Learning Strictly Local Subsequential Functions

Jane Chandlee
University of Delaware
Department of Linguistics
and Cognitive Science
125 East Main Street
Newark, DE 19716
janemc@udel.edu

Rémi Eyraud
QARMA team
Laboratoire d'Informatique
Fondamentale
Marseille, France
remi.eyraud@
lif.univ-mrs.fr

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

Abstract

We define two proper subclasses of subsequential functions based on the concept of Strict Locality (McNaughton and Papert, 1971; Rogers and Pullum, 2011; Rogers et al., 2013) for formal languages. They are called Input and Output Strictly Local (ISL and OSL). We provide an automata-theoretic characterization of the ISL class and theorems establishing how the classes are related to each other and to Strictly Local languages. We give evidence that local phonological and morphological processes belong to these classes. Finally we provide a learning algorithm which provably identifies the class of ISL functions in the limit from positive data in polynomial time and data. We demonstrate this learning result on appropriately synthesized artificial corpora. We leave a similar learning result for OSL functions for future work and suggest future directions for addressing non-local phonological processes.

1 Introduction

In this paper we define two proper subclasses of subsequential functions based on the properties of the well-studied Strictly Local formal languages (McNaughton and Papert, 1971; Rogers and Pullum, 2011; Rogers et al., 2013). These are languages that can be defined with grammars of substrings of length k (called k-factors), such that a string is in the language only if its own k-factors are a subset of the grammar. These languages have also been characterized by Rogers and Pullum (2011) as those that have the property expressed in the following theorem (which can be taken as a defining property):

Theorem 1 (Suffix Substitution Closure)**.** *L is Strictly Local iff for all strings u_1, v_1, u_2, v_2, there exists $k \in \mathbb{N}$ such that for any string x of length $k-1$, if u_1xv_1, $u_2xv_2 \in L$, then $u_1xv_2 \in L$.*

These languages can model natural language phonotactic constraints which pick out contiguous substrings bounded by some length k (Heinz, 2007; Heinz, 2010). We define Input Strictly Local (ISL) and Output Strictly Local (OSL) functions which model phonological *processes* for which the target and triggering context are a bounded contiguous substring. Here our use of 'process' is not specific to rule-based grammatical formalisms (such as SPE (Chomsky and Halle, 1968)). ISL and OSL functions model *mappings* from underlying forms to surface forms, which are also the bedrock of constraint-based frameworks like Optimality Theory (Prince and Smolensky, 1993).

By showing that local phonological processes can be modeled with ISL (and OSL) functions, we provide the strongest computational characterization of the input-output mappings these processes represent. While it has been shown that phonological mappings describable with rules of the form A $\rightarrow$ B / C $\_\_$ D (where A, B, C, and D are regular languages) are regular (Johnson, 1972; Kaplan and Kay, 1994), and even subsequential (Chandlee and Heinz, 2012; Heinz and Lai, 2013), many logically possible regular and subsequential mappings are not plausible phonological mappings. Since these implausible mappings cannot be modeled with ISL or OSL functions, we provide a more precise notion of

Transactions of the Association for Computational Linguistics, vol. 2, pp. 491–503, 2014. Action Editor: Alexander Clark.
Submitted 4/2014; Revised 8/2014; Published November 1, 2014.

what constitues plausible phonological mappings.

In addition, we present the Input SL Function Learning Algorithm (ISLFLA) and prove that it identifies this class in the limit (Gold, 1967) in polynomial time and data (de la Higuera, 1997). Our approach follows previous work on learning subsequential transductions, namely Oncina et al. (1993), Oncina and Varò (1996), Castellanos et al. (1998), and Gildea and Jurafsky (1996). Oncina et al. (1993) present OSTIA (Onward Subsequential Transducer Inference Algorithm), an algorithm that learns the class of subsequential functions in the limit from positive data. OSTIA is only guaranteed to identify total functions exactly, but Oncina and Varò (1996) and Castellanos et al. (1998) present the modifications OSTIA-N, OSTIA-D, and OSTIA-R, which learn partial functions using negative data, domain information, and range information, respectively.

In terms of linguistic applications, Gildea and Jurafsky (1996) show that OSTIA fails to learn the phonological mapping of English flapping when given natural language data. The authors modified OSTIA with three learning heuristics (context, community, and faithfulness) and showed that the modified learner successfully learns flapping and several other phonological rules. *Context* encodes the idea that phonological changes depend on the context of the segment undergoing the change. *Community* gives the learner the ability to deduce that segments belonging to a natural class are likely to behave similarly. Lastly, *faithfulness*, by which underlying segments are assumed to be realized similarly on the surface, was encoded with a forced alignment between the input-output strings in the data set.[1] We believe this alignment removes OSTIA's guarantee that all subsequential functions are learned.

Similar to the approach of Gildea and Jurafsky (1996), our learner employs a context bias because it knows its target is an ISL function and therefore the transduction only involves bounded contiguous substrings. And similar to OSTIA-D (Oncina and Varò, 1996; Castellanos et al., 1998), the ISLFLA makes use of domain information, because it makes decisions based on the input strings of the data set. It also employs a faithfulness bias in terms of the property *onwardness* (see §2). The ISLFLA is supported by a theoretical result like Oncina et al. (1993), but learns a more restrictive class of mappings. We believe the theoretical results for this class will lead to new algorithms which include something akin to the community bias and that will succeed on natural language data while keeping strong theoretical results.

The proposed learner also builds on earlier work by Chandlee and Koirala (2014) and Chandlee and Jardine (2014) which also used strict locality to learn phonological processes but with weaker theoretical results. The former did not precisely identify the class of functions the learner could learn, and the latter could only guarantee learnability of the ISL functions with a closed learning sample.

The paper is organized as follows. §2 presents the mathematical notations to be used. §3 defines ISL and OSL functions, provides an automata-theoretic characterization for ISL, and establishes some properties of these classes. §4 demonstrates how these functions can model local phonological processes, including substitution, insertion, and deletion. §5 presents the ISLFLA, proves that it efficiently learns the class of ISL functions, and provides demonstrations. §6 concludes.

2 Preliminaries

The set of all possible finite strings of symbols from a finite alphabet Σ and the set of strings of length $\leq n$ are Σ^* and $\Sigma^{\leq n}$, respectively. The unique empty string is represented with λ. The length of a string w is $|w|$, so $|\lambda| = 0$. The set of prefixes of w, $\texttt{Pref}(w)$, is $\{p \in \Sigma^* \mid (\exists s \in \Sigma^*)[w = ps]\}$, and the set of suffixes of w, $\texttt{Suff}(w)$, is $\{s \in \Sigma^* \mid (\exists p \in \Sigma^*)[w = ps]\}$. For all $w \in \Sigma^*$ and $n \in \mathbb{N}$, $\texttt{Suff}^n(w)$ is the single suffix of w of length n if $|w| \geq n$; otherwise $\texttt{Suff}^n(w) = w$. If $w = ps$, then $ws^{-1} = p$ and $p^{-1}w = s$. The longest common prefix of a set of strings S, $\texttt{lcp}(S)$, is $p \in \cap_{w \in S}\texttt{Pref}(w)$ such that $\forall p' \in \cap_{w \in S}\texttt{Pref}(w), |p'| < |p|$.

A function f with domain A and co-domain B is written $f : A \rightarrow B$. We sometimes write $x \mapsto_f y$ for $f(x) = y$. When A and B are free monoids (like Σ^*), the input and output languages of a function f are the stringsets $\texttt{dom}(f) = \{x \mid (\exists y)[x \mapsto_f y]\}$ and $\texttt{image}(f) = \{y \mid (\exists x)[x \mapsto_f y]\}$, respectively.

Following Oncina et al. (1993), a *subsequen-*

[1] The alignment was similar to the string-edit distance mehod used in Hulden et al. (2011).

tial finite state transducer (SFST) is a 6-tuple $(Q, q_0, \Sigma, \Gamma, \delta, \sigma)$, where Q is a finite set of states, Σ and Γ are finite alphabets, $q_0 \in Q$ is the initial state, $\delta \subseteq Q \times \Sigma \times \Gamma^* \times Q$ is a set of edges, and $\sigma : Q \to \Gamma^*$ is the final output function that maps states to strings that are appended to the output if the input ends in that state. δ recursively defines a mapping δ^*: $(q, \lambda, \lambda, q) \in \delta^*$; if $(q, u, v, q') \in \delta^*$ and $(q', \sigma, w, q'') \in \delta$ then $(q, u\sigma, vw, q'') \in \delta^*$. SFSTs are deterministic, which means their edges have the following property: $[(q, a, u, r), (q, a, v, s) \in \delta \Rightarrow u = v \wedge r = s]$. Hence, we also refer to δ as the transition *function*, and $\forall (q, a, u, r) \in \delta$, we let $\delta_1(q, a) = u$ and $\delta_2(q, a) = r$.

The relation that a SFST $\mathcal{T}$ recognizes/accepts/generates is

$$R(\mathcal{T}) = \Big\{ (x, yz) \in \Sigma^* \times \Gamma^* \mid (\exists q \in Q) \big[(q_0, x, y, q) \in \delta^* \wedge z = \sigma(q)\big] \Big\}.$$

Since SFSTs are deterministic, the relations they recognize are functions. *Subsequential functions* are defined as those describable with SFSTs.

For any function $f : \Sigma^* \to \Gamma^*$ and $x \in \Sigma^*$, let the *tails* of x with respect to f be defined as

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

If strings $x_1, x_2 \in \Sigma^*$ have the same set of tails with respect to a function f, they are *tail-equivalent* with respect to f and we write $x_1 \sim_f x_2$. Clearly, $\sim_f$ is an equivalence relation which partitions Σ^*.

Theorem 2 (Oncina and Garcia, 1991)**.** *A function f is* subsequential *iff $\sim_f$ partitions Σ^* into* finitely *many blocks.*

The above theorem can be seen as the functional analogue to the MyHill-Nerode theorem for regular languages. Recall that for any stringset L, the tails of a word w w.r.t. L is defined as $\texttt{tails}_L(w) = \{u \mid wu \in L\}$. These tails can be used to partition Σ^* into a finite set of equivalence classes iff L is regular. Furthermore, these equivalence classes are the basis for constructing the (unique up to isomorphism) smallest deterministic acceptor for a regular language. Likewise, Oncina and Garcia's proof of Theorem 2 shows how to construct the (unique up to isomorphism) smallest SFST for a subsequential function f. We refer to this transducer as the canonical transducer for f and denote it with $\mathcal{T}_f^{\mathsf{C}}$. The states of $\mathcal{T}_f^{\mathsf{C}}$ are in one-to-one correspondence with $\texttt{tails}_f(x)$ for all $x \in \Sigma^*$ (Oncina and García, 1991). To construct $\mathcal{T}_f^{\mathsf{C}}$ first let, for all $x \in \Sigma^*$ and $a \in \Sigma$, the contribution of a w.r.t. x be $\texttt{cont}_f(a, x) = \texttt{lcp}(f(x\Sigma^*)^{-1}\texttt{lcp}(f(xa\Sigma^*))$. Then,

- $Q = \{\texttt{tails}_f(x) \mid x \in \Sigma^*\}$,
- $q_0 = \texttt{tails}_f(\lambda)$,
- $\forall\, \texttt{tails}_f(x) \in Q$, $\sigma(\texttt{tails}_f(x)) = \texttt{lcp}(f(x\Sigma^*))^{-1} f(x)$ if $x \in \texttt{dom}(f)$ and is undefined otherwise,
- $\delta = \big\{(\texttt{tails}_f(x), a, \texttt{cont}_f(a, x), \texttt{tails}_f(xa))\big\}$.

$\mathcal{T}_f^{\mathsf{C}}$ has an important property called *onwardness*.

Definition 1 (onwardness)**.** *For all $q \in Q$ let the outputs of the edges out of q be $\texttt{outs}(q) = \big\{u \mid (\exists a \in \Sigma)(\exists r \in Q)[(q, a, u, r) \in \delta]\big\}$. A SFST $\mathcal{T}$ is* onward *if for all $q \in Q \setminus \{q_0\}$, $\texttt{lcp}(\texttt{outs}(q)) = \lambda$.*

Informally, this means that the writing of output is never delayed. Readers are referred to Oncina and García (1991), Oncina et al. (1993), and Mohri (1997) for more on SFSTs.

3 Strictly Local Functions

In this section we define Input and Output Strictly Local functions and provide properties of these classes. These definitions are analogous to the language-theoretic definition of Strictly Local languages (Theorem 1) (Rogers and Pullum, 2011; Rogers et al., 2013).

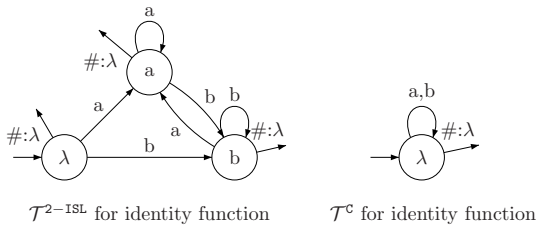
Definition 2 (Input Strictly Local Function)**.** *A function f is Input Strictly Local (ISL) if there is a k such that for all $u_1, u_2 \in \Sigma^*$, if $\texttt{Suff}^{k-1}(u_1) = \texttt{Suff}^{k-1}(u_2)$ then $\texttt{tails}_f(u_1) = \texttt{tails}_f(u_2)$.*

The theorem below establishes an automata-theoretic characterization of ISL functions.

Theorem 3. *A function f is ISL iff there is some k such that f can be described with a SFST for which*

1. *$Q = \Sigma^{\leq k-1}$ and $q_0 = \lambda$*
2. *$(\forall q \in Q, \forall a \in \Sigma, \forall u \in \Gamma^*)$ $\big[(q, a, u, q') \in \delta \Rightarrow q' = \texttt{Suff}^{k-1}(qa)\big]$.*

This theorem helps make clear how ISL functions are Markovian: the output for input symbol a depends on the last $(k-1)$ input symbols. Also, since the transducer defined in Theorem 3 is deterministic, it is unique and we refer to it as $\mathcal{T}_f^{\texttt{ISL}}$. $\mathcal{T}_f^{\texttt{ISL}}$ may not be isomorphic to $\mathcal{T}_f^{\texttt{C}}$. Figure 1 shows $\mathcal{T}_f^{\texttt{ISL}}$ (with $k=2$) and $\mathcal{T}_f^{\texttt{C}}$ for the identity function.[2]



$\mathcal{T}^{2-\texttt{ISL}}$ for identity function $\mathcal{T}^{\texttt{C}}$ for identity function

Figure 1: Non-isomorphic $\mathcal{T}_f^{\texttt{ISL}}$ (left) and $\mathcal{T}_f^{\texttt{C}}$ (right)

Before we present the proof of Theorem 3, we make the following remarks, and then prove a lemma.

Remark 1. *For all $n, m \in \mathbb{N}$ with $n \leq m$, and for all $w \in \Sigma^*$, $\texttt{Suff}^n(\texttt{Suff}^m(w)) = \texttt{Suff}^n(w)$ since both w and $\texttt{Suff}^m(w)$ share the same n-long suffix.*

Remark 2. *For all $w, v \in \Sigma^*, k \in \mathbb{N}$, $\texttt{Suff}^{k-1}\big(\texttt{Suff}^{k-1}(w)v\big) = \texttt{Suff}^{k-1}(wv)$. This is a direct consequence of Remark 1.*

Lemma 1. *Let $\mathcal{T}$ be a SFST constructed as in Theorem 3. If (λ, w, u, q) in δ^* then $q = \texttt{Suff}^{k-1}(w)$.*

Proof. The proof is by induction on δ^*. The base case follows from the facts that $(\lambda, \lambda, \lambda, \lambda) \in \delta^*$ (by definition of δ^*), and that $\lambda = \texttt{Suff}^{k-1}(\lambda)$.

Next assume the inductive hypothesis (IH) that there exists $w \in \Sigma^*$, $u \in \Gamma^*$, such that $(\lambda, w, u, q) \in \delta^*$ implies $q = \texttt{Suff}^{k-1}(w)$. We show $\forall a \in \Sigma$, $\forall x \in \Gamma^*$ that $(\lambda, wa, ux, q') \in \delta^*$ implies $q' = \texttt{Suff}^{k-1}(wa)$. Now $(\lambda, w, u, \texttt{Suff}^{k-1}(w)) \in \delta^*$ (by IH) so $(\texttt{Suff}^{k-1}(w), a, x, q') \in \delta$. By construction of $\mathcal{T}$, $q' = \texttt{Suff}^{k-1}(\texttt{Suff}^{k-1}(w)a)$, which by Remark 2, equals $\texttt{Suff}^{k-1}(wa)$. □

Proof of Theorem 3. ($\Leftarrow$) Fix $k \in \mathbb{N}$ and let f be a function described by $\mathcal{T} = \{Q, q_0, \Sigma, \Gamma, \delta, \sigma\}$ constructed as in Theorem 3. Let $w_1, w_2 \in \Sigma^*$ such that $\texttt{Suff}^{k-1}(w_1) = \texttt{Suff}^{k-1}(w_2)$. By Lemma 1, both w_1 and w_2 lead to the same state and therefore $\texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$. Thus f is k-ISL.

($\Rightarrow$) Consider any ISL function f. Then there is a k such that $(\forall w_1, w_2 \in \Sigma^*)[\texttt{Suff}^{k-1}(w_1) = \texttt{Suff}^{k-1}(w_2) \Rightarrow \texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)]$. We show that $\mathcal{T}_f^{\texttt{ISL}}$ exists. Since $\Sigma^{\leq k-1}$ is a finite set, the equivalence relation $\sim_f$ partitions Σ^* into at most $|\Sigma^{\leq k-1}|$ blocks. Hence by Theorem 2, f is subsequential and a canonical subsequential transducer $\mathcal{T}_f^{\texttt{C}} = \{Q_c, q_{0c}, \Sigma, \Gamma, \delta_c, \sigma_c\}$ exists.

Construct $\mathcal{T} = \{Q, q_0, \Sigma, \Gamma, \delta, \sigma\}$ as follows:

- $Q = \Sigma^{\leq k-1}$ and $q_0 = \lambda$
- $\forall q \in Q, \sigma(q) = \sigma_c\big(\texttt{tails}_f(q)\big)$ if $f(q)$ is defined, else $\sigma(q)$ is undefined.
- $\forall q \in Q$, $\forall a \in \Sigma$, $\forall u \in \Gamma^*$, $\big(q, a, u, \texttt{Suff}^{k-1}(qa)\big) \in \delta$ iff $\big(\texttt{tails}_f(q), a, u, \texttt{tails}_f(qa)\big) \in \delta_c$.

First, by its construction, the states and transitions of $\mathcal{T}$ meet requirements (1) and (2) of Theorem 3.

Next, we show that $\mathcal{T}$ computes the same function as $\mathcal{T}_f^{\texttt{C}}$. We first establish by induction on δ^* the following ($\star$): $\forall w \in \Sigma^*$, $\forall u \in \Gamma^*$, $\big(q_0, w, u, \texttt{Suff}^{k-1}(w)\big) \in \delta^*$ iff $\big(\texttt{tails}_f(\lambda), w, u, \texttt{tails}_f(w)\big) \in \delta_c^*$.

Clearly $\big(\texttt{tails}_f(\lambda), \lambda, \lambda, \texttt{tails}_f(\lambda)\big) \in \delta_c^*$ and $\big(\lambda, \lambda, \lambda, \lambda\big) \in \delta^*$ by definition of the extended transition function. Thus the base case is satisfied.

Next assume the inductive hypothesis (IH) for $w \in \Sigma^*$ and $u \in \Gamma^*$. We show $\forall a \in \Sigma$ and $\forall x \in \Gamma^*$ that $\big(\lambda, wa, ux, \texttt{Suff}^{k-1}(wa)\big) \in \delta^*$ iff $\big(\texttt{tails}_f(\lambda), wa, ux, \texttt{tails}_f(wa)\big) \in \delta_c^*$.

Suppose $\big(\texttt{tails}_f(\lambda), wa, ux, \texttt{tails}_f(wa)\big) \in \delta_c^*$. By IH, $\big(\texttt{tails}_f(\lambda), w, u, \texttt{tails}_f(w)\big) \in \delta_c^*$; hence $\big(\texttt{tails}_f(w), a, x, \texttt{tails}_f(wa)\big) \in \delta_c$.

Let $q = \texttt{Suff}^{k-1}(w)$. Observe that $\texttt{Suff}^{k-1}(w) = \texttt{Suff}^{k-1}(q)$ by Remark 1. Consequently, since f is k-ISL, $\texttt{tails}_f(w) = \texttt{tails}_f(q)$. Similarly, $\texttt{Suff}^{k-1}(wa) = \texttt{Suff}^{k-1}(qa)$ and thus $\texttt{tails}_f(wa) = \texttt{tails}_f(qa)$. By substitution then, we have $\big(\texttt{tails}_f(\lambda), w, u, \texttt{tails}_f(q)\big) \in \delta_c^*$ and $\big(\texttt{tails}_f(q), a, x, \texttt{tails}_f(qa)\big) \in \delta_c$.

By construction of $\mathcal{T}$, $\big(q, a, x, \texttt{Suff}^{k-1}(qa)\big) \in \delta$. By IH, $\big(\lambda, w, u, \texttt{Suff}^{k-1}(w)\big) = \big(\lambda, w, u, q\big) \in \delta^*$. Thus, $\big(\lambda, wa, ux, \texttt{Suff}^{k-1}(qa)\big) \in \delta^*$ which is the same as saying $\big(\lambda, wa, ux, \texttt{Suff}^{k-1}(wa)\big) \in \delta^*$.

Conversely, consider any $a \in \Sigma$ and $x \in \Gamma^*$ and suppose $\big(\lambda, wa, ux, \texttt{Suff}^{k-1}(wa)\big) \in \delta^*$.

[2] In all figures, single-symbol transition labels indicate that the input and output are identical, and the final output function is represented as a transition on the end-of-word symbol #.

By IH, $\big(\lambda, w, u, \texttt{Suff}^{k-1}(w)\big) \in \delta^*$; hence $\big(\texttt{Suff}^{k-1}(w), a, x, \texttt{Suff}^{k-1}(wa)\big) \in \delta$. As before, letting $q = \texttt{Suff}^{k-1}(w)$, it follows that $\texttt{Suff}^{k-1}(w) = \texttt{Suff}^{k-1}(q)$ so $\texttt{tails}_f(w) = \texttt{tails}_f(q)$, and $\texttt{Suff}^{k-1}(wa) = \texttt{Suff}^{k-1}(qa)$ so $\texttt{tails}_f(wa) = \texttt{tails}_f(qa)$. Therefore, $\big(q, a, x, \texttt{Suff}^{k-1}(qa)\big) \in \delta$.

By construction of $\mathcal{T}$, this means $\big(\texttt{tails}_f(w), a, x, \texttt{tails}_f(wa)\big) \in \delta_c$. By IH, $\big(\texttt{tails}_f(\lambda), w, u, \texttt{tails}_f(w)\big) \in \delta_c^*$. By the definition of the extended transition function, it follows that $\big(\texttt{tails}_f(\lambda), wa, ux, \texttt{tails}_f(wa)\big) \in \delta_c^*$. This completes the induction, establishing ($\star$).

Let $w \in \Sigma^*$. If $f(w)$ is defined then $f(w) = uv$ where $\big(\texttt{tails}_f(\lambda), w, u, \texttt{tails}_f(w)\big) \in \delta_c^*$ and $\sigma_c\big(\texttt{tails}_f(w)\big) = v$. From ($\star$), we know $\big(\lambda, w, u, \texttt{Suff}^{k-1}(w)\big) \in \delta^*$. Also, $\sigma\big(\texttt{Suff}^{k-1}(w)\big) = \sigma_c\big(\texttt{tails}_f(\texttt{Suff}^{k-1}(w))\big)$. Since f is k-ISL, $\texttt{tails}_f(w) = \texttt{tails}_f(\texttt{Suff}^{k-1}(w))$ (cf. Remark 1). Thus $\sigma\big(\texttt{Suff}^{k-1}(w)\big) = \sigma_c\big(\texttt{tails}_f(w)\big) = v$. Therefore $\mathcal{T}(w) = uv$ also.

If $f(w)$ is not defined then there are two cases. Case 1: $\big(\texttt{tails}_f(\lambda), w, u, \texttt{tails}_f(w)\big) \notin \delta_c^*$. By the above lemma, it follows $\big(\lambda, w, u, \texttt{Suff}^{k-1}(w)\big) \notin \delta^*$ and so $\mathcal{T}(w)$ is also defined. Case 2: $\big(\texttt{tails}_f(\lambda), w, u, \texttt{tails}_f(w)\big) \in \delta_c^*$ but $\sigma_c(\texttt{tails}_f(w))$ is undefined. By construction of $\mathcal{T}$, $\sigma(\texttt{Suff}^{k-1}(w))$ is also undefined. □

Similar to Definition 2 above, the following definition of OSL functions says that if the outputs of two input strings share the same suffix of length $(k-1)$, then those inputs have the same tails.

Definition 3 (Output Strictly Local Function)**.** *A function f is Output Strictly Local (OSL) if there is a k such that for all $u_1, u_2 \in \Sigma^*$, if $\texttt{Suff}^{k-1}(f(u_1)) = \texttt{Suff}^{k-1}(f(u_2))$ then $\texttt{tails}_f(u_1) = \texttt{tails}_f(u_2)$.*

The automata-theoretic characterization of this class is left as future work.

Next, some properties of ISL and OSL functions are identified. The proofs of the following theorems will refer to the example SFSTs in Figure 2.

First we establish that ISL and OSL functions are proper subclasses of the subsequential functions. (They are subclasses by definition.)

Theorem 4. *There are subsequential functions which are neither ISL nor OSL.*

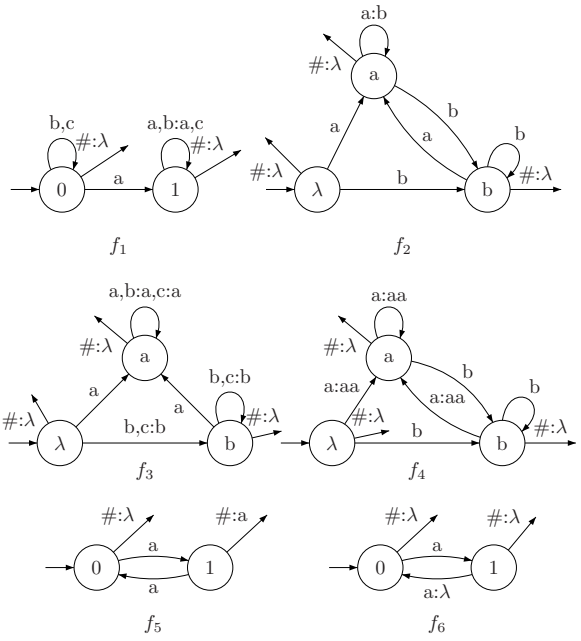


Figure 2: Examples for proofs of Theorems 4-7

Proof. Consider the subsequential function f_1 in Figure 2 and choose any $k \in \mathbb{N}$. Since $f_1(bc^kb) = bc^kb$ and $f_1(ac^kb) = ac^ka$ it follows that $\texttt{tails}_{f_1}(bc^k) \neq \texttt{tails}_{f_1}(ac^k)$ even though inputs bc^k and ac^k share a common suffix of length $(k-1)$. Likewise, the outputs of these inputs $f_1(bc^k) = bc^k$ and $f_1(ac^k) = ac^k$ share a common suffix of length $(k-1)$, but the tails of these inputs, as mentioned, are distinct. Hence by Definitions 2 and 3, there is no k for which f_1 is ISL or OSL. □

Theorem 5. *The class of ISL functions is incomparable to the class of OSL functions.*

Proof. Consider f_2 in Figure 2. This function is ISL by Theorem 3. However it is not OSL. Consider any $k \in \mathbb{N}$. Observe that $f_2(aa^k) = f_2(ab^k) = ab^k$ and so the outputs share the same suffix of length $(k-1)$. However, $\texttt{tails}_{f_2}(aa^k) \neq \texttt{tails}_{f_2}(ab^k)$ since $(a, b) \in \texttt{tails}_{f_2}(aa^k)$ but $(a, a) \in \texttt{tails}_{f_2}(ab^k)$.

Similarly, consider f_3 in Figure 2. This function is 2-OSL because inputs mapped to outputs which end in a have the same tails (i.e., they all lead to state a), and inputs mapped to outputs ending in b have the same tails.

However, f_3 is not ISL. Consider any $k \in \mathbb{N}$. The inputs cb^k and ab^k share the same suffix of length $(k-1)$. However, $\texttt{tails}_{f_3}(cb^k) \neq$

$\texttt{tails}_{f_3}(ab^k)$ since $(b, b) \in \texttt{tails}_{f_3}(cb^k)$ but $(b, a) \in \texttt{tails}_{f_3}(ab^k)$.

Finally, the two classes are obviously not disjoint, since the identity function is both ISL and OSL. □

Theorem 6. *The output language of an ISL or OSL function is not necessarily a Strictly Local language.*

Proof. Consider f_4 in Figure 2. By Theorem 3, it is ISL. It is also 2-OSL since inputs mapped to outputs which end in a have the same tails (i.e., they all lead to state a). Similarly, inputs mapped to outputs ending in b have the same tails.

Let $k \in \mathbb{N}$. Observe that $f_4(a^k) = a^{2k}$ and further that there is no input x and k such that $f_4(x) = a^{2k+1}$. Since $a^{2k} = a^{k-1}a^k a = a^{k-2}a^k aa$, if the output language of f_4 were SL it would follow, by Theorem 1, that $a^{k-1}a^k aa = a^{2k+1}$ also belongs to the output language. But there is no input in Σ^* which f_4 maps to an odd sequence of a's. □

Theorem 7. *If either the output or input language of a subsequential function f is Strictly Local then it does not follow that f belongs to either ISL or OSL.*

Proof. Let $\Sigma = \Gamma = \{a\}$ and consider the function f_5 which, for all $n \in \mathbb{N}$, maps a^n to a^n if n is even and a^n to $a^n a$ if n is odd. f_5 is subsequential, as shown in Figure 2, and its domain, a^*, is a Strictly Local language. However, f_5 is neither ISL nor OSL for any k since the $\texttt{tails}$ of a^{2k} includes (λ, λ) but the $\texttt{tails}$ of aa^{2k} includes (λ, a).

Next consider f_6, which for all $n \in \mathbb{N}$ maps a^n to a^m where m = (n div 2) if n is even and m = (n div 2)+1 if n is odd. f_6 is subsequential, as shown in Figure 2, and the $\texttt{image}(f) = a^*$ is Strictly Local. However, f is neither ISL nor OSL for any k since the $\texttt{tails}$ of a^{2k} includes (a, a) but the $\texttt{tails}$ of aa^{2k} includes (a, λ). □

4 Relevance to Phonology

This section will briefly discuss the range of phonological processes that can be modeled with ISL and/or OSL functions by providing illustrative examples for three common, representative processes. These are shown in (1) with SPE-style rules.

(1) a. Devoicing: D → T / __ #
b. Epenthesis: ∅ → ə / {l, r} __ [-coronal]
c. Deletion: {θ, ð} → ∅ / __ {θ, s}

See Chandlee (2014) for more details.

First, consider the process of final obstruent devoicing (1-a), attested in German and other languages. This process causes an underlying voiced obstruent in word-final position to surface as its voiceless counterpart. In (1-a), D abbreviates the set of voiced obstruents and T the voiceless obstruents.[3]

The mapping from underlying form to surface form that this process describes is represented with the 2-ISL function in Figure 3 (note N represents any sonorant).

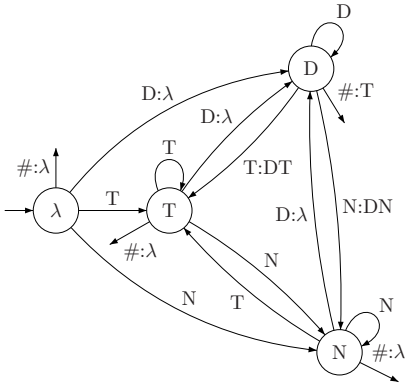


Figure 3: $\mathcal{T}_f^{\texttt{ISL}}$ for (1-a) with $k = 2$ and $\Sigma = \{\text{T, D, N}\}$

In addition to substitution processes like (1-a), another common type of process is insertion of a segment, such as the ə-epenthesis process attested in Dutch (Warner et al., 2001). This process inserts a schwa between a liquid and a non-coronal consonant, as specified by (1-b). Using L to represent the liquids {l, r} and K to represent any non-coronal consonant, Figure 4 presents $\mathcal{T}_f^{\texttt{ISL}}$ for this process. Following Beesley and Karttunen (2003), the symbol '?' represents any segment of the alphabet other than the ones for which transitions are defined.[4]

Lastly, an example deletion process from Greek (Joseph and Philippaki-Warburton, 1987) deletes interdental fricatives before /θ/ or /s/, as in (1-c). Figure 5 presents $\mathcal{T}_f^{\texttt{ISL}}$ for this process.

[3] These abbreviations serve to improve the readability of the FST - it is assumed that the transduction replaces a given voiced obstruent with its voiceless counterpart (not with any voiceless obstruent).

[4] This means Figure 4 is actually a shorthand for $\mathcal{T}_f^{\texttt{ISL}}$, which would otherwise have a distinct state for each symbol now represented with '?'.

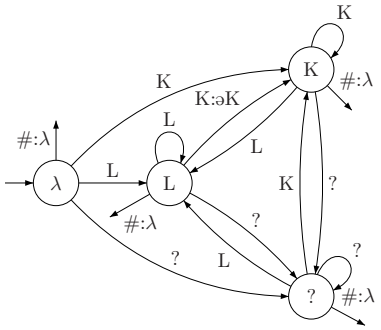


Figure 4: $\mathcal{T}_f^{\mathtt{ISL}}$ for (1-b) with $k = 2$ and $\Sigma = \{\text{L, K, ?}\}$

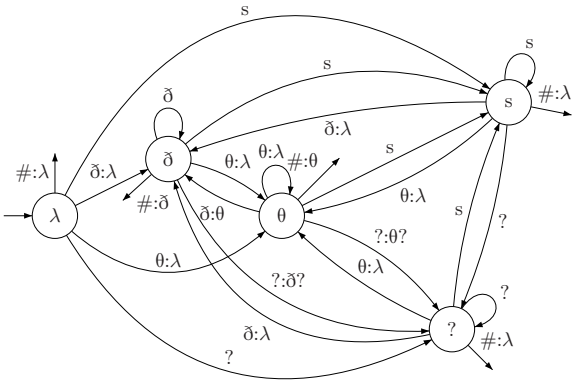


Figure 5: $\mathcal{T}_f^{\mathtt{ISL}}$ for (1-c) with $k = 2$ and $\Sigma = \{\theta, \eth, \text{s, ?}\}$

The German, Dutch, and Greek examples demonstrate how ISL functions can be used to model the input-output mapping of a phonological rule. Beyond substitution, insertion, and deletion, it is shown in Chandlee (2014) that ISL and/or OSL functions can also model many metathesis patterns, specifically those for which there is an upper bound on the amount of material that intervenes between a segment's original and surface positions (this appears to include all synchronic patterns). In addition, morpho-phonological processes such as local partial reduplication (i.e., in which the reduplicant is affixed adjacent to the portion of the base it was copied from) and general affixation are also shown to be ISL or OSL. More generally, we currently conjecture that a SPE-style rewrite rule of the form A $\rightarrow$ B/ C__ D which applies simultaneously (left-to-right) describes an Input (Output) Strictly Local function iff there is a k such that for all $w \in CAD$ it is the case that $|w| \leq k$. We refer readers to Kaplan and Kay (1994) and Hulden (2009) for more on how SPE rules and application modes determine mappings.

In contrast, non-local (long-distance) processes such as vowel harmony with transparent vowels (Gainor et al., 2012; Heinz and Lai, 2013), long distance consonant harmony (Hansson, 2001; Rose and Walker, 2004; Luo, 2013) and dissimilation (Suzuki, 1998; Bennett, 2013; Payne, 2013), unbounded displacement/metathesis (Chandlee et al., 2012; Chandlee and Heinz, 2012), non-local partial reduplication (Riggle, 2003), and some tonal patterns (Jardine, 2013) cannot be modeled with ISL or OSL functions. In §6 we comment on the potential for adapting the current analysis to non-local mappings.

The next section presents a learning algorithm for ISL functions, the ISLFLA. The development of a corresponding algorithm for OSL functions is the subject of ongoing work, but see §6 for comments.

5 Learning Input Strictly Local Functions

We now present a learning algorithm for the class of ISL functions that uses its defining property as an inductive principle to generalize from a finite amount of data to a possibly infinite function. This learner begins with a prefix tree representation of this data and then generalizes by merging states.[5] Its state merging criterion is based on the defining property of ISL functions: two input strings with the same suffix of length $(k-1)$ have the same tails. The next section explains in detail how the algorithm works.

5.1 The Algorithm: ISLFLA

Given a finite dataset D of input-output string pairs (w, w') such that $f(w) = w'$, where f is the target function, the learner tries to build $\mathcal{T}_f^{\mathtt{ISL}}$. The dataset is submitted to the learner in the form of a prefix tree transducer (PTT), which is defined in Definition 4.

Definition 4 (Prefix Tree Transducer)**.** *A* prefix tree transducer *for finite set* $D = \{(w, w') \mid f(w) = w'\}$ *for some* f *is* $PTT(D) = (Q, q_0, \Sigma, \Gamma, \delta, \sigma)$*, where*

- $Q = \bigcup\{\mathtt{Pref}(w) | (w, w') \in D\}$ *and* $q_0 = \lambda$
- $\delta = \{(u, a, \lambda, ua) \mid u, ua \in Q\}$
- $\sigma(w) = w'$ *for all* $(w, w') \in D$

[5] This general strategy is common in grammatical inference. See Angluin (1982), Heinz (2009), and de la Higuera (2010).

As an example, the sample of data in (2) exemplifies the final devoicing rule in (1-a). Figure 6 gives the PTT for this data.

(2) {(D, T), (DD, DT), (DNT, DNT), (NND, NNT), (TDN, TDN)}

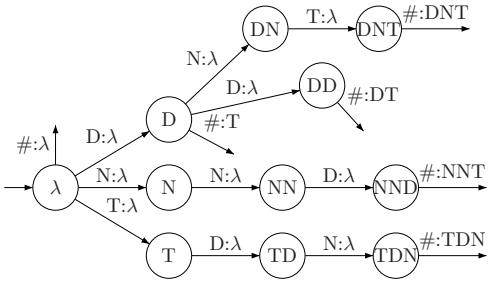


Figure 6: PTT for the data in (2)

Given such a PTT, the learner's first step is to make it *onward*. In the PTT, the output string is withheld until the end of the input (i.e., #) is reached. In the onward version (shown in Figure 7), output is advanced as close to the root as possible. This involves determining the `lcp` of all the output strings of all outgoing transitions of a state (starting from the leaves) and suffixing that `lcp` to the output of the single incoming transition of the state.

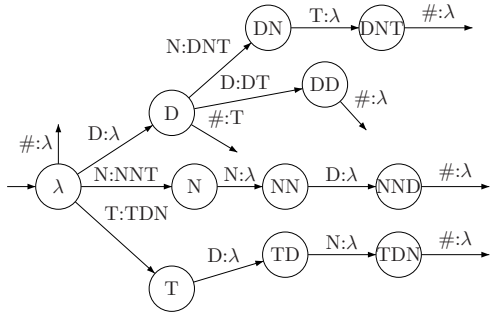


Figure 7: Onward version of the PTT in Figure 6

Once the learner has constructed this onward representation of the data, it begins to merge states, using two nested loops. The outer loop proceeds through the entire state set (ordered lexicographically) and merges each state with the state that corresponds to its $(k-1)$-length suffix. For example, for final devoicing $k = 2$, so each state will be merged with the state that represents its final symbol. This merging may introduce non-determinism, which must be removed since by definition $\mathcal{T}_f^{\texttt{ISL}}$ is deterministic. Non-determinism is removed in the inner loop with additional state merges.

Consider the situation depicted on the lefthand side of Figure 8, which resulted from a previous merge. The non-determinism could be resolved by

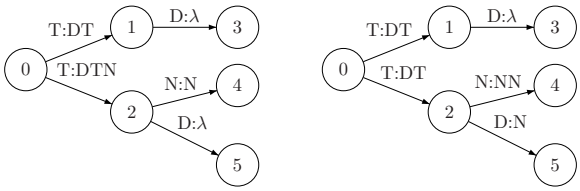


Figure 8: Before (left) and after (right) `pushback`

merging states 1 and 2, except for the fact that the output strings of the two transitions differ. Before merging 1 and 2, therefore, the learner performs an operation called `pushback` that retains the `lcp` of the two output strings and prefixes what's left to all output strings of all outgoing transitions from the respective destination states.

Definition 5 (`pushback` (Oncina et al., 1993))**.** *For SFST* $\mathcal{T} = (Q, q_0, \Sigma, \Gamma, \delta, \sigma)$*,* $v \in \Sigma^*$*, and* $e = (q, a, w, q') \in \delta$*,* $\texttt{pushback}(\mathcal{T}, v, e) = (Q, q_0, \Sigma, \Gamma, \delta', \sigma)$ *where* $\delta' = (\delta \cup \{q, a, wv^{-1}, q')\} \cup \{(q', b, vz, q'') | (q', b, z, q'') \in \delta\}) \setminus (\{(q', b, z, q'')\} \cup \{e\})$.

In the example in Figure 8, `pushback` is applied to the edges $(0, T, DT, 1)$ and $(0, T, DTN, 2)$. Only the `lcp` of the output strings, which is DT, is retained as the output string of both edges. The remainder (which is λ and N, respectively) is prefixed to the output string of all transitions leaving the respective destination state. The result is shown on the right-hand side of Figure 8. Essentially, `pushback` 'undoes' onwardness when needed.

After `pushback`, states 1 and 2 can be merged. This removes the initial non-determinism but creates new non-determinism. The inner loop iterates until all non-determinism is resolved, after which the outer loop continues with the next state in the order. If the inner loop encounters non-determinism that cannot be removed, the ISLFLA exits with a message indicating that the data sample is insufficient.

Non-removable non-determinism occurs if and only if the situation depicted on the left in Figure 9 obtains. The normal procedure for removing non-

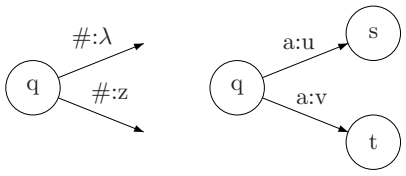


Figure 9: Irreparable non-determinism (left) and non-determinism from an outer loop merge (right).

determinism cannot be applied in this case. Assuming $z \neq \lambda$, all of z would have to be pushed back, but since this transition has no destination state there is nowhere for z to go. OSTIA handles this situation by rejecting the outer loop merge that led to it, restoring the FST to its state before that merge and moving on to the next possible merge. But the ISLFLA cannot reject merges. If it could, the possibility would arise that two states with the same $(k-1)$-length suffix would remain distinct in the final FST the learner outputs. Such a FST would not (by definition) be ISL. Therefore, the algorithm is at an impasse: rejecting a merge can lead to a non-ISL FST, while allowing it can lead to a non-subsequential (hence non-ISL) FST. It therefore terminates. Below is pseudo-code for the ISLFLA.

Data: $PTT(D)$
Result: $\mathcal{T}_f^{\mathtt{ISL}}$
$\tau = \mathtt{onward}(PTT)$
$q = \mathtt{next}(Q_\tau, \mathtt{first}(Q_\tau))$
while $q < \mathtt{last}(Q_\tau)$ **do**
  $\mathtt{merge}(q, \mathtt{Suff}^{k-1}(q))$
  while $\exists (q, a, x, q_1), (q, a, y, q_2) \in \delta_\tau$ **do**
    if $a = \#$ *and* $x \neq y$ **then**
      exit 'Insufficient data'
    else
      $\mathtt{pushback}(\tau, (q, a, x, q_1), \mathtt{lcp}(x, y))$
      $\mathtt{pushback}(\tau, (q, a, y, q_2), \mathtt{lcp}(x, y))$
      $\mathtt{merge}(q_1, q_2)$
  $q = \mathtt{next}(Q_\tau, q)$
return τ

Algorithm 1: Pseudo-code for the ISLFLA

5.2 Learning Results

Here, we present a proof that the ISLFLA identifies the class of ISL functions in the limit from positive data, in the sense of Gold (1967), with polynomial bounds on time and data (de la Higuera, 1997). First, we establish the notion of characteristic sample.

Definition 6 (Characteristic sample)**.** *A sample CS is characteristic of a function f for an algorithm $\mathfrak{A}$ if for all samples S s.t. $CS \subseteq S$, $\mathfrak{A}$ returns a representation τ such that for all $x \in \mathtt{dom}(f)$, $\tau(x) = f(x)$, and for all $x \notin \mathtt{dom}(f)$, $\tau(x)$ is not defined.*

We can now define the learning criteria.

Definition 7 (Identification in polynomial time and data (de la Higuera, 1997))**.** *A class $\mathbb{F}$ of functions is identifiable in polynomial time and data using a class $\mathbb{T}$ of representations iff there exist an algorithm $\mathfrak{A}$ and two polynomials $p()$ and $q()$ such that:*

1. *Given a sample S of size m for $f \in \mathbb{F}$, $\mathfrak{A}$ returns a hypothesis in $\mathcal{O}(p(m))$ time;*
2. *For each representation τ of size k of a function $f \in \mathbb{F}$, there exists a characteristic sample CS of f for $\mathfrak{A}$ of size at most $\mathcal{O}(q(k))$.*

Essentially the proof for convergence follows from the fact that given a sufficient sample the algorithm merges *all* and *only* states with the same $(k-1)$-length suffix.

Clearly, merges in the outer loop only involve states with the same $(k-1)$-length suffix. This is also the case for inner loop merges. Consider the scenario depicted on the right in Figure 9, in which q is a state created by an outer loop merge.

After `pushback`, states s and t will be merged. If $x = \mathtt{Suff}^{k-1}(q)$, then both s and t must have xa as a suffix. Since $|xa| = k$, it follows that $\mathtt{Suff}^{k-1}(s) = \mathtt{Suff}^{k-1}(t)$. It also follows that additional states merged to remove non-determinism resulting from the merge of s and t will have the same suffix of length $(k-1)$. To show that all states with the same $(k-1)$-length suffix will be merged, it is shown that the ISLFLA will *never* encounter the situation in Figure 9, provided the data set includes a *seed* defined as follows.

Definition 8 (ISLFLA seed)**.** *Given a k-ISL function f and $\mathcal{T}_f^{\mathtt{ISL}}$, let $Q' = \{q \in Q \mid \delta_1(q, \#) \neq \lambda\}$. A* seed *for f is $S = S' \cup S''$, where*

- $S' = \{(w, w') \mid [w \in \Sigma^{\leq k} \wedge f(w) = w']\}$
- $S'' = \{(wa, w'') \mid a \in \Sigma, [w \in \Sigma^k \wedge \mathtt{Suff}^{k-1}(w) \in Q' \wedge f(wa) = w'']\}$.

Lemma 2 (Convergence)**.** *A seed for an ISL function f is a characteristic sample for ISLFLA.*

Proof. We show that for any ISL function f and a dataset D that contains a seed S the output of the ISLFLA is $\mathcal{T}_f^{\mathtt{ISL}}$.

Let $PTT(D) = (Q_{PTT}, q_{0_{PTT}}, \Sigma, \Gamma, \delta_{PTT}, \sigma_{PTT})$ be the input to the ISLFLA and $\mathcal{T} = (Q_\mathcal{T}, q_{0_\mathcal{T}}, \Sigma, \Gamma, \delta_\mathcal{T}, \sigma_\mathcal{T})$ be its output. First we show that $Q_\mathcal{T} = \Sigma^{\leq k-1}$. By Definitions 4 and 8, $\Sigma^{\leq k-1} \subseteq Q_{PTT}$. Since the ISLFLA only merges states with the same $(k-1)$-length suffix, $\Sigma^{\leq k-1} \subseteq Q_\mathcal{T}$. Since it does not exit until all states q have been merged with $\mathtt{Suff}^{k-1}(q)$, $Q_\mathcal{T} = \Sigma^{\leq k-1}$.

Next we show that given S, the algorithm will never need to merge two states q_1 and q_2 such that $\delta_1(q_1, \#) \neq \delta_1(q_2, \#)$. Let $\delta_1(q_1, \#) = z$, and $\delta_1(q_2, \#) = x$ with $z \neq x$ and $q_1 = \mathtt{Suff}^{k-1}(q_2)$. By Definition 2, $\mathtt{tails}_f(q_1) = \mathtt{tails}_f(q_2)$, so if $z \neq x$ it must be the case that q_2 does not have transitions for all $a \in \Sigma$. This is because the only way for the output strings of the outgoing transitions of q_2 to differ from those of q_1 is if fewer transitions were present on q_2 when the PTT was made onward. (By definition of S we know q_1 has transitions for all $a \in \Sigma$.) But since $\mathtt{tails}_f(q_1) = \mathtt{tails}_f(q_2)$, we also know that $z = ux$ for some $u \in \Gamma^*$.

By Definition 8, all states up to length k have transitions for all $a \in \Sigma$; therefore, $|q_2| \geq k+1$. This means $\exists q' \in \Sigma^k$ between q_1 and q_2, which will be merged with some other state before q_2 will. This merge will cause non-determinism, which in turn will trigger `pushback` and cause u to move further down the branch toward q_2. By extension there will be $|q_2| - k$ states between q_1 and q_2, each of which will be merged, triggering `pushback` of u, so that by the time q_1 and q_2 are merged, $\delta_1(q_2, \#) = ux = z = \delta_1(q_2, \#)$. Thus, all non-determinism can be removed, and so $\mathcal{T}$ is subsequential.

It remains to show that $\forall q \in Q_\mathcal{T}, a \in \Sigma$, $\delta_2(q, a) = \mathtt{Suff}^{k-1}(qa)$. Since state merging preserves transitions, this follows from the construction of $PTT(D)$. By Theorem 3, $\mathcal{T} = \mathcal{T}_f^{\mathtt{ISL}}$. □

Next we establish complexity bounds on the runtime of the ISLFLA and the size of the characteristic sample for ISL functions. We observe that both of these bounds improve the bounds of OSTIA. While not surprising, since ISL functions are less general than subsequential functions, the result is important since it is an example of greater *a priori* knowledge enabling learning with less time and data.

Let m be the length of the longest output string in the sample and let n denote the number of states of the PTT; n is at most the sum of the lengths of the input strings of the pairs in the sample.

Lemma 3 (Polynomial time)**.** *The time complexity of the ISLFLA is in $\mathcal{O}(n \cdot m \cdot k \cdot |\Sigma|)$.*

Proof. First, making the PTT onward can be done in $\mathcal{O}(m \cdot n)$: it consists of a depth-first parsing of the PTT from its root, with a computation at each state of the `lcp` of the outgoing transition outputs after the recursive computation of the function (see de la Higuera (2010), Chap. 18, for details). As the computation of the `lcp` takes at most m steps, and as it has to be done for each state, the complexity of this step is effectively in $\mathcal{O}(m \cdot n)$.

For the two loops, we need to find a bound on the number of merges that can occur. States q such that $|q| < k$ do not yield any merges in the outer loop. All other states q' are merged with $\mathtt{Suff}^{k-1}(q)$, in the outer loop or in the inner one. The number of merges is thus bounded by n. Computing the suffix of length $(k-1)$ of any word can be done in $\mathcal{O}(k)$ with a correct implementation of strings of characters. The test of the inner loop can be done in constant time and so can the `merge` and `pushback` procedures. After each merge, the test of the inner loop needs to be done at most $|\Sigma|$ times. As computing the `lcp` has a complexity in $\mathcal{O}(m)$, the overall complexity of the two loops is in $\mathcal{O}(n \cdot m \cdot k \cdot |\Sigma|)$.

The overall complexity of the algorithm is thus $\mathcal{O}(m \cdot n + n \cdot m \cdot k \cdot |\Sigma|) = \mathcal{O}(n \cdot m \cdot k \cdot |\Sigma|)$. □

Let $\mathcal{T}_f^{\mathtt{ISL}} = (Q, q_0, \Sigma, \Gamma, \delta, \sigma)$ be a target transducer. We define $m = \mathtt{max}\{|u| \mid (q, a, u, q') \in \delta\}$, $n = |Q|$, and $p = \mathtt{max}\ \{|v| \mid (q, \#, v) \in \sigma\}$.

Lemma 4 (Polynomial data)**.** *The size of the characteristic sample is in $\mathcal{O}(n \cdot |\Sigma| \cdot k \cdot m + n^2 \cdot m + n \cdot |\Sigma| \cdot p)$.*

Proof. The first item of the seed, S', covers all and only the states of the target: the left projection of these pairs is thus linear in n and every right projection is at most $n \cdot m + p$. Thus the size of S' is at most $n \cdot (n + n \cdot m + p) = \mathcal{O}(n^2 \cdot m + n \cdot p)$.

Concerning the second part, S'', its cardinality is at most $n \cdot |\Sigma|$ (in the rare case where $Q' = Q$).

Each element of the left projection of S'' is of length $(k+1)$ and each element of its right projection is at most of length $(k+1)\cdot m+p$. The size of S'' is thus in $\mathcal{O}(n\cdot|\Sigma|\cdot(k\cdot m+p))$.

Therefore, the size of the characteristic sample is in $\mathcal{O}(n\cdot|\Sigma|\cdot k\cdot m+n^2\cdot m+n\cdot|\Sigma|\cdot p)$, which is clearly polynomial in the size of the target transducer. □

Theorem 8. *The ISLFLA identifies the class of ISL functions in polynomial time and data.*

Proof. Immediate from Lemmas 2, 3, and 4. □

5.3 Demonstrations

We tested the ISLFLA with the three examples in §4, as well as the English flapping process (t → ɾ / V́ __ V). For each case, a data set was constructed according to Definition 8 using the alphabets presented in §4. The alphabet for English flapping was {V́, V, t, ?}. The value of k is 2 for final devoicing, ə-epenthesis, and fricative deletion and 3 for English flapping. A few additional data points of length 5 or 6 were also added to make the data set a superset of the seed. In all four cases, the learner returned the correct $\mathcal{T}_f^{\mathtt{ISL}}$.

The decision to use artificial corpora in these demonstrations was motivated by the fact that the sample in Definition 8 will not be present in a natural language corpus. That sample includes all possible sequences of symbols from the alphabet of a given length, whereas a natural language corpus will reflect the language-particular restrictions against certain segment sequences (i.e., phonotactics).

As discussed in the introduction, Gildea and Jurafsky (1996) address this issue with natural language data by equipping OSTIA with a community bias, whereby segments belonging to a natural class (i.e., stops, fricatives, sonorants) are expected to behave similarly, and a faithfulness bias, whereby segments are assumed to be realized similarly on the surface. In our demonstrations we put aside the issue of the behavior of segments in a natural class by using abbreviated alphabets (e.g., T for all voiceless stops). But if in fact knowledge of natural classes precedes the learning of phonological processes, the use of such an alphabet is appropriate.

In future developments of the ISLFLA we likewise aim to accommodate natural language data, but in a way that maintains the theoretical result of identification in the limit. The restrictions on segment sequences represented in natural language data amount to 'missing' transitions in the initial prefix tree transducer that is built from that data. In other words, the transducer represents a partial, not a total function. Thus it seems the approach of Oncina and Varò (1996) and Castellanos et al. (1998) could be very instructive, as their use of domain information enabled OSTIA to learn partial functions. In our case, the fact that the domain of an ISL function is an SL language could provide a means of 'filling in' the missing transitions. The details of such an approach are, however, being left for future work.

6 Conclusion

This paper has defined Input and Output Strictly Local functions, which synthesize the properties of subsequential transduction and Strictly Local formal languages. It has provided language-theoretic characterizations of these functions and argued that they can model many phonological and morphological processes. Lastly, an automata-theoretic characterization of ISL functions was presented, along with a learning algorithm that efficiently learn this class in the limit from positive data.

Current work includes developing a comparable automata characterization and learning algorithm for OSL functions, as well as defining additional functional classes to model those phonological processes that cannot be modeled with ISL or OSL functions. The SL languages are just one region of a subregular hierarchy of formal languages (McNaughton and Papert, 1971; Rogers and Pullum, 2011; Rogers et al., 2013). The ISL and OSL functions defined here are the first step in developing a corresponding hierarchy of subregular functions. Of immediate interest to phonology are functional counterparts for the Tier-Based Strictly Local and Strictly Piecewise language classes, which have been shown to model long-distance phonotactics (Heinz, 2010; Heinz et al., 2011). Such functions might be useful for modeling the long-distance processes that repair violations of these phonotactic constraints.

Acknowledgements

We thank Adam Jardine, Jim Rogers, and the three anonymous reviewers for helpful comments. This research was supported by NSF CPS#1035577.

References

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

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

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

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

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

Jane Chandlee and Adam Jardine. 2014. Learning phonological mappings by learning Strictly Local functions. In John Kingston, Claire Moore-Cantwell, Joe Pater, and Robert Staubs, editors, *Proceedings of the 2013 Meeting on Phonology*. LSA.

Jane Chandlee and Cesar Koirala. 2014. Learning local phonological rules. In *Proceedings of the 37th Penn Linguistics Conference*.

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

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

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

Colin de la Higuera. 1997. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27(2):125–138.

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

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

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

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

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

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

Jeffrey Heinz, 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*, pages 58–64. Association for Computational Linguistics.

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

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

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

Mans Hulden, Iñaki Alegria, Izaskun Etxeberria, and Montse Maritxalar. 2011. Learning word-level dialectal variation as phonological replacement rules using a limited parallel corpus. In *Proceedings of the First Workshop on Algorithms and Resources for Modelling of Dialects and Language Varieties*, DIALECTS '11, pages 39–48. Association for Computational Linguistics.

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

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

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

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

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

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

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

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

José Oncina and Pedro García. 1991. Inductive learning of subsequential functions. Technical Report DSIC II-34, University Politécnia de Valencia.

José Oncina 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*, pages 313–325.

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

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

Alan Prince and Paul Smolensky. 1993. Optimality Theory: Constraint interaction in generative grammar. Technical Report 2, Rutgers University Center for Cognitive Science.

Jason Riggle. 2003. Non-local reduplication. In *Proceedings of the 34th Annual Meeting of the North Eastern Linguistic Society*.

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

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

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

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

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