Learning Phonological Mappings by Learning Strictly Local Functions*

Jane Chandlee and Adam Jardine
University of Delaware

1 Introduction

The current study identifies *locality* as a near-universal property of phonological input-output mappings that describe processes with local triggers and presents a learning algorithm which uses locality as an inductive principle to generalize such mappings from finite data. Input-output (or UR-SR) mappings like the one in (1) are integral to both the rewrite rules of Sound Pattern of English (Chomsky & Halle, 1968) and constraint-driven grammars like Optimality Theory (Prince & Smolensky, 1993, 2004).

(1) {(dip, dip), (baːd, baːt), (bom, bom), (deːg, deːk), ...}

In the above pairs, all word-final obstruents in the input are voiceless in the output. Phonologists will likely recognize this mapping as 'word-final obstruent devoicing', and within an SPE framework, it might be described with a rule like (2a). In OT, it can be captured in part by the constraint ranking in (2b).

(2) a. $[-\text{son}] \Rightarrow [-\text{voice}] / \_\_ \#$

b. $*[-\text{son}, +\text{voice}]\# \gg \text{IDENT}(\text{voice})$

Implicit in both formalisms is the fact that this mapping is *infinite*: for *any* input to which the rule or constraint applies, its output will have a word-final voiceless obstruent. The question of interest for the learning problem is how do children induce these infinite generalizations given only finite data?

We address this question by showing how the computational properties of the target mappings can aid in learning. It's been known since Johnson (1972), Koskenniemi (1983), and Kaplan & Kay (1994) that phonological rules describe regular relations, but regular relations are not believed to be identifiable in the limit from positive data (Gold, 1967). We instead propose that the phonological learner takes as its hypothesis space the class of Strictly Local functions, a subclass of the subsequential functions. Our interest in subsequential mappings stems from a body of work that has classified many phonological processes as subsequential (Gainor et al., 2012; Chandlee et al., 2012; Chandlee & Heinz, 2012; Heinz & Lai, 2013; Luo, 2013; Payne, 2013; Jardine, 2013). However, though Oncina et al. (1993) prove that subsequential functions are identifiable in the limit from positive data by the algorithm known as OSTIA, Gildea & Jurafsky (1996) found that OSTIA fails to learn phonological rules from natural language corpora. The current study builds on this work by modifying OSTIA so that it targets the Strictly Local subclass of subsequential functions. The resulting algorithm successfully learns a wide range of local phonological processes while also excluding other subsequential but non-phonological mappings.

This paper is structured as follows. Strictly Local functions and the formal languages on which they are based are discussed in §2. The algorithm OSTIA is presented in §3. The Strictly Local Function Learning Algorithm, an augmentation of OSTIA based on the properties of Strictly Local functions, is introduced in §4, along with the theoretical results and several simulations. §5 further discusses the issues that arise when using natural language corpora, and §6 concludes.

* Thanks to Jeffrey Heinz, Rémi Eyraud, James Rogers, and the audience at Phonology 2013 for valuable feedback and guidance.

Proceedings of *Phonology 2013*
Completed February 15, 2014

2 Strictly Local Functions

2.1 *Strictly Local languages* As the well-studied Strictly Local formal languages form the basis for the Strictly Local functions that our learning algorithm targets, we will first introduce the SL language class and discuss how it has been applied to natural language phonotactics.

2.1.1 *Formal Languages* Formal languages can be finite or infinite. An example of an infinite language is given in (3). This language includes all strings of a's such that the length of the string is a multiple of three. Note that the symbol λ represents the empty string, which has length zero.

(3) {λ, aaa, aaaaaa, aaaaaaaaa, aaaaaaaaaaaa, ...}

One area of research in formal language aims to categorize such languages in terms of their complexity (i.e., what it takes to compute whether or not a string is in the language). The most-recognized complexity categories form the Chomsky Hierarchy (Chomsky, 1956; Partee et al., 1993), shown in Figure 1.

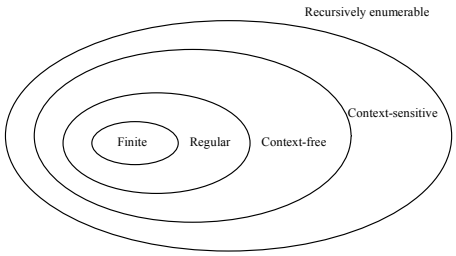

Figure 1: The Chomsky Hierarchy

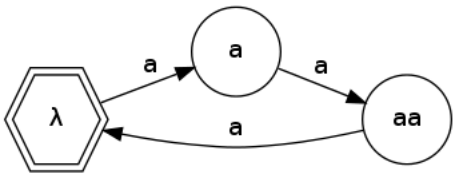

Figure 2: FSA for (3)

The language in (3) is *regular*. The regular languages are those which are describable by finite-state acceptors (FSAs). Languages belonging to regions beyond the regular region, such as the context-free languages, cannot be described by a FSA. An example FSA for the language in (3) is shown on the right in Figure 1. FSAs are made up of states, marked by shapes, and transitions, marked by arrows. In all FSAs in this paper, hexagons are 'start' states, and 'final' states are marked with double lines. The FSA reads a string symbol-by-symbol and transitions between states based on each symbol. For a string to be 'accepted' (i.e., in the language), the FSA must be in a final state when it reaches the end of the string.

For example, given the input aaa, the FSA in Figure 2 will follow the path represented in (4).

(4)

Input:		a		a		a	
State:	λ	$\Rightarrow$	a	$\Rightarrow$	aa	$\Rightarrow$	λ

The FSA starts in the start state λ, reads the first 'a' and takes the transition labelled 'a' to state a. It then reads the second 'a' of the string, moving to state aa, and then moves back to the λ state on the third 'a'. There are no more symbols left to read, but the machine is in a final state, so it accepts the string. Note that if the FSA was presented with the string aa, it would end in state aa, which is not a final state. Thus, the FSA would correctly reject aa, which does not belong to the language in (3).

2.1.2 *Phonology and sub-regular languages* Most phonologists would agree that the pattern represented by the language in (3) would be a bizarre phonotactic pattern, and indeed, to our knowledge, no such pattern exists in natural language phonotactics. However, there is a well-studied hierarchy of *sub-regular* languages (McNaughton & Papert, 1971; Rogers & Pullum, 2011; Rogers et al., 2013), shown in Figure 3, that better capture the kinds of phonotactic patterns we observe in natural language (Heinz, 2009, 2010).

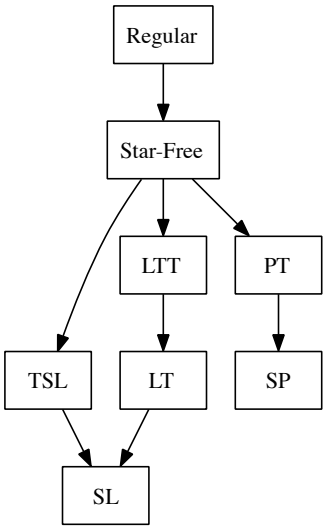

Figure 3: The sub-regular hierarchy

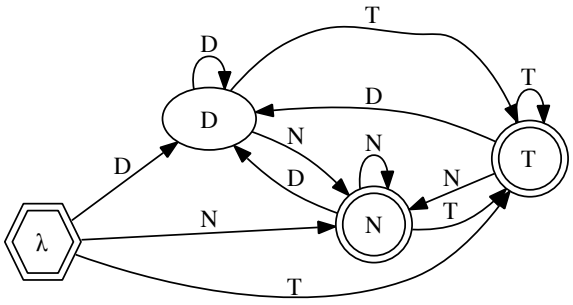

Figure 4: Strictly Local FSA for a language without D#

At the very bottom of the sub-regular hierarchy lies the *Strictly Local* (SL) class of languages, which can be defined with grammars of permissible substrings of a certain length k. Consider the language in (5), which represents the allowable surface forms in a language with word-final obstruent devoicing.

(5) {dip, ju:p, ba:t, bit, bom, bor, de:k, ...}

This language crucially does not allow the substrings $b\#$, $d\#$, $g\#$, $v\#$, $z\#$, $\text{ʒ}\#$, or $d\text{ʒ}\#$. Since these substrings are of length 2, the k-value of this language is 2 (i.e., it is a SL-2 language). Strictly Local languages are recognized by FSAs with a couple of special properties. For example, the 'no final voiced obstruents' language is recognized by the FSA shown in Figure 4. For readability, we use an abbreviated alphabet of D (for voiced obstruents), T (for voiceless obstruents), and N (for sonorants). What makes this FSA a Strictly Local one is the state set, which corresponds to all possible substrings of length $k-1$ (thus in this example, the states are of length 1), and the transition function, which guarantees that being in a state is only possible if that particular substring is the most recent input. In Figure 4, the restriction against word-final D is enforced since D is not a final state. Strings that end in D will thus be rejected.

All phonotactic generalizations that can be defined with a contiguous substring bounded by some k share this property of strict locality (Heinz, 2007, 2009). On the other hand, the 'multiples of three a's' pattern in 3 is provably not Strictly Local for *any* value of k (see Rogers & Pullum (2011) and Rogers et al. (2013) for details). In the next section, we extend this conception of locality to mappings.

2.2 *Strictly Local functions* As stated at the outset, our aim is to model local phonological processes with a restrictive class of functions, called Strictly Local functions. It was shown in the previous subsection that as a set of strings, a language can be represented with a finite state acceptor that either accepts or rejects a given input string. The SL functions, on the other hand, are string-to-string mappings (i.e., input $\mapsto$ output), so to characterize them we use finite state *transducers*. These are finite state machines that read in an input string (just like acceptors) and produce a corresponding output string (unlike acceptors).

Consider the phonological rule in (6), which describes the process of final devoicing. Here we again use the abbreviated alphabet {D, T, N}, with the understanding that a given voiced obstruent will only be mapped to its voiceless counterpart (e.g., d $\mapsto$ t, b $\mapsto$ p, etc.).

(6) D $\Rightarrow$ T / __#

A finite state transducer that represents the mapping of this rule is shown in Figure 5.

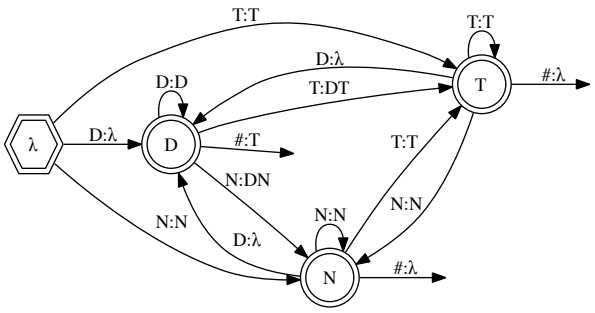

Figure 5: Finite state transducer for the devoicing mapping

Consider the input string TND. The FST reads this string one symbol at a time and produces the output as indicated on the transition labels. The path for this input is shown in (7).

	Input:		T		N		D		#
(7)	State:	λ	⇒	T	⇒	N	⇒	D	
	Output:		T		N		λ		T

Notice that when the FST reads in a D it initially produces no output (i.e., it produces λ). This is because it does not have enough information to determine whether that D should be mapped to T or D. If this D turns out to be the last symbol in the input string, as in this case, then the transition on the end-of-word marker # produces the output T. Note this transition does not lead to another state, indicating that it cannot be taken unless the end of the input has been reached. If any additional input follows, the D is outputted unchanged.

The FST in Figure 5 is a *subsequential* FST, because 1) it is deterministic on the input, 2) all states are final, and 3) it assigns to each state a final output string that is appended to the current output when the end of the input is reached in that state (this is the output on #) (see Mohri (1997) for more on subsequential transducers). It is also a Strictly Local FST, because (like the Strictly Local FSAs described in §2.1.2) 1) the state set corresponds to all possible substrings of length $k-1$, and 2) the transitions are defined such that the FST can only be in a state if those are the most recent $k-1$ symbols of the input. Our claim is that a mapping that can be described with such a FST is a SL function. Formally, we define SL functions as follows.

Definition 1. *A function f is Strictly Local iff there is some k such that f can be described with a subsequential FST for which $Q = \Sigma^{\leq k-1}$ and $\forall q \in Q, a \in \Sigma, (q, a, o, \textit{Suff}^{k-1}(qa)) \in \delta$.*[1]

Recall from the previous section that the SL functions are a subclass of the subsequential functions, which is why by definition all SL functions are also subsequential. It is the restrictions on the state set Q and the transitions δ that make a subsequential function a SL one. What is the value of k? Given a rule like (6), the length of the structural description (in this case, D#) is the k-value of the SL function that models the process. So in this example, again $k = 2$.

2.3 *Learning SL* Now that we have a definition for the class of functions we are interested in learning, we turn to the learning approach. The key idea is that the learner specifically targets the class of SL functions and therefore does not consider any hypothesis function that happens to be consistent with the data it has seen so far but is not a SL function. In this way, the property of strict locality delimits the hypothesis space, as depicted in Figure 6.[2]

[1] By $\text{Suff}^{k-1}(qa)$ we mean the suffix of length $k-1$ of the substring q plus the current input symbol a. Here the term suffix has no morphological meaning, but is simply a portion of the string starting from the end.

[2] The subsequential functions can be divided into two overlapping subclasses, known as left subsequential and right subsequential, with the SL functions cross-cutting these two regions. For reasons of space we abstract away from this detail, but see Heinz & Lai (2013) and Chandlee (2014).

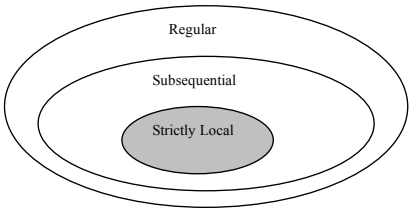

Figure 6: Hypothesis space of the SL learner

As discussed in §1, neither the regular relations nor the subsequential functions appear to be a feasible hypothesis space for a phonological learner. The SL functions, as a proper subset of subsequential (Chandlee, 2014), provide a useful restriction and a defining property that we will show can be used to learn local phonological processes. In terms of empirical coverage, a survey of the approximately 5500 phonological patterns in the P-Base database (v1.95, Mielke (2008)) revealed that at least 96% can be modeled with Strictly Local functions. This includes local substitution, deletion, and epenthesis, as well as bounded metathesis, local partial reduplication, and general affixation (see Chandlee et al. (2012); Chandlee & Heinz (2012); Chandlee (2014)). Excluded from the list of processes that are SL-describable are long-distance processes like vowel harmony with transparent vowels (Nevins, 2010; Gainor et al., 2012; Heinz & Lai, 2013), long-distance consonant harmony (Hansson, 2001; Rose & Walker, 2004; Payne, 2013) and dissimilation (Suzuki, 1998; Bennett, 2013; Luo, 2013), certain tonal patterns (Jardine, 2013), unbounded displacement/metathesis (Chandlee et al., 2012; Chandlee & Heinz, 2012), and non-local partial reduplication (Riggle, 2003). However, we believe the approach to learning local processes presented in this paper can be adapted to the learning of these non-local phenomena. More will be said about this potential in §6. In the next section we present the learning algorithm known as OSTIA, on which our learner is based.

3 OSTIA

The class of subsequential functions, of which the SL functions are a subset, are identifiable in the limit from positive data by the Onward Subsequential Transducer Induction Algorithm (OSTIA) (Oncina et al., 1993). As our learning algorithm for SL functions is a modification of OSTIA, we will first explain in some detail how OSTIA works.

The input data to the algorithm is a set of string pairs that reflect the target subsequential function (i.e., $\{(w, w')$ such that $f(w) = w'\}$, where f is the target function) and the output is a subsequential FST that represents the target function. The first step is to build a prefix tree transducer for this data, which is a finite state transducer that represents *only* the string pairs in the set it was built from. In other words, it will produce the correct output for every input string in the data set, but it cannot generalize to produce output for an input string not present in that set. As an example, consider the small data set in (8). A prefix tree transducer for this set is shown in Figure 7.

(8) {(DND, DNT), (DNT, DNT), (DNN, DNN)}

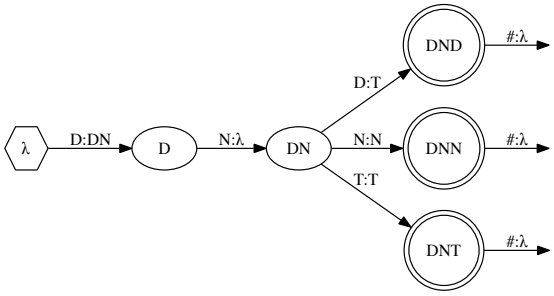

Figure 7: Prefix tree transducer for the data set in (8)

These prefix tree transducers have a particular property known as *onwardness* (Oncina et al., 1993). This property means that output is produced by the transducer as soon as possible (i.e., on a transition as close to the 'root' of the tree as possible).[3] In other words, the transducer produces as much output as possible as soon as it has enough information. Consider the tree in Figure 7. As soon as it reads in a D, it produces the output DN. This reflects the fact that no matter what input follows the D, the output string will begin with DN. While not true generally (i.e., it is not the case that all voiced obstruents are immediately followed by a sonorant), this fact is true based on the small dataset on which the transducer was built.

To generalize from this prefix tree representation to the FST of the target function, OSTIA merges states. State merging is a technique by which two (or more) states are collapsed into a single state that preserves any transitions of the original states.[4] As the states of any FST can be thought of as recording certain information (e.g., state DN in Figure 7 records the fact that the input so far is DN), state merging can be thought of as discarding unnecessary information. What information is either discarded or retained (i.e., what states can be merged) depends on the particular state merging strategy. The state merging strategy of OSTIA is to merge any two states provided that any resulting non-determinism can be removed without altering the transduction the FST represents. As determinism is one of the defining properties of subsequential transducers, this state merging criterion guarantees that the FST the learner outputs will be subsequential.

3.1 *Gildea & Jurafsky (1996)* It was mentioned above that a wide range of phonological processes can be represented with subsquential functions (Gainor et al., 2012; Chandlee et al., 2012; Chandlee & Heinz, 2012; Heinz & Lai, 2013; Luo, 2013; Payne, 2013; Jardine, 2013). It follows that OSTIA should be able to learn phonological mappings. However, OSTIA was put to just such a test by Gildea & Jurafsky (1996), who found that it failed to produce the correct transducer for the English flapping rule, shown in (9).

(9) t ⇒ ɾ / V́ɾ*__V

What went wrong? The input data given to OSTIA was derived from a natural language corpus, which turned out to be insufficient to generalize the target rule. To the degree that we take natural language corpora to simulate the data available to human language learners, this failure of OSTIA is troubling. Gildea & Jurafsky (1996) modified OSTIA with three learning biases: faithfulness (underlying segments are realized similarly on the surface), community (similar segments behave similarly), and context (rules can access contextual variables). This modified OSTIA successfully learned a range of phonological rules, including flapping, and this success was taken as evidence that these biases are useful/necessary for phonological learning.

The learner we present in this paper is also a modification of OSTIA. Specifically, we modify the state merging strategy to enforce not just subsequentiality, but also strict locality (details in the next section). In this way we likewise encode a bias for context, as was done in Gildea & Jurafsky (1996). The advantage of our approach, however, is that our modification does not allow the learner to venture outside of the subsequential class of functions at any point in the algorithm's run. Indeed, it draws a boundary further inside that class, targeting the SL subclass of subsequential. More will be said about the other two learning biases employed by Gildea & Jurafsky (1996), as well as the use of natural language corpora, in §5. In the next section we present our learner, the Strictly Local Function Learning Algorithm (SLFLA), as well as the theoretical results that establish the class of functions it can learn.

4 The SLFLA

The key difference between our learner, the SLFLA, and OSTIA is in the state merging strategy. Like OSTIA, our learner preserves subsequentiality when it merges states. Additionally, it only merges states that meet a certain criterion, one that requires the FST it eventually outputs to be a Strictly Local one. Before we present that state merging criterion, it is necessary to clarify two terms. First, each state in a prefix tree transducer (such as in Figure 7) represents a *prefix* of one of the input strings in the data set the prefix tree was built from (hence the name *prefix* tree). This term has no morphological import; a prefix is simply some portion of a string starting from the beginning.[5] Thus each state in the prefix tree can be thought of as a record

[3] Formally, a subsequential transducer $\tau = \{Q, \Sigma \cup \{\#\}, q_0, \delta\}$ is onward if $\forall q \in Q - \{q_0\}$ the longest common prefix of $\{w|(q, a, w, q') \in \delta\} = \lambda$. See Oncina et al. (1993) for more on onwardness.

[4] For more examples of state merging algorithms see Angluin (1982), Heinz (2009), and de la Higuera (2010).

[5] Formally, a string p is a prefix of a string w if there exists some (possibly empty) string s such that $w = ps$.

of the symbols that were read in up until that state. Second, a *suffix* of a string is some portion of that string starting from the end.[6] Thus a suffix of a certain length n of a state in a prefix tree can be thought of as the n *most recent* symbols that were read in to reach that state.

We can now define the state merging criterion of the SLFLA. The SLFLA only merges states in the prefix tree transducer that have the same suffix of length $k-1$. This means that the only information that is preserved when two states are merged is the most recent $k-1$ symbols by which the states were reached. For example, given the prefix tree in Figure 7 and a k value of 2, the SLFLA would merge states {D, DND} and {DN, DNN}. The result would have a separate state for each possible sequence up to length $k-1$: D, N, T, and λ. This is how the learner converges on a Strictly Local representation of the data; it collapses the prefix tree down to a FST that only maintains a state for each possible sequence of length $k-1$. This means the SLFLA considers fewer merges than OSTIA, since the latter does not have this additional restriction on which states can be merged. In that way the SLFLA can only learn a subset of the functions that OSTIA can learn. As discussed above, this restriction is desirable from a phonological standpoint, because the subset of functions within reach of the SLFLA is sufficient to model local phonological processes.

This method of building a prefix tree and merging states that share a $k-1$-length suffix was previously applied to the learning of SL languages by Heinz (2007) and SL mappings by Chandlee & Koirala (2014). The latter work, however, did not use an onward prefix tree and did not have a principled method for removing non-determinism after state merging. As a result, it is unclear whether that learner *provably* learns the class of SL functions. By incorporating the SL state merging strategy into OSTIA, the current approach is more amenable to a theoretical result of the class of functions the learner can learn.

4.1 *Theoretical result* We can show that the SLFLA will learn any SL function, but its success depends on the input data having a certain property. Recall that OSTIA will reject a merge between two states if the resulting FST has irreparable non-determinism. More specifically, the merge will be rejected if the result contains non-determinism of the form in Figure 8.

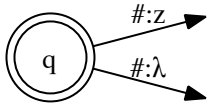

Figure 8: Problematic non-determinism resulting from state merging

The reason this situation must be avoided is that non-determinism is usually removed by OSTIA by first moving any difference between the outputs (in this case z and λ) further down their respective branches and then merging the destination states that can be reached on the same symbol. In Figure 8, however, there are no destination states, so there is nowhere for the output 'residue' to go. Thus whatever state merging led to this situation must be rejected (see Oncina et al. (1993) for more on this method of removing non-determinism).

In the case of the SLFLA, however, it is crucial that the algorithm never encounter this situation (and therefore never have to reject a merge). Since only states with the same $k-1$ suffix are considered for merging, allowing such a merge to be rejected means that two states with the same $k-1$ suffix could remain distinct in the FST the learner outputs, contra to the definition of a Strictly Local transducer. However, it is predictable given a data set whether the learner will encounter the fatal situation, which means we known the learner will succeed provided its input data is closed under a certain property. To understand the nature of this property, consider two states, q_1 and q_2, that need to be merged but differ on the output of #, so q_1 on # has the output z but q_2 has the output λ. Merging these states would lead to the situation in Figure 8. What is needed is another state, q', on the same branch of the prefix tree as q_2, which will be merged prior to the attempted merge of q_1 and q_2. This prior merge will create non-determinism, and the adjustment of the outputs described in the previous paragraph will result in q_2 having the necessary output of z on #. At that point, q_1 and q_2 can be safely merged. Since the presence or absence of states in the prefix tree is entirely dependent on the dataset, we can guarantee the presence of q' by placing a restriction on the data.[7] With this

[6] Formally, a string s is a suffix of a string w if there exists some (possibly empty) string p such that $w = ps$.

[7] Formally, the learner succeeds on data for which the following holds. If the sample contains $(v, y), (v', y')$ such that

understanding of the meaning of closed learning sample, we now present the following theorem and a sketch of its proof.

Theorem 1. *The class of SL functions is learnable from a closed learning sample.*

Proof. Let f be a SL-k function and τ_f its Strictly Local transducer. Given a closed learning sample, the SLFLA will output τ_f. We know this because,

a. **Only** states with the same $k-1$ suffix will be merged. When searching for a state to merge a given state with, the SLFLA only considers states with the same $k-1$ suffix. Additional state merges that are performed to remove non-determinism will also be between states with the same $k-1$ suffix. This is true by construction of the prefix tree. Let q be the origin state of these two states and $x = \text{Suff}^{k-1}(q)$. If the two destination states involved in the non-determinism are reached on some a in the alphabet, then both will have xa as a suffix. Since the length of x is $k-1$, it follows that they have the same $k-1$ suffix.

b. **All** states with the same $k-1$ suffix will be merged, since the closed learning sample guarantees that no merge will be rejected.

c. After all merges are complete, the remaining state set will be $Q = \Sigma^{\leq k-1}$. By construction of the initial prefix tree, $\forall q \in Q, a \in \Sigma, (q, a, o, \text{Suff}^{k-1}(qa)) \in \delta$. Therefore the learner's output is τ_f, by Definition 1.

□

4.2 *Simulations* In this section we present the results of the learner on four test cases based on attested phonological processes.

4.2.1 *Final devoicing* Using the alphabet $\Sigma = \{\text{D, T, N}\}$, where again D is a voiced obstruent, T is a voiceless obstruent, and N is a sonorant, we generated a dataset as follows. All possible input strings up to length 5 were paired with the correct output string according to the devoicing rule, repeated in (10) (for a total of 363 pairs). The output of the learner is shown in Figure 9.

(10) D ⇒ T / __#

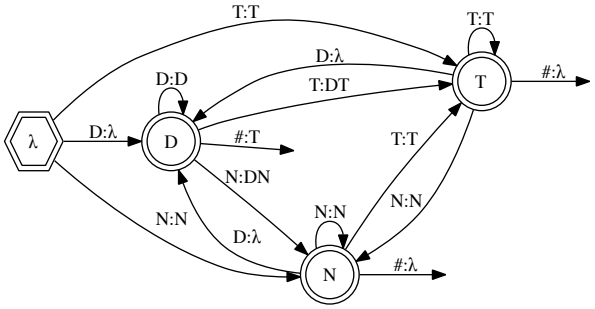

Figure 9: SLFLA output for final devoicing test, k =2

4.2.2 *Fricative deletion* The second test was based on a fricative deletion process found in Greek and represented with the rule in (11) (Joseph & Philippaki-Warburton, 1987).

(11) {θ, ð} ⇒ λ / __ {s, θ}

Suff$^{k-1}(v)$ = Suff$^{k-1}(v')$ but the output from state v on # is z and the output from state v' on # is λ, then the sample must also contain (uv', xz) such that $v' = ua$ for some $a \in \Sigma$.

Using the alphabet $\Sigma = \{\theta, \eth, s, ?\}$, where (following Beesley & Karttunen (2003)) ? stands in for any segment except for the three already listed, we again generated an artificial corpus of input-output strings that reflect the target rule (1364 pairs total). The output of the learner is shown in Figure 10.

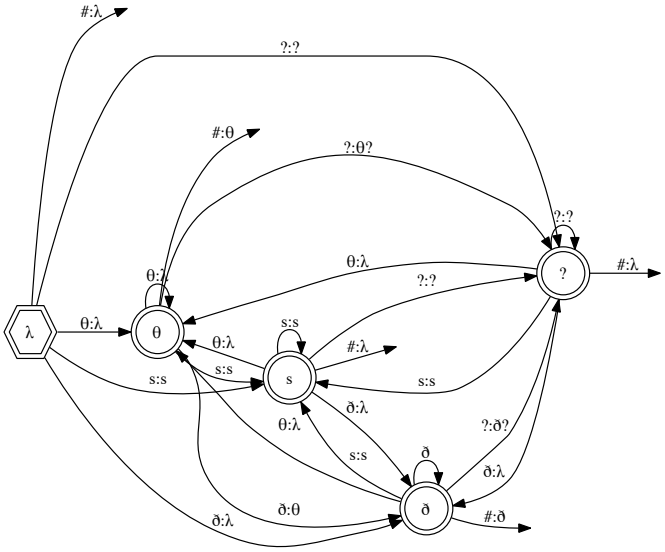

Figure 10: SLFLA output for fricative deletion test, k =2

4.2.3 *Schwa epenthesis* The third test was based on a Dutch schwa-epenthesis process, as expressed with the rule in (12) (Warner et al., 2001).

(12) $\lambda \Rightarrow$ ə / {l, r} __ [-coronal]

Using the alphabet $\Sigma = \{$l, r, k, ?$\}$, where k represents any non-coronal consonant and ? again stands for any segment except the other three, we generated 1364 input-output pairs for this rule. The output of the learner is shown in Figure 11.

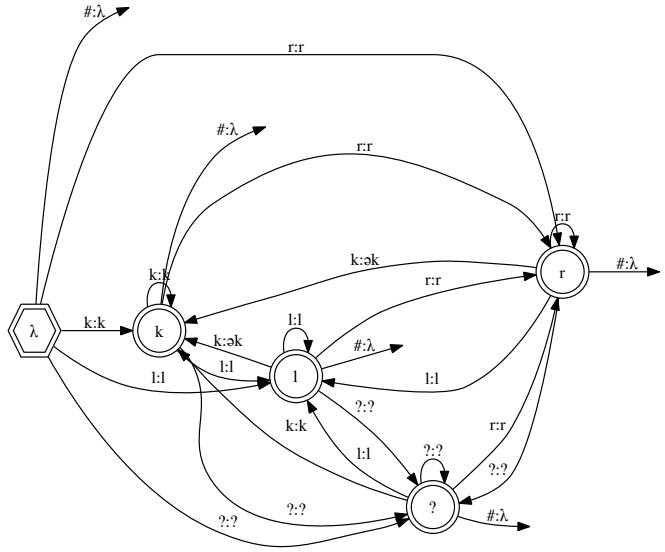

Figure 11: SLFLA output for schwa-epenthesis test, k =2

4.2.4 *English flapping* The fourth and final test was based on the English flapping rule in (13).

(13) $t \Rightarrow ɾ \;/\; \acute{V} \_\_ V$

Using the alphabet $\Sigma = \{V, v, t, ?\}$, where V is a stressed vowel, v is an unstressed vowel, and ? again stands for any segment except the other three, we generated 1364 input-output pairs for this rule. The output of the learner is shown in Figure 12. Note that this FST has been minimized for ease of reading.

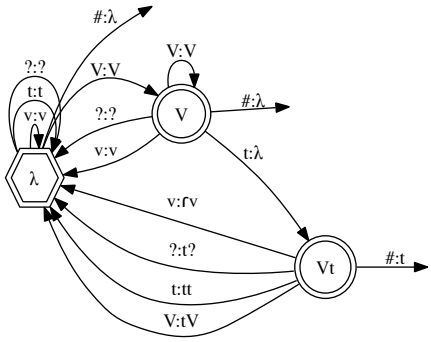

Figure 12: SLFLA output for flapping test, k =3

In all four tests, the learner's output is the correct Strictly Local transducer for the target rule. We thus conclude that the property of strict locality is not only a defining property of many phonological mappings, but one that can serve as a useful inductive principle of a learner generalizing such mappings from finite data. In the next section we will comment further on our use of artificial as opposed to natural language corpora, as well as the modifications to the SLFLA that will be necessary to prove that it identifies the class of SL functions in the limit from positive data.

5 Discussion

The tests of the SLFLA conducted for the current paper were, as noted above, based on artificial corpora. As Gildea & Jurafsky (1996) observed, a natural language corpus may not have enough data for a learner to correctly converge on the target generalization. However, 'not enough data' refers to quality, not quantity. In the case of OSTIA, they note that it needs to see six consonants followed by a 't' in order to learn the correct flapping rule. But of course no corpus of natural English data will include such an input, since phonotactic constraints make such a word impossible. In general, any corpus of natural language data is going to reflect not just the particular generalization (e.g., flapping, devoicing) the learner is trying to learn, but also phonotactic constraints on allowable sequences of sounds.

The artificial corpora, on the other hand, include all possible sequences over the given alphabet, which means the data includes sufficient information for the learner to generalize a *total* function. This appears to be crucial for both OSTIA and the SLFLA to learn a phonological mapping. Gildea & Jurafsky (1996)'s insight was to augment OSTIA with learning biases based on widely-recognized properties of phonological processes. The first, *context*, is precisely the bias incorporated into the SLFLA via the concept of strict locality. We have also, albeit manually, encoded something like *community* in our tests, by reducing the alphabets of the data to symbols representing natural classes. Clearly, it would be preferable for the learner to discover these natural classes on its own, though it may be argued that this takes place in a different stage of learning (i.e., before the learning of processes), in which case it makes sense for the data given to the SLFLA to be encoded in this way.

The *faithfulness* and *community* biases employed by Gildea & Jurafsky (1996) allowed OSTIA to overcome the mistakes it was making on account of the lack of data. For the SLFLA to likewise succeed on natural language corpora, we could likewise pursue additional modifications that allow the learner to work with incomplete data, but in such a way that we can still exactly identify its hypothesis space. The modifications employed by Gildea & Jurafsky (1996) make it unclear what class of functions the learner

can learn. It should be noted, however, that their objective was to demonstrate the utility of their learning biases, not to establish a theoretical result of identification in the limit. On the other hand, we think both our results and those of Gildea & Jurafsky (1996) raise interesting questions about how best to factor the learning problem. Especially in light of the OT assumption of Richness of the Base (Prince & Smolensky, 1993, 2004), which places no restrictions on the input to the phonological grammar, viewing a phonological process as a total function does not seem so incorrect. Indeed, even in the SPE formalism the constraints on the input (i.e., morpheme structure constraints) are separated from the processes, so that the rules only specify as much context as is necessary to capture the single process. Thus if the learner is tasked with learning an isolated process such as final devoicing, then perhaps it ignores phonotactic information and somehow induces from natural data a dataset that resembles the artificial corpora constructed for our simulations. In that sense the failure of a learner on a natural language corpus is neither unexpected nor problematic. Of course more work needs to be done to assess the plausibility of such an account.

6 Conclusion and next steps

We have presented an algorithm that learns the input-output mapping underlying local phonological rules by assuming its target is a Strictly Local function, which we have defined using the formalism of finite state transducers. We demonstrated that our learner can learn phonological processes such as final devoicing, fricative deletion, schwa-epenthesis, and flapping. More generally, we proved that given a closed learning sample, the learner can learn any phonological process that can be modeled with a Strictly Local function. This includes a range of phonological and morpho-phonological phenomena, including substitution, deletion, epenthesis, bounded metathesis, local partial reduplication, and general affixation.

Of course this list excludes non-local processes, such as vowel harmony with transparent vowels and long distance consonant harmony and dissimilation. However, we believe there is potential to adapt the approach presented here for learning local processes to this non-local domain. As discussed in §2.1, the Strictly Local languages, of which the Strictly Local functions are the functional counterpart, are just one region of a hierarchy of sub-regular formal languages (McNaughton & Papert, 1971; Rogers & Pullum, 2011; Rogers et al., 2013). It was shown by Heinz (2010) that other regions of this hierarchy can model long-distance phonological patterns at the phonotactic level. We suggest that additional functional classes can be defined that correspond to these sub-regular languages and model the non-local processes. The approach to learning presented here can then likewise be adapted to the learning of these non-local (but still sub-regular) functions.

In addition, the current implementation of the SLFLA learns the mapping that corresponds to the simultaneous application of a given rule. In some cases, however, applying the rule left-to-right would result in a different mapping. The simple example rule in (14) demonstrates.

(14) a $\Rightarrow$ b / a __

Given the input $aaaa$, the simultaneous application of (14) maps this form to $abba$, while left-to-right application yields $abaa$. As left-to-right application is necessary to capture spreading phenomena, it is both desirable and necessary for the learner to be able to acquire these mappings as well. We believe the SLFLA can indeed learn left-to-right mappings with a slight modification to its state merging strategy. Currently it merges states with the same $k-1$-length suffix on the input side of the incoming transition path. Learning the left-to-right application mapping would require merging states with the same $k-1$-length suffix on the *output* side (Kaplan & Kay, 1994; Hulden, 2009). We leave this modification for future work.

Lastly, we are currently working on a reimplementation of the SLFLA that we believe will identify the class of SL functions in the limit from positive data. This means the restriction to a closed learning sample can be lifted so that the learner succeeds not just on a characteristic sample of data but on all supersets of that sample. This is possible due to a change in the manner in which the algorithm selects the states it will merge (see Chandlee (2014) for details). The stronger theoretical result of identification in the limit will be further evidence for the utility of strict locality in phonological learning.

References

Angluin, Dana (1982). Inference of reversible languages. *Journal for the Association of Computing Machinery* 29:3, 741–765.

Beesley, Kenneth R. & Lauri Karttunen (2003). *Finite State Morphology*. CSLI Publications.

Bennett, William (2013). *Dissimilation, Consonant Harmony, and Surface Correspondence*. Ph.D. thesis, Rutgers University.

Chandlee, Jane (2014). *Strictly Local Phonological Processes*. Ph.D. thesis, University of Delaware.

Chandlee, Jane & Jeffrey Heinz (2012). Bounded copying is subsequential: Implications for metathesis and reduplication. *Proceedings of SIGMORPHON 12*.

Chandlee, Jane & Cesar Koirala (2014). Learning local phonological rules. *Proceedings of the 37th Penn Linguistics Colloquium*.

Chandlee, Jane, Angeliki Athanasopoulou & Jeffrey Heinz (2012). Evidence for classifying metathesis patterns as subsequential. *The Proceedings of the 29th West Coast Conference on Formal Linguistics*, Cascadilla, Somerville, MA, 303–309.

Chomsky, Noam (1956). Three models for the description of language. *IRE Transactions on Information Theory* 113–124.

Chomsky, Noam & Morris Halle (1968). *The Sound Pattern of English*. Harper & Row.

Gainor, Brian, Regine Lai & Jeffrey Heinz (2012). Computational characterizations of vowel harmony patterns and pathologies. *The Proceedings of the 29th West Coast Conference on Formal Linguistics*, Cascadilla, Somerville, MA, 63–71.

Gildea, Daniel & Daniel Jurafsky (1996). Learning bias and phonological-rule induction. *Computational Linguistics* :22:4, 497–530.

Gold, E. Mark (1967). Language identification in the limit. *Information and Control* :10, 447–474.

Hansson, Gunnar (2001). *Theoretical and Typological Issues in Consonant Harmony*. Ph.D. thesis, University of California, Berkeley.

Heinz, Jeffrey (2007). *The Inductive Learning of Phonotactic Patterns*. Ph.D. thesis, University of California, Los Angeles.

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

Heinz, Jeffrey (2010). Learning long-distance phonotactics. *Linguistic Inquiry* 41:4, 623–661.

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

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

Hulden, Mans (2009). *Finite-State Machine Construction Methods and Algorithms for Phonology and Morphology*. Ph.D. thesis, University of Arizona.

Jardine, Adam (2013). Tone is (computationally) different. Unpublished manuscript, University of Delaware.

Johnson, C. Douglas (1972). *Formal Aspects of Phonological Description*. Mouton.

Joseph, Brian D. & Irene Philippaki-Warburton (1987). *Modern Greek*. Croom Helm, Wolfeboro, NH.

Kaplan, Ronald M. & Martin Kay (1994). Regular models of phonological rule systems. *Computational Linguistics* :20, 371–387.

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

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

McNaughton, Robert & Seymour A. Papert (1971). *Counter-Free Automata*. MIT Press.

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

Mohri, Mehryar (1997). Finite-state transducers in language and speech processing. *Computational Linguistics* 23:2, 269–311.

Nevins, A. (2010). *Locality in Vowel Harmony*. MIT Press.

Oncina, José, Pedro García & Enrique Vidal (1993). Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* :15:5, 448–457.

Partee, Barbara, Alice ter Meulen & Robert Wall (1993). *Mathematical Methods in Linguistics*. Kluwer Academic Publishers, Dordrect, Boston, London.

Payne, Amanda (2013). Dissimilation as a subsequential process. Unpublished manuscript, University of Delaware.

Prince, Alan & Paul Smolensky (1993). Optimality theory: Constraint interaction in generative grammar. *Rutgers University Center for Cognitive Science Technical Report* 2.

Prince, Alan & Paul Smolensky (2004). *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell Publishing.

Riggle, Jason (2003). Nonlocal reduplication. *Proceedings of the 34th annual meeting of the North Eastern Linguistic Society*.

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

Rogers, James, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert & Sean Wibel (2013). Cognitive and sub-regular complexity. *Formal Grammar*, Springer, vol. 8036 of *Lecture Notes in Computer Science*, 90–108.

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

Suzuki, Keiichiro (1998). *A Typological Investigation of Dissimilation*. Ph.D. thesis, University of Arizona.

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