = \Gamma =$ {T, D, V, N}. The sets C, D, and $A \to B$ are as follows: $C =$ {V}, $D =$ {V}, $A =$ {T}, $B =$ {D}, $A \to B =$ {(T, D)}. The longest string in CAD is VTV, so $k = 3$. We define the ISL FST $T = \{Q, q_0, \delta, \sigma\}$ as follows.

The state set Q includes {λ, #, #T, #D, #V, #N, TT, TD, TV, TN, DT, DD, DV, DN, VT, VD, VV, VN, NT, ND, NV, NN}. The start state $q_0 = \lambda$.

The set $H =$ {TV} and so $PP(H) =$ {T}. The four conditions on δ add transitions as follows:

Condition 1 (start state) adds (λ, #, λ #) to δ.

Condition 2 (holding) adds the following transitions to δ:

(#V, T, λ, VT) (VV, T, λ, VT) (NV, T, λ, VT) (DV, T, λ, VT)

(TV, T, λ, VT)

Condition 3 (releasing match) adds the following transitions to δ:

(VT, V, DV, TV)

Condition 4 (releasing, no match) adds the following transitions to δ:

(VT, T, TT, TT) (VT, D, TD, TD) (VT, N, TN, TN)

Condition 5 (elsewhere) adds the following transitions to δ:

(#, D, D, #D)	(#, N, N, #N)	(#, T, T, #T)	(#, V, V, #V)
(#D, D, D, DD)	(#D, N, N, DN)	(#D, T, T, DT)	(#D, V, V, DV)
(#N, D, D, ND)	(#N, N, N, NN)	(#N, T, T, NT)	(#N, V, V, NV)
(#T, D, D, TD)	(#T, N, N, TN)	(#T, T, T, TT)	(#T, V, V, TV)
(#V, D, D, VD)	(#V, N, N, VN)	(#V, V, V, VV)	(DD, D, D, DD)
(DD, N, N, DN)	(DD, T, T, DT)	(DD, V, V, DV)	(DN, D, D, ND)
(DN, N, N, NN)	(DN, T, T, NT)	(DN, V, V, NV)	(DT, D, D, TD)
(DT, N, N, TN)	(DT, T, T, TT)	(DT, V, V, TV)	(DV, D, D, VD)
(DV, N, N, VN)	(DV, V, V, VV)	(ND, D, D, DD)	(ND, N, N, DN)
(ND, T, T, DT)	(ND, V, V, DV)	(NN, D, D, ND)	(NN, N, N, NN)
(NN, T, T, NT)	(NN, V, V, NV)	(NT, D, D, TD)	(NT, N, N, TN)
(NT, T, T, TT)	(NT, V, V, TV)	(NV, D, D, VD)	(NV, N, N, VN)
(NV, V, V, VV)	(TD, D, D, DD)	(TD, N, N, DN)	(TD, T, T, DT)
(TD, V, V, DV)	(TN, D, D, ND)	(TN, N, N, NN)	(TN, T, T, NT)
(TN, V, V, NV)	(TT, D, D, TD)	(TT, N, N, TN)	(TT, T, T, TT)
(TT, V, V, TV)	(TV, D, D, VD)	(TV, N, N, VN)	(TV, V, V, VV)
(VD, D, D, DD)	(VD, N, N, DN)	(VD, T, T, DT)	(VD, V, V, DV)
(VN, D, D, ND)	(VN, N, N, NN)	(VN, T, T, NT)	(VN, V, V, NV)
(VV, D, D, VD)	(VV, N, N, VN)	(VV, V, V, VV)	

Finally, the final output function σ (which in this example is identical to the auxiliary function π) is defined as follows: for $q = \text{VT}$, $\sigma(q) = \text{T}$, and for all other q, $\sigma(q) = \lambda$.

Notes

[0]Thanks to James Rogers and Rémi Eyraud for valuable feedback and collaboration throughout the development of this research. We also thank our two anonymous reviewers for questions and suggestions that greatly improved the paper.

[1]We use the word 'function' instead of 'relation' here because these maps by definition map each input to a single output. Thus we make the simplifying assumption that processes do not apply optionally. However, the results presented here can be extended to accommodate phonological variation and optionality, for example by using p-subsequential functions (Mohri, 1997), which map each input to at most p outputs.

[2]There is evidence that this process is not necessarily neutralizing in every language (Ernestus and Baayen, 2003). Whether the rule partially or fully devoices the word-final obstruent is irrelevant to the point under discussion that there is a map, or transformation, from underlying to surface forms.

[3]Savitch (1993) provides another argument for analyzing linguistic generalizations as infinite objects based on the fact that the descriptions thereof can be much smaller than descriptions of finite objects.

[4]See also Baković (2013, chapter 4), who shows how to translate any rule of the form A→B / C__D into a core ranking where a markedness constraint like *CAD outranks those faithfulness constraints violated by A↦B. As he explains, this ranking "is assumed to be embedded within a constraint hierarchy" whose other constraints must also be ranked a certain way.

[5]In other words, the map is a *total* function. A function can also be *partial*, meaning it is not defined for every string in its domain. An example of a partial phonological map would be final devoicing that is only defined for words with at least two vowels. Whether or not to treat domain restrictions as part of a process or separate from it raises interesting theoretical questions, but for the purposes of this paper we will be treating all phonological maps as total.

[6]Software for implementing rule-based and constraint-based phonological analyses based on these insights includes `xfst` (Beesley and Karttunen, 2003) and the open-source `foma` (Hulden, 2009b; Gerdemann and Hulden, 2012).

[7]Without elaborating, Mohri asserts that "Most phonological and morphological rules correspond to p-subsequential functions" (Mohri, 1997, p. 11).

[8]We are simplifying for clarity of presentation. Hawaiian does not allow codas generally. A constraint like NoCoda is also SL if syllable boundaries are represented in the strings.

[9]Note that the term suffix here has no morphological sense. It refers only to some portion of a string starting from its end. Formally, a string s is a suffix of a string w if there is some (possibly empty) string p for which $w = ps$.

[10]Here again the term *prefix* is not morphological, but refers simply to some portion of a string starting from the beginning (i.e., p is a prefix of w if there is some (possibly empty) string s such that $w = ps$).

[11]Even if all of the strings in a set start with a different symbol, the `lcp` for that set is still defined. Since λ is a prefix of every string, the `lcp` in this case would be λ.

[12]In all FSTs in this paper we use reduced alphabets of either natural class abbreviations, like N for nasal consonants, or segmental alphabets with just a subset of the language's sounds. This is for the sake of making the FST more readable (i.e., smaller); whether or not the process is ISL does not depend on this reduced alphabet. In other words, even if we used the full segment alphabet, the process is still ISL for the same value of k. The FST in that case would just have more states and transitions.

[13]The subsequential functions also have a language-theoretic definition, which is that they are those functions which only distinguish finitely many distinct sets of tails. Comparing this definition to Definition 1 it can be seen how the ISL functions are more restrictive in requiring the strings that share a set of tails to *also* share a suffix

of length $k-1$.

[14]This means there will be states for all possible strings of length 0, 1,...,$k-1$. As the unique string of length 0, λ is always included in the state set of an ISL FST. Furthermore, it is always the FST's initial state, because it corresponds to the state where no input has yet been read. In the body of this article, we abstract away from the initial word boundary symbol, but we treat it in the appendix.

[15]This same map was already described with the non-deterministic FST in Figure 3, but we now show that non-determinism is not necessary for modeling final devoicing, as it is an ISL process.

[16]Mielke and Hume (2001) actually argue that all synchronic metathesis involves adjacent segments, and long distance cases like Cuzco Quechua tend to be non-regular or are otherwise amenable to a non-movement analysis. Regardless of whether such maps are genuine metathesis is tangential to our point that they are ISL.

[17]The term 'displacement' is often used instead of metathesis in cases like this where only one segment moves.

[18]OSL FSTS also treat word boundary symbols differently than ISL FSTs. Interested readers may consult Chandlee *et al.* (2015a) for details.

[19]See Chandlee and Heinz (2012) and Heinz and Lai (2013) for more details on the left and right subsequential classes and their relevance to phonology.

[20]OSTIA in fact only provably learns *total* subsequential functions exactly. A total function is defined for any possible input in its domain, in contrast to a *partial* function which may be undefined for certain inputs. Variations on OSTIA (Oncina and Varò, 1996; Castellanos *et al.*, 1998) have been proposed for learning partial functions, with the help of additional information such as negative data or properties of the domain and/or range.

[21]Variables like C_n^m do place an upper bound on the amount of material, though

Halle (1975) claims 'no particular use has been found for the superscript/subscript notation', aside from the unbounded usage like C_0.

[22]Following Johnson (1972), we observe that rules whose targets are of zero length (epenthesis) can be transformed into rules whose targets are of length one.