Output Strictly Local Functions
Jane Chandlee, Rémi Eyraud, Jeffrey Heinz

To cite this version:
Jane Chandlee, Rémi Eyraud, Jeffrey Heinz. Output Strictly Local Functions. 14th Meeting on the Mathematics of Language, Jul 2015, Chicago, United States. pp.112–125, 10.3115/v1/W15-2310 . hal-01193051

HAL Id: hal-01193051
https://hal.science/hal-01193051
Submitted on 7 Sep 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Abstract

This paper characterizes a subclass of subsequential string-to-string functions called Output Strictly Local (OSL) and presents a learning algorithm which provably learns any OSL function in polynomial time and data. This algorithm is more efficient than other existing ones capable of learning this class. The OSL class is motivated by the study of the nature of string-to-string transformations, a cornerstone of modern phonological grammars.

1 Introduction

Motivated by questions in phonology, this paper studies the Output Strictly Local (OSL) functions originally defined by Chandlee (2014) and Chandlee et al. (2014). The OSL class is one way Strictly Local (SL) stringsets can be generalized to string-to-string maps. Their definition is a functional version of a defining characteristic of SL stringsets called Suffix Substitution Closure (Rogers and Pullum, 2011). Similar to SL stringsets, the OSL functions contain nested subclasses parameterized by a value $k$, which is the length of the suffix of output strings that matters for computing the function.

As Chandlee (2014) argues, almost all local phonological processes can be modeled with Input Strictly Local (ISL) functions. Yet there is one notable class of exceptions: so-called spreading processes, in which a feature like nasality iteratively assimilates over a contiguous span of segments. As we show, the OSL functions are needed to describe this sort of phenomenon.

Here we provide a slight, but important, revision to the original definition of OSL functions in Chandlee (2014) and Chandlee et al. (2014), which allows two important theoretical contributions while preserving the previous results. The first is a finite-state transducer (FST) characterization of OSL functions, which leads to the second result, the OSLFIA (OSL Function Inference Algorithm) and a proof that it efficiently identifies the $k$-OSL functions from positive examples. We compare this algorithm to OSTIA (Onward Subsequential Transducer Inference Algorithm, Oncina et al. (1993)) which identifies total subsequential functions in cubic time, its modifications OSTIA-D and OSTIA-R, which can learn particular subclasses of subsequential functions using domain and range information, respectively, in at least cubic time (Oncina and Varò, 1996; Castellanos et al., 1998), and SOSFIA (Structured Onward Subsequential Inference Algorithm, Jardine et al. (2014)), which can learn particular subclasses of subsequential functions in linear time and data. We show these algorithms either cannot learn the OSL functions or do so less efficiently than the OSLFIA. These contributions were missing from the initial research on OSL functions (except for a preliminary FST characterization in Chandlee (2014)). Finally, we explain how a unified theory of local phonology will have to draw insights from both the ISL and OSL classes and offer an idea of how this might work. Thus, this paper is a crucial and necessary intermediate step towards an empirically adequate but restrictive characterization of phonological locality.

The remainder of the paper is organized as follows. Motivation and related work are given in sec-
tion 2, including an example of the spreading processes that cannot be modeled with ISL functions. Notations and background concepts are presented in section 3. In section 4 we define OSL functions, and the theoretical characterization and learning results are given in sections 5 and 6. In section 7, we explain how OSL functions model spreading processes. In section 8 we elaborate on a few important areas for future work, and in section 9 we conclude.

2 Background and related work

A foundational principle of modern generative phonology is that systematic variation in morpheme pronunciation is best explained with a single underlying representation of the morpheme that is transformed into various surface representations based on context (Kenstowicz and Kisseberth, 1979; Odden, 2014). Thus, much of generative phonology is concerned with the nature of these transformations.

One way to better understand the nature of linguistic phenomena is to develop strong computational characterizations of them. Discussing SPE-style phonological rewrite rules (Chomsky and Halle, 1968), Johnson (1972, p. 43) expresses the reasoning behind this approach:

It is a well-established principle that any mapping whatever that can be computed by a finitely storable, well-defined procedure can be effected by a rewriting system (in particular, by a Turing machine, which is a special kind of rewriting system). Hence any theory which allows phonological rules to simulate arbitrary rewriting systems is seriously defective, for it asserts next to nothing about the sorts of mappings these rules can perform.

This leads to the important question of what kinds of transformations ought a theory of phonology allow?

Earlier work suggests that phonological theories ought to exclude nonregular relations (Johnson, 1972; Kaplan and Kay, 1994; Frank and Satta, 1998; Graf, 2010). More recently, it has been hypothesized that phonological theory ought to only allow certain subclasses of the regular relations (Gainor et al., 2012; Chandlee et al., 2012; Chandlee and Heinz, 2012; Payne, 2013; Luo, 2014; Heinz and Lai, 2013). This research places particular emphasis on subsequential functions, which can informally be characterized as functions definable with a weighted, deterministic finite-state acceptor where the weights are strings and multiplication is concatenation. The aforementioned work suggests that this hypothesis enjoys strong support in segmental phonology, with interesting and important exceptions in the domain of tone (Jardine, 2014).

Recent research has also showed an increased awareness and understanding of irregular classes of stringsets (formal languages) and their importance for theories of phonotactics (Heinz, 2007; Heinz, 2009; Heinz, 2010; Rogers et al., 2010; Rogers and Pullum, 2011; Rogers et al., 2013). While many of these classes and their properties were studied much earlier (McNaughton and Papert, 1971; Thomas, 1997), little to no attention has been paid to similar classes properly contained within the subsequential functions. Thus, at least within the domain of segmental phonology, there is an important question of whether stronger computational characterizations of phonological transformations are possible, as seems to be the case for phonotactics.

As mentioned above, Chandlee (2014) shows that many phonological processes belong to a subclass of subsequential functions, the Input Strictly Local (ISL) functions. Informally, a function is k-ISL if the output of every input string \(a_0a_1\cdots a_n\) is \(u_0u_1\cdots u_n\) where \(u_i\) is a string which only depends on \(a_i\) and the \(k-1\) input symbols before \(a_i\) (so \(a_{i-k+1}a_{i-k+2}\cdots a_{i-1}\)). (A formal definition is given in section 4). ISL functions can model a range of processes including local substitution, epenthesis, deletion, and metathesis. For more details on the exact range of ISL processes, see Chandlee (2014) and Chandlee and Heinz (to appear).

Processes that aren’t ISL include long-distance processes as well as local iterative spreading processes. As an example of the latter, consider nasal spreading in Johore Malay (Omn, 1980). As shown in (1), contiguous sequences of vowels and glides are nasalized following a nasal:

\[(1) \quad /p@n@j@aw@a@san/ \rightarrow [p@n@j@\ddot{a}@\ddot{w}@\ddot{a}@\ddot{a}@san], ‘supervision’\]

This process is not ISL, because the initial trigger of the spreading (the nasal) can be arbitrarily far from a target (as suggested by the nasalization of the glide
and the second [ä]) when the distance is measured on the input side. However, on the output side, the triggering context is local; the second [ä] is nasalized because the preceding glide on the output side is nasalized. Every segment between the trigger and target is affected; nasalization applies to a contiguous, but arbitrarily long, substring. It is this type of process that we will show requires the notion of Output Strict Locality.

Processes in which a potentially unbounded number of unaffected segments can intervene between the trigger and target - such as long-distance consonant agreement (Hansson, 2010; Rose and Walker, 2004), vowel harmony (Nevins, 2010; Walker, 2011), and consonant dissimilation (Suzuki, 1998; Bennett, 2013) - are neither ISL nor OSL. More will be said about such long-distance processes in §7.

3 Preliminaries

The set of all possible finite strings of symbols from a finite alphabet Σ and the set of strings of length ≤ n are Σ* and Σn, respectively. The cardinality of a set S is denoted card(S). The unique empty string is represented with λ. The length of a string w is |w|, so |λ| = 0. If w1 and w2 are strings then w1w2 is their concatenation. The prefixes of w, Pref(w), is \{p ∈ Σ* | (∃s ∈ Σ*)[w = ps]\}, and the suffixes of w, Suff(w), is \{s ∈ Σ* | (∃p ∈ Σ*)[w = ps]\}. For all w ∈ Σ* and n ∈ ℤ, Suffn(w) is the single suffix of w of length n if |w| ≥ n; otherwise Suffn(w) = w. The following reduction will prove useful later.

Remark 1. For all w, v ∈ Σ*, n ∈ ℤ, Suffn(Suffn(w)v) = Suffn(wv).

If w = vw is a string then let v = u−1 · w and u = w · v−1. Trivially, λ−1 · w = w = λ1 · w, uu−1 · w = w, and w · v-v = w.

We assume a fixed but arbitrary total order < on the letters of Σ. As usual, we extend < to Σ* by defining the hierarchical order (Oncina et al., 1993), denoted ≺, as follows: ∀w1, w2 ∈ Σ*, w1 < w2 iff

\[
\begin{cases}
|w_1| < |w_2| \text{ or } \\
|w_1| = |w_2| \text{ and } \exists u, v_1, v_2 \in Σ, \exists a_1, a_2 \in Σ \\
\text{s.t. } w_1 = ua_1v_1, w_2 = ua_2v_2 \text{ and } a_1 < a_2.
\end{cases}
\]

≺ is a total strict order over Σ*, and if Σ = \{a, b\} and a < b, then λ≺a≺b≺aa≺ab≺ba≺bb≺aaa≺... .

The longest common prefix of a set of strings S, lcp(S), is p ∈ ∩w∈S Pref(w) such that ∀p′ ∈ ∩w∈S Pref(w), |p′| < |p|. Let f : A → B be a function f with domain A and co-domain B. When A and B are stringsets, the input and output languages of f are pre_image(f) = \{x | (∃y)[x → f y]\} and image(f) = \{y | (∃x)[x → f y]\}, respectively.

Jardine et al. (2014) introduce delimited subsequential FSTs (DSFSTs). The class of functions describable with DSFSTs is exactly the class representable by traditional subsequential FSTs (Oncina and García, 1991; Oncina et al., 1993; Mohri, 1997), but DSFSTs make explicit use of symbols marking both the beginnings and ends of input strings.

Definition 1. A delimited subsequential FST (DSFST) is a 6-tuple \(\langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle\) where Q is a finite set of states, q0 ∈ Q is the initial state, qf ∈ Q is the final state, Σ and Δ are finite alphabets of symbols, \(\delta \subseteq Q \times (\Sigma \cup \{\ast, \ast\}) \times \Delta^* \times Q\) is the transition function (where \(\ast \notin \Sigma\) indicates the 'start of the input' and \(\ast \notin \Sigma\) indicates the 'end of the input'), and the following hold:

1. if (q, σ, u, q′) ∈ \(\delta\) then q ≠ qf and q′ ≠ q0.
2. if (q, σ, u, qf) ∈ \(\delta\) then σ = \(\ast\) and q ≠ q0.
3. if (q0, σ, u, q0) ∈ \(\delta\) then σ = \(\ast\) and if (q, \(\ast\), u, q′) ∈ \(\delta\) then q = q0.
4. if (q, σ, w, r), (q, σ, v, s) ∈ \(\delta\) then (r = s) ∧ (w = v).

In words, in DSFST, initial states have no incoming transitions (1) and exactly one outgoing transition for input \(\ast\) (3) which leads to a nonfinal state (2), and final states have no outgoing transitions (1) and every incoming transition comes from a noninitial state and has input \(\ast\) (2). DSFSTs are also deterministic on the input (4). In addition, the transition function may be partial. We extend the transition function to \(\delta^*\) recursively in the usual way: \(\delta^*\) is the smallest set containing \(\delta\) and which is closed under the following condition: if (q, w, u, q′) ∈ \(\delta^*\) and (q′, σ, v, q″) ∈ \(\delta\) then (q, w, σ, uv, q″) ∈ \(\delta^*\). Note no elements of the form (q, \(\lambda\), \(\lambda\), q′) are elements of \(\delta^*\).

The size of a DSFST T = \(\langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle\) is \(|T| = \text{card}(Q) + \text{card}(\Delta) + \sum_{(q,a,u,q') \in \delta} |u|\).

A DSFST T defines the following relation:

\[
R(T) = \{ (x, y) \in \Sigma^* \times \Delta^* \mid \exists (q_0, \ast x \ast, y, q_f) \in \delta^* \}
\]
Since DSFSTs are deterministic, the relations they recognize are (possibly partial) functions. Sequential functions are defined as those representable with DSFSTs for which for all \((q, x, u, q_f) \in \delta\), \(u = \lambda\).

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

\[
\text{tails}_f(x) = \{(y, v) \mid f(xy) = uv \land u = 1\text{cp}(f(x\Sigma^*))\}.
\]

If \(x_1, x_2 \in \Sigma^*\) have the same set of tails with respect to \(f\), they are \(f\)-tail-equivalent with respect to \(f\), written \(x_1 \sim_f x_2\). Clearly, \(\sim_f\) is an equivalence relation which partitions \(\Sigma^*\).

**Theorem 1** (Oncina and Garcia, 1991). A function \(f\) is subsequential iff \(\sim_f\) partitions \(\Sigma^*\) 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 \(\text{tails}_L(w) = \{u \mid wu \in L\}\). These tails can be used to partition \(\Sigma^*\) 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 1 shows how to construct the (unique up to isomorphism) smallest subsequential transducer for a subsequential function \(f\). With little modification to their proof, the smallest DSFST for \(f\) can also be constructed. We refer to this DSFST as the **canonical DSFST for \(f\)** and denote it \(\mathcal{T}_C(f)\). (If \(f\) is understood from context, we may write \(\mathcal{T}_C\).) States of \(\mathcal{T}_C(f)\) which are neither initial nor final are in one-to-one correspondence with \(\text{tails}_f(x)\) for all \(x \in \Sigma^*\) (Oncina and Garcia, 1991). To construct \(\mathcal{T}_C(f)\) we first let, for all \(x \in \Sigma^*\) and \(a \in \Sigma\), the **contribution of** \(a\) w.r.t. \(x\) be \(\text{cont}_f(a, x) = 1\text{cp}(f(x\Sigma^*)^{-1} \cdot 1\text{cp}(f(xa\Sigma^*))\).

- For all \(x \in \Sigma^*\), \(a \in \Sigma\), if \(\exists y \in \Sigma^*\) with \(xya \in \text{pre_image}(f)\) then \(\text{tails}_f(x), a, \text{cont}_f(a, x), \text{tails}_f(xa) \in \delta\).
- Nothing else is in \(\delta\).

Observe that unlike the traditional construction, the initial state \(q_0\) is not \(\text{tails}_f(\lambda)\). The single outgoing transition from \(q_0\), however, goes to this state with the input \(\times\). Canonical DSFSTs have an important property called **onwardness**.

**Definition 2** (onwardness). A DSFST \(\mathcal{T}\) is **onward** if for every \(w \in \Sigma^*\), \(u \in \Delta^*\), \((q_0, \times w, u, q) \in \delta^* \iff \exists u = 1\text{cp}(\{f(w\Sigma^*)\})\).

Informally, this means that the writing of output is never delayed. For all \(q \in Q\) let the **outputs of the edges out of** \(q\) be \(\text{outputs}(q) = \{u \mid (\exists \sigma \in \Sigma \cup \{\times, \times\})(\exists q' \in Q)[(q, \sigma, a, u, q') \in \delta]\} \) (Oncina and Garcia, 1991). To construct \(\mathcal{T}\), \(\mathcal{T}\) is **onward** if \(\text{outputs}(q_0) = \lambda\) and \(1\text{cp}(\text{outputs}(q_0)) = 1\text{cp}(f(\Sigma^*))\).

**Lemma 1.** If DSFST \(\mathcal{T}\) recognizes \(f\) and is onward then \(\forall q \neq q_0\) \(1\text{cp}(\text{outputs}(q)) = \lambda\) and \(1\text{cp}(\text{outputs}(q_0)) = 1\text{cp}(f(\Sigma^*))\).

**Proof.** By construction of a DSFST, only one transition leaves \(q_0\): \((q_0, \times, u, q.f)\) for every \(u \in \Sigma^*, q \in \Delta^*\), \((q_0, \times \lambda, u, q) \in \delta^*\) and as the transducer is onward we have \(1\text{cp}(\text{outputs}(q_0)) = 1\text{cp}(u) = u = 1\text{cp}(f(\Sigma^*)) = 1\text{cp}(f(\Sigma^*))\).

Now take \(q \neq q_0\) and \(w \in \Sigma^*\) such that \((q_0, \times w, u, q) \in \delta^*\). Suppose \(1\text{cp}(\text{outputs}(q)) = v \neq \lambda\). Then \(v\) is a prefix of \(1\text{cp}(\{f(w\Sigma^*)\})\) which implies \(wv\) is a prefix of \(1\text{cp}(f(w\Sigma^*))\).

Readers are referred to Oncina and Garcia (1991), Oncina et al. (1993), and Mohri (1997) for more on subsequential transducers, and Eisner (2003) for generalizations regarding onwardness.

## 4 Output Strictly Local functions

Here we define Output Strictly Local (OSL) functions, which were originally introduced by Chandlee (2014) and Chandlee et al. (2014) along with the Input Strictly Local (ISL) functions. Both classes generalize SL stringsets to functions based on a defining property of SL languages, the Suffix Substitution Closure (Rogers and Pullum, 2011).

**Theorem 2** (Suffix Substitution Closure). \(L\) is **Strictly Local** iff for all strings \(u_1, v_1, u_2, v_2\), there

\[\exists y \in \Sigma^* \text{ s.t. } f(u_1v_1u_2v_2) = f(u_1yv_1u_2v_2).\]
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 \( u_1xv_2 \in L \).

An important corollary of this theorem follows.

**Corollary 1** (Suffix-defined Residuals). \( L \) is Strictly Local iff \( \forall w_1, w_2 \in \Sigma^* \), there exists \( k \in \mathbb{N} \) such that if \( \text{Suff}^{k-1}(w_1) = \text{Suff}^{k-1}(w_2) \) then the residuals (the tails) of \( w_1, w_2 \) with respect to \( L \) are the same; formally, \( \{ v \mid w_1v \in L \} = \{ v \mid w_2v \in L \} \).

Input and Output Strictly Local functions were defined by Chandlee (2014) and Chandlee et al. (2014) in the manner suggested by the corollary.

**Definition 3** (Input Strictly Local Functions). A function \( f : \Sigma^* \to \Delta^* \) is ISL if there is a \( k \) such that for all \( u_1, u_2 \in \Sigma^* \), \( \text{Suff}^{k-1}(u_1) = \text{Suff}^{k-1}(u_2) \) then \( \text{tails}_f(u_1) = \text{tails}_f(u_2) \).

**Definition 4** (Output Strictly Local Functions (original)). A function \( f : \Sigma^* \to \Delta^* \) is OSL if there is a \( k \) such that for all \( u_1, u_2 \in \Sigma^* \), if \( \text{Suff}^{k-1}(f(u_1)) = \text{Suff}^{k-1}(f(u_2)) \) then \( \text{tails}_f(u_1) = \text{tails}_f(u_2) \).

While Definition 3 lead to an automata-theoretic characterization and learning results for ISL (Chandlee et al., 2014), such results do not appear possible with the original definition of OSL. The problem is with subsequential functions that are not sequential. The value returned by the function includes the writing that occurs when the input string has been fully read (i.e., the output of transitions going to the final state in a corresponding DSFST). This creates a problem because it does not allow for separation of what happens during the computation from what happens at its end.

Figure 1 illustrates the distinction Definition 4 is unable to make.\(^2\) Function \( f \) is sequential, but \( g \) is not. Otherwise, they are identical. While \( f(bab) = bba \), \( g(bab) = bbaa \). With the original OSL definition, there is no way to refer to the output for input \( bab \) before the final output string has been appended.

To deal with this problem we first define the prefix function associated to a subsequential function.

**Definition 5** (Prefix function). Let \( f : \Sigma^* \to \Delta^* \) be a subsequential function. We define the prefix function \( f^p : \Sigma^* \to \Delta^* \) associated to \( f \) such that \( f^p(w) = \text{lcp} \{ f(w\Sigma^*) \} \).

\(^2\)Here and in Figure 3, state \( q_I \) is not shown. Non-initial states are labeled \( q : u \) with \( q \) being the state’s name and \( (q, \varkappa, u, q_I) \in \delta \).

Input and Output Strictly Local functions were defined by Chandlee (2014) and Chandlee et al. (2014) in the manner suggested by the corollary.

**Definition 3** (Input Strictly Local Functions). A function \( f : \Sigma^* \to \Delta^* \) is ISL if there is a \( k \) such that for all \( u_1, u_2 \in \Sigma^* \), \( \text{Suff}^{k-1}(u_1) = \text{Suff}^{k-1}(u_2) \) then \( \text{tails}_f(u_1) = \text{tails}_f(u_2) \).

**Definition 4** (Output Strictly Local Functions (original)). A function \( f : \Sigma^* \to \Delta^* \) is OSL if there is a \( k \) such that for all \( u_1, u_2 \in \Sigma^* \), if \( \text{Suff}^{k-1}(f(u_1)) = \text{Suff}^{k-1}(f(u_2)) \) then \( \text{tails}_f(u_1) = \text{tails}_f(u_2) \).

While Definition 3 lead to an automata-theoretic characterization and learning results for ISL (Chandlee et al., 2014), such results do not appear possible with the original definition of OSL. The problem is with subsequential functions that are not sequential. The value returned by the function includes the writing that occurs when the input string has been fully read (i.e., the output of transitions going to the final state in a corresponding DSFST). This creates a problem because it does not allow for separation of what happens during the computation from what happens at its end.

Figure 1 illustrates the distinction Definition 4 is unable to make.\(^2\) Function \( f \) is sequential, but \( g \) is not. Otherwise, they are identical. While \( f(bab) = bba \), \( g(bab) = bbaa \). With the original OSL definition, there is no way to refer to the output for input \( bab \) before the final output string has been appended.

To deal with this problem we first define the prefix function associated to a subsequential function.

**Definition 5** (Prefix function). Let \( f : \Sigma^* \to \Delta^* \) be a subsequential function. We define the prefix function \( f^p : \Sigma^* \to \Delta^* \) associated to \( f \) such that \( f^p(w) = \text{lcp} \{ f(w\Sigma^*) \} \).

\(^2\)Here and in Figure 3, state \( q_I \) is not shown. Non-initial states are labeled \( q : u \) with \( q \) being the state’s name and \( (q, \varkappa, u, q_I) \in \delta \).

**Remark 2.** If \( T \) is an onward DSFST for \( f \), then \( \forall w \in \Sigma^* \), \( f^p(w) = u \iff \exists q, (\varkappa, w, u, q) \in \delta^* \).

**Remark 3.** If \( f \) is sequential then \( f = f^p \).

We can now revise the definition of OSL functions.

**Definition 6** (Output Strictly Local Function (revised)). We say that a subsequential function \( f \) is \( k \)-OSL if for all \( w_1, w_2 \in \Sigma^* \), \( \text{Suff}^{k-1}(f^p(w_1)) = \text{Suff}^{k-1}(f^p(w_2)) \Rightarrow \text{tails}_f(w_1) = \text{tails}_f(w_2) \).

Chandlee et al. (2014) provide several theorems which relate ISL functions, OSL functions (defined as in Definition 4), and SL stringsets. Here we explain why those results still hold with Definition 6.

The proofs for those results depend on the six functions \( (f_i, 1 \leq i \leq 6) \) reproduced here in Figure 2. The transducers shown there are not DSFSTs but traditional subsequential transducers; readers are referred to Chandlee et al. (2014) for formal definitions. With the exception of \( f_5 \), these functions are clearly sequential since each state outputs \( \lambda \) on \( \varkappa \) (shown as \( \# \) in Figure 2). The transducer for \( f_5 \) is not onward, but an onward, sequential version of this transducer recognizing exactly the same function is obtained by suffixing \( a \) (which is the \( \text{lcp} \) of the outputs of state 1) onto the output of state 1’s incoming transition. Thus, \( f_5 \) is also sequential. By Remark 3 then, Theorems 4, 5, 6, and 7 of that paper still hold under Definition 6.

**5 Automata characterization**

First we show, for any non-initial state of any canonical transducer recognizing an OSL function, that if reading a letter \( a \) implies writing \( \lambda \), then this corresponds to a self-loop. So writing the empty string never causes a change of state (except from \( q_0 \)).

**Lemma 2.** For any OSL function \( f \) whose canonical DSFST is \( T_C \), if \( \exists q \neq q_0, a \in \Sigma, \) and \( q' \in Q \) such that \( f(q, \varkappa, u, q') \neq \lambda \), then \( q' \in Q \) and \( q' \notin \delta(q, a) \).
that \((q, a, \lambda, q') \in \delta_C\) then \(q' = q\).

**Proof.** Consider \(w\) and \(u\) such that \((q_0, \times w, u, q) \in \delta_C^\ast\) and suppose \((q, a, \lambda, q') \in \delta_C\). Then \(f^P(w) = f^P(wa)\) which implies \(\text{Suff}^k(f^P(w)) = \text{Suff}^k(f^P(wa))\). As \(f\) is \(k\)-OSL, \(\text{tails}_f(w) = \text{tails}_f(wa)\). As \(\mathcal{T}_C\) is canonical the non-initial and non-final states correspond to unique tail-equivalence classes, and two distinct states correspond to two different classes. Therefore \(q' = q\). \(\square\)

Next we define \(k\)-OSL transducers.

**Definition 7 (\(k\)-OSL transducer).** An onward DFST \(\mathcal{T} = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle\) is \(k\)-OSL if

1. \(Q = S \cup \{q_0, q_f\}\) with \(S \subseteq \Delta^{\leq k-1}\)
2. \((\forall u \in \Delta^\ast)[(q_0, \times, u, q) \in \delta \implies q' = \text{Suff}^k(u)]\)
3. \((\forall q \in Q \setminus \{q_0\}, \forall a \in \Sigma, \forall u \in \Delta^\ast)\) \([[(q, a, u, q') \in \delta \implies q' = \text{Suff}^k(qu)]\).

Next we show that \(k\)-OSL functions and functions represented by \(k\)-OSL DFSTs exactly correspond.

**Lemma 3** (extended transition function). Let \(\mathcal{T} = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle\) be a \(k\)-OSL DFST. We have \((q_0, \times w, u, q) \in \delta^\ast \implies q = \text{Suff}^k(u)\)

**Proof.** By recursion on the size of \(w \in \Sigma^\ast\). The initial case is valid for \(|w| = 0\) since \((q_0, \times, u, q) \in \delta^\ast\) then \((q_0, \times, u, q) \in \delta\). By Definition 7, \(q = \text{Suff}^{k-1}(u)\). Suppose now that the lemma holds for inputs of size \(n \neq 0\). Let \(w\) be of size \(n\) such that \((q_0, \times w, u, q) \in \delta^\ast\) and suppose \((q, a, v, q') \in \delta\) (i.e., \((q_0, \times wa, uv, q') \in \delta^\ast\)). By recursion, we know that \(q = \text{Suff}^{k-1}(u)\). By Definition 7, \(q' = \text{Suff}^{k-1}(qv) = \text{Suff}^{k-1}(\text{Suff}^{k-1}(u)v) = \text{Suff}^{k-1}(uv)\) (by Remark 1). \(\square\)

**Lemma 4.** Any \(k\)-OSL DFST corresponds to a \(k\)-OSL function.

**Proof.** Let \(\mathcal{T} = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle\) be a \(k\)-OSL DFST computing \(f\) and let \(w_1, w_2 \in \Sigma^\ast\) such that \(\text{Suff}^k(f^P(w_1)) = \text{Suff}^k(f^P(w_2))\). Since \(\mathcal{T}\) is onward, by Remark 2 there exists \(q, q' \in Q\) such that \((q_0, \times w_1, f^P(w_1), q) \in \delta^\ast\) and \((q_0, \times w_2, f^P(w_2), q') \in \delta^\ast\). By Lemma 3, \(q = \text{Suff}^k(f^P(w_1)) = \text{Suff}^k(f^P(w_2)) = q'\) which implies \(\text{tails}_f(w_1) = \text{tails}_f(w_2)\). Therefore \(f\) is a \(k\)-OSL function. \(\square\)

We now need to show that every \(k\)-OSL function can be represented by a \(k\)-OSL DFST. An issue here is that one cannot work from \(\mathcal{T}_C\) since its states are defined in terms of its tails, which themselves are defined in terms of input strings, not output strings. Hence, the proof below is constructive.

**Theorem 3.** Let \(f\) be a \(k\)-OSL function. The DFST \(\mathcal{T}\) defined as followed computes \(f\):

- \(Q = S \cup \{q_0, q_f\}\) with \(S \subseteq \Delta^{\leq k-1}\)
- \((q_0, \times, u, \text{Suff}^{k-1}(u)) \in \delta \iff u = f^P(\lambda)\)
- \(a \in \Sigma\), \((q, a, u, \text{Suff}^{k-1}(qu)) \in \delta\) \iff \((\exists \gamma)[\text{Suff}^k(f^P(\gamma)) = q \land f^P(wa) = vqu]\) with \(v = f^P(w) \cdot \gamma^{-1}\).
- \((q, \times, u, q_f) \in \delta \iff u = f^P(w_q)^{-1} \cdot f(w_q)\), where \(w_q = \min \{w \mid \exists u, (q_0, \times w, u, q) \in \delta^\ast\}\).

The diagram below helps express pictorially how the transitions are organized per the second and third bullets above. The input is written above the arrows, and the output written below.

\[
\begin{array}{ccc}
q_0 \xrightarrow{\times w} q & \xrightarrow{a} & q' \\
\end{array}
\]
Note that $T$ is a $k$-OSL SFST. To prove this result, we first show the following lemma:

**Lemma 5.** Let $T$ be the transducer defined in Theorem 3. We have:

$$(q_0, \times w, u, q) \in \delta^* \iff f^p(w) = u$$

**Proof.** (⇒) By recursion on the length of $w$. Suppose $(q_0, \times w, u, q) \in \delta^*$ and $|w| = 0$. Then $(q_0, \times, u, q) \in \delta$; by construction, $q = \text{Suff}^{k-1}(u)$ and $f^p(\lambda) = u$ which validates the initial case.

Suppose the result holds for $w$ of size $n$ and pick such a $w$. Suppose then that $(q_0, \times wa, u, q) \in \delta^*$. By definition of $\delta^*$, there exists $u_1, u_2, q'$ such that $u = u_1u_2, (q_0, \times w, u_1, q') \in \delta^*$ and $(q', a, u_2, q) \in \delta$. We have $f^p(w) = u_1$ by recursion and thus $q' = \text{Suff}^{k-1}(f^p(w))$ (by Lemma 3).

By construction of $T$, $q = \text{Suff}^{k-1}(q'u_2)$ and thus $f^p(wa) = vq'u_2$ with $v = f^p(w) \cdot \text{Suff}^{k-1}(f^p(w))^{-1}$. Therefore $f^p(wa) = vq'u_2 = f^p(w) \cdot \text{Suff}^{k-1}(f^p(w))^{-1} f^p(w)$.

$(\Leftarrow)$ By recursion on the length of $w$. If $|w| = 0$, then $f^p(\lambda) = u$. By construction of $T$, $(q_0, \times, u, \text{Suff}^{k-1}(u)) \in \delta$, which validates the base case.

Now fix $n > 0$ and suppose the result holds for all $w$ of size $n$. Pick such a $w$ and let $f^p(wa) = u$. As $f$ is subsequential, there exists $u_1$ such that $f^p(w) = u_1$. By recursion, there exists $q$ such that $(q_0, \times w, u_1, q) \in \delta^*$. By Lemma 3, $q = \text{Suff}^{k-1}(u_1) = \text{Suff}^{k-1}(f^p(w))$. By definition $f^p(wa) = u_1u_2^{-1} \cdot u$, which equals $u_1 \cdot \text{Suff}^{k-1}(u_1)^{-1} \text{Suff}^{k-1}(u_1)u_1^{-1} \cdot u$. Hence $f^p(wa) = vqu'$, with $u' = u_1^{-1} \cdot u$ and $v = u_1 \cdot \text{Suff}^{k-1}(u_1)^{-1}$, which equals $f^p(w) \cdot \text{Suff}^{k-1}(f^p(w))^{-1}$. Thus, by construction $(q, a, u', \text{Suff}^{k-1}(qu')) \in \delta$. Since $u_1u' = u_1u_1^{-1} \cdot u = u, (q_0, \times wa, u, \text{Suff}^{k-1}(qu')) \in \delta^*$. □

Figure 3 presents a 2-OSL transducer that models Johore Malay nasal spreading. $\Sigma = \{C, N, V\}$ and $\Delta = \{C, N, V, V\}$.

6 Learning OSL functions

6.1 Learning criterion

We adopt the identification in the limit learning paradigm (Gold, 1967), with polynomial bounds on time and data (de la Higuera, 1997). The underlying idea of the paradigm is that if the data available to the algorithm does not contain enough information to distinguish the target from other potential targets, then it is impossible to learn.

We first need to define the following notions. A class $\mathcal{T}$ of functions is represented by a class $\mathbb{R}$ of representations if every $r \in \mathbb{R}$ is of finite size.
and there is a total and surjective naming function \( L : \mathbb{R} \to T \) such that \( L(r) = t \) if and only if for all \( w \in \text{pre_image}(t) \), \( r(w) = t(w) \), where \( r(w) \) is the output of representation \( r \) on the input \( w \). We observe that the class of \( k \)-OSL functions can be represented by the class of \( k \)-OSL DFSTs.

**Definition 8.** Let \( T \) be a class of functions represented by some class \( \mathbb{R} \) of representations.

1. A sample \( S \) for a function \( t \in T \) is a finite set of data consistent with \( t \), that is to say \((w,v) \in S \) iff \( t(w) = v \). The size of a sample \( S \) is the sum of the length of the strings it is composed of: \( |S| = \sum_{(w,v) \in S} |w| + |v| \).
2. A \((T,R)\)-learning algorithm \( \mathcal{A} \) is a program that takes as input a sample for a function \( t \in T \) and outputs a representation from \( R \).

The paradigm relies on the notion of characteristic sample, adapted here for functions:

**Definition 9 (Characteristic sample).** For a \((T,R)\)-learning algorithm \( \mathcal{A} \), a sample \( CS \) is a characteristic sample of a function \( t \in T \) if for all samples \( S \) for \( t \) it is the case that \( CS \subseteq S \) and \( \mathcal{A} \) returns a representation \( r \) such that \( L(r) = t \).

This definition is the one used in the proof of the OSTIA algorithm. The learning paradigm can now be defined as follows.

**Definition 10 (Identification in polynomial time and data).** A class \( T \) of functions is identifiable in polynomial time and data if there exists a \((T,R)\)-learning algorithm \( \mathcal{A} \) and two polynomials \( p() \) and \( q() \) such that:

1. For any sample \( S \) of size \( m \) for \( t \in T \), \( \mathcal{A} \) returns a hypothesis \( r \in R \) in \( O(p(m)) \) time.
2. For each representation \( r \in R \) of size \( n \), with \( t = L(r) \), there exists a characteristic sample of \( t \) for \( \mathcal{A} \) of size at most \( O(q(n)) \).

### 6.2 Learning algorithm

We show here that Algorithm 1 learns the OSL functions under the criterion introduced. We call this the Output Strictly Local Function Inference Algorithm (OSLFIA). We assume \( \Sigma, \Delta, \) and \( k \) are fixed and not part of the input to the learning problem.

Essentially, the algorithm computes a breadth-first search through the states that are reachable given the learning sample: the set \( C \) contains the states already checked while \( R \) is a queue made of the states that are reachable but have not been treated yet. Initially, the only transition leaving the initial state is writing the \( lcp \) of the output strings of the sample and reaches the state corresponding to the \( k-1 \) suffix of this \( lcp \). At each step of the main loop, OSLFIA treats the first state that is in the queue \( R \) and computes whenever possible the transitions that leave that state. The outputs associated with each added transition are the longest common prefixes of the outputs associated with the smallest input prefix in the sample that allows the state to be reachable. We show that provided the algorithm is given a sufficient sample the transducer outputted by OSLFIA is onward and in fact a \( k \)-OSL transducer.

After adding transitions with input letters from \( \Sigma \) to a state, the transition to the final state is added, provided it can be calculated.

### 6.3 Theoretical results

Here we establish the theoretical results, which culminate in the theorem that OSLFIA identifies the \( k \)-
OSL functions in polynomial time and data.

**Lemma 6.** For any input sample $S$, OSLFIA produces its output in time polynomial in the size of $S$.

**Proof.** The main loop is used at most $|\Delta|^{k-1}$ which is constant since both $\Delta$ and $k$ are fixed for any learning sample. The smaller loop is executed $|\Sigma|$ times. At each execution: first condition can be tested in time linear in $n$, where $n = \sum_{(w,u)\in S} |w|$: the computation of the $1cp$ can be done in $nm$ steps where $m = \max\{|w| : (w,u) \in S\}$ with an appropriate data structure (for instance a prefix tree); computing the suffix requires at most $m$ steps. The second condition can be tested in at most $\text{card}(S) \cdot m$ steps; the computation of the final transitions can be done in less than $m$ steps; all the other instructions can be done in constant time. The overall computation time is thus $O(|\Delta|^{k-1} |\Sigma| (n + nm + \text{card}(S) \cdot m + 2m)) = O(n + m(n + \text{card}(S)))$ which is polynomial (in fact bounded by a quadratic function) in the size of the learning sample.

Next we show that for each $k$-OSL function $f$, there is a finite kernel of data consistent with $f$ (a ‘seed’) that is a characteristic sample for OSLFIA.

**Definition 11 (A OSLFIA seed).** Given a $k$-OSL transducer $(Q,q_0,q_f,\Sigma,\Delta,\delta)$ computing a $k$-OSL function $f$, a sample $S$ is a OSLFIA seed for $f$ if

- For all $q \in Q$ such that $\exists v \in \Delta^*$ $(q,x,v,q_f) \in \delta$, $(\times w_q x, f(w_q)) \in S$, where $w_q = \min\{w : \exists u, (q_0, \times w, u, q) \in \delta^*\}$

- For all $(q,a,u,q') \in \delta$ with $q' \neq q_f$ and $a \in \{\times\} \cup \Sigma$, for all $b \in \Sigma$ such that there exists $(q',b,u',q'') \in \delta$, there exists $(\times w x, f(w)) \in S$ and $x \in \Sigma^*$ such that $w = w_q abx$ and $f(w)$ is defined. Also, if there exists $v$ such that $(q',x,v,q_f) \in \delta$ then $(\times w_q ax, f(w_q a)) \in S$.

In what follows, we set $T^\circ = \langle Q, q_0, q_f, \Sigma, \Delta, \delta^\circ \rangle$ be the target $k$-OSL transducer, $f$ the function it computes, and $T = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle$ be the transducer OSLFIA constructs on a sample that contains a seed.

**Lemma 7.** If a learning sample $S$ contains a seed then $(q_0, \times w, u, r) \in \delta^* \iff (q_0, \times w, u, r) \in \delta^*_S$.

**Proof.** $(\Rightarrow)$. By induction on the length of $w$. If $|w| = 0$ then $(q_0, x, u, r) \in \delta$ and so $u = 1cp\{y : (x,y) \in S\}$ and $r = \text{Suff}^{k-1}(u)$ (initial steps of the algorithm). As $S$ is a seed there is an element $(\times bx, f(bx)) \in S$ for all $b \in \Sigma$ and $(\times x, f(\lambda)) \in S$ if $\lambda \in \text{pre}_{\lambda}(f)$, which implies that $u = 1cp(f(\lambda^*))$. As the target is onward, we have $(q_0, x, 1cp(f(\lambda^*)), r') \in \delta$ and since it is a $k$-OSL DSFST $r' = \text{Suff}^{k-1}(1cp(f(\lambda^*))) = \text{Suff}^{k-1}(u) = r$.

Suppose the lemma is true for strings of length less than or equal to $n$. We refer to this as the first Inductive Hypothesis (IH1). Let $wa$ be of size $n + 1$ such that $(q_0, \times wa, u, r) \in \delta^*$. By definition of $\delta^*$, there exist $u_1, u_2, q$ such that $(q_0, \times w, u_1, q) \in \delta^*$, $(q, a, u_2, r) \in \delta$, and $u = u_1 u_2$. By IH1 $(q_0, \times w, u_1, q) \in \delta^*_S$. We want to show $(q_0, \times wa, u, r) \in \delta^*_S$ (i.e., $(q, a, u_2, r) \in \delta$).

First we show that IH1 also implies that $s = \text{smallest}(q)$ such that $s = \times w$. Since the algorithm searches breadth-first, $s$ is the smallest input that reaches $q$ in $T$. If $\times w < s$ then $\exists q' \neq q$ such that $(q_0, \times w, u', q') \in \delta^*$ because $\times w$ is a prefix of an input string of the sample $S$ (since $S$ contains a seed). Since $\times w < s$ and $|s| \leq n$, by IH1 then $(q_0, \times s, u', q') \in \delta^*_S$ which implies $q = q'$ which contradicts the supposition that $\times w < s$. If $s \leq \times w$, then again since $(q_0, \times s, u', q') \in \delta^*$ then by IH1 $(q_0, \times s, u', q') \in \delta^*_S$. This contradicts the definition of $w_q$. Therefore $s = \times w$.

Next we show that IH1 implies $f^p(w_q) = \text{out}(q)$. By construction of the seed, $(\times w_q x, f(w_q)) \in S$ if $\exists v \in \Sigma \cup \{\times\}$ $(q, q, v, q_f) \in \delta_0$ and $(\times w_q x, f(w_q a)) \in S$ for all transitions $(q, a, x, q')$ leaving $q$ in $T^\circ$. As the target is onward, $1cp\{x \mid (q, \sigma, x, q_0) \in \delta_0, \sigma \in \Sigma \cup \{\times\}\} = \lambda$ (Lemma 1). This implies $\text{out}(q) = 1cp\{y \mid \exists a \in \Sigma, x \in \Sigma^*\{\times\}, (\times w_q ax, y) \in S\} = 1cp\{y \mid \exists a \in \Sigma, x \in \Sigma^*\{\times\}, (\times w_q ax, y) \in S\} = 1cp\{f(w_q)\}$.

Recalling that $(q,a,w_2,r) \in \delta$, we now characterize $u_2$ to help establish $(q,a,w_2,r) \in \delta$.

By construction of a seed, there exist elements $(\times w_q abx, f(w_q abx))$ in $S$ for all possible $b \in \Sigma$ and an element $(\times w_q ax, f(w_q a)) \in S$ if $f(w_q a)$ is defined. By the onwardness of the target, this implies that $v = 1cp\{y \mid \exists b, x, (\times s a bx, y) \in S\} \cup \{f(sa)\} = 1cp(f(sa^*)) = f^p(sa)$. Therefore $u_2 = \text{out}(q)^{-1} \cdot v = f^p(s)^{-1} \cdot f^p(sa) = f^p(w_q)^{-1} \cdot f^p(w_q)$.

Finally we identify $r$ to complete this part...
of the proof. As the target is OSL, we have $(q_0, x_wu_a, f^p(w_a), r') \in \delta_0^*$ (Lemma 3). The fact that $(q_0, x_wq, f^p(wq), q) \in \delta_0^*$ by IH1 and the fact the target is OSL implies $(q, a, f^p(wq)^{-1}, f^p(wq_a), r') = (q, a, \text{out}(q)^{-1} \cdot s, r') \in \delta_0$. As $T^\circ$ is k-OSL, $r' = \text{Suff}^{k-1}(\text{out}(q)^{-1} \cdot s)$ which is $r$ by construction of the transitive in the algorithm. Therefore, as $(q_0, x_w, u_1, q) \in \delta_0^*$ by IH1, we have $(q_0, x_wa, u_1 \text{out}(q)^{-1} \cdot s, r) = (q_0, x_wu, u_1u_2, r) = (q_0, x_wu, u, r) \in \delta_0^*$.

$(\leftarrow)$. This is also by induction on the length of $w$. If $|w| = 0$, as $T^\circ$ is onward we have $\text{1cp}(\text{output}(q_0)) = 1\text{cp}(f(\Sigma^*))$ (Lemma 1) and thus $(q_0, x, 1\text{cp}(f(\Sigma^*)), r) \in \delta_0^*$ with $r = \text{Suff}^{k-1}(1\text{cp}(f(\Sigma^*)))$ as $T^\circ$ is k-OSL. By construction of the seed, there is at least one element in $S$ using each transition leaving $r$. As $1\text{cp}(\text{output}(r)) = \lambda$ (Lemma 1), this implies $1\text{cp} \{ y \mid (x, y) \in S \} = 1\text{cp}(f(\Sigma^*))$. Therefore $(q_0, x, 1\text{cp}(f(\Sigma^*)), r) \in \delta_0^*$.

Suppose the lemma is true for all strings up to length $n$. We refer to this as the second Inductive Hypothesis (IH2). Pick $wa$ of length $n + 1$ such that $(q_0, x_wa, u, r) \in \delta_0^*$. By definition of $\delta_0^*$, $q, u_1, u_2 \exists$ such that $(q_0, x_w, u_1q) \in \delta_0^*$ and $(q, a, u_2r) \in \delta_0^*$ with $u_1u_2 = u$. By IH2, we have $(q_0, x_w, u_1q), (q_0, x_wu_1r, q) \in \delta_0^*$ (since $w_q < w$). We want to show $(q, a, u_2r) \in \delta_0^*$.

We first show that $s = \text{smallest}(q) = x_wq$. Suppose $s < x_wq$. By construction of the SFST $s$ is a prefix of an element of $S$ which means there exists $q'$ such that $(q_0, s, f^p(s), q') \in \delta_0^*$. But by IH2, this implies that $q' = q$ and the definition of $w_q$ contradicts $s < x_wq$. Suppose now that $x_wq < s$. By the construction of the seed, $x_wq$ is a prefix of an element of the sample, which implies it is considered by the algorithm. As $(q_0, x_wq, u_1q) \in \delta_0^*$ by IH2, $x_wq$ is a smaller prefix than $s$ that reaches the same state which is impossible as $s$ is the earliest prefix that makes the state $q$ reachable. Therefore $x_wq = s$ and thus the transition from state $q$ reading $a$ is created when $s = x_wq$.

Next we show that $f^p(w) = \text{out}(q)$. By construction of the seed, there is an element $(x_wqaw', q, f(w_qaw')) \in S$ for all transitions $(q, a, x, w') \in \delta_0$ leaving $q$ and $(x_wqaw', f(w_qaw')) \in S$ if $\exists w$, $(q, x, v, qf) \in \delta_0$. As the target is onward, $1\text{cp} \{ y \mid (q, x, q, x, q) \in \delta_0^*, q, x, \sigma \in \Sigma \cup \kappa \} = \lambda$ (Lemma 1). This implies $\text{out}(q) = 1\text{cp} \{ y \mid \exists a, x, \sigma \in \Sigma, y \in S \} = 1\text{cp}(f(w_q\Sigma^*)) = f^p(w_q) = f^p(s)$.

Now let $v = 1\text{cp} \{ y \mid \exists b, x, (sabx, y) \in S \}$. Since $s = x_wq, (q_0, x_wq, v, r) \in \delta_0^*$ since, as before, the onwardness of the target implies $1\text{cp}$ of the output written from $r$ is $\lambda$. This is because each possible output from $r$ is in $S$ (because it is in the seed according to the second item of Definition 11). Consequently $v = f^p(w_qa) = f^p(sa)$.

Together these results imply that $u_2 = f^p(w_q)^{-1} \cdot f^p(w_qa) = f^p(s)^{-1} \cdot f^p(sa) = \text{out}(q)^{-1} \cdot v$.

As the target is a k-OSL transducer (and thus deterministic) $\text{Suff}^{k-1}(q_{u_2}) = r$. Therefore the transition $(q, a, \text{out}(q)^{-1} \cdot v, r)$ that is added to $\delta$ is the same as the transition $(q, a, u_2r) \in \delta_0^*$. This implies $(q_0, x_wu, u, r) \in \delta_0^*$ and proves the lemma.

**Lemma 8.** Any seed for the OSL Learner is a characteristic sample for this algorithm.

**Proof.** A corollary of Lemma 7 is that if a seed is contained in a learning sample we have $(q_0, x_wu, u, q) \in \delta_0^* \iff f^p(w) = u$ (Lemma 3) as the target transducer is k-OSL. For all states $q$ where $\exists v, (q, x, v, q_f) \in \delta_0$, we have $(x_wq(x, f(w_q))$ in the seed, which implies the algorithm will add $(q, x, f^p(w_q)^{-1} \cdot f(w_q), q_f)$ to $\delta$ which is exactly the output function of the target. As every state is treated only once, this holds for any learning set containing a seed. Therefore, from any super-set of a seed, for any $w$, the function computed by outputted transducer of Algorithm 1 is equal to $f^p(w) f^p(w)^{-1} \cdot f(w) = f(w)$.

Observe that OSLFIA is designed to work with seeds, which contains minimal strings. We believe both the seed and algorithm can be adjusted to relax this requirement, though this is left for future work.

**Lemma 9.** Given any k-OSL transducer $T^\circ$, there exists a seed for the OSL learner that is of size polynomial in the size of $T^\circ$.

**Proof.** Let $T^\circ = (Q_0, q_0, q_f, \Sigma, \Delta, \delta_0, \cdot)$. There are at most $\text{card}(Q_0)$ pairs $(x_wq(x, f(w)))$ in a seed that corresponds to the first item of Definition 11, each of which is such that $|x_wq| \leq \text{card}(Q)^o$.
and \(|f(w_q)| \leq \sum_{(q, \sigma, u, q') \in \delta} |u|\). We denote by \(m_\diamond\) this last quantity and note that \(m_\diamond = O(|T_\diamond|)\).

For the elements of the second item of Definition 11 we restrict ourselves without loss of generality to pairs \(\langle q, \sigma, u, q' \rangle \in \delta \diamond\) where \(w' = \min_{<} \{ x : f(w_qabx) \text{ is defined} \}\). We have \(|w'| \leq \text{card}(Q_\diamond)\) and \(|f(w_qabw')|\) is in \(O(\text{card}(Q_\diamond)m_\diamond)\). There are at most \(|\Sigma|\) pairs \(\langle q, \sigma, u, q' \rangle \in \delta \diamond\) for a given transition \((q, a, u, q')\) which implies that the overall bound on the number of such pairs is in \(O(|\Sigma|\text{card}(\delta))\).

The overall length of the elements in the seed that fulfill the second item of the definition is in \(O(\text{card}(Q_\diamond)(\text{card}(Q_\diamond) + m_\diamond + |\Sigma|\text{card}(\delta)m_\diamond))\).

The size of the seed studied in this proof is thus in \(O((m_\diamond + |Q_\diamond|)(|Q_\diamond| + |\Sigma|\text{card}(\delta)m_\diamond))\) which is polynomial (in fact quadratic) in the size of the target transducer. \(\square\)

**Theorem 4.** OSLFIA identifies the \(k\)-OSL functions in polynomial time and data.

**Proof.** Immediate from Lemmas 6, 7, 8, and 9. \(\square\)

We conclude this section by comparing this result to other subsequential function-learning algorithms.

OSTIA (Oncina et al., 1993) is a state-merging algorithm which can identify the class of total subsequential functions in cubic time. (Partial subsequential functions cannot be learned exactly; for a partial function, OSTIA will learn some superset of it.) \(k\)-OSL functions include both partial and total functions, so the classes exactly learnable by OSTIA and OSLFIA are, strictly speaking, incomparable.

SOSFIA (Jardine et al., 2014) identifies subclasses of subsequential functions in linear time and data. These subclasses are determined by fixing the structure of a transducer in advance. For every input string, SOSFIA knows exactly which state in the transducer is reached. The sole carrier of information regarding reached states is the input string. But for \(k\)-OSL functions, the output strings carry the information about the states reached. As the theorems demonstrate, the destination of a transition is only determined by the output of the transition. Thus no class learned by SOSFIA contains any \(k\)-OSL class.

OSTIA-D (OSTIA-R) (Oncina and Varò, 1996; Castellanos et al., 1998) identify a class of subsequential functions with a given domain \(D\) (range \(R\)) in at least cubic time because it adds steps to OSTIA to prevent merging states that would result in a transducer whose domain (range) is not compatible with \(D\) (\(R\)). OSTIA-D cannot represent \(k\)-OSL functions for the same reasons SOSFIA cannot: domain information is about input strings, not output strings. On the other hand, the range of a \(k\)-OSL function is a \(k\)-OSL stringset which can be represented with a single acceptor, and thus OSL functions may be learned by OSTIA-R. However, OSLFIA is more efficient both in time and data.\(^3\)

To sum up, OSLFIA is the most efficient algorithm for learning \(k\)-OSL functions.

### 7 Phonology

The example of Johore Malay nasal spreading given in §2 is an example of progressive spreading, since it proceeds from a triggering segment (the nasal) to vowels and glides that follow it. There also exist regressive spreading processes, in which the trigger follows the target(s). An example from the Môba dialect of Yoruba (Ajibóyè, 2001; Ajibóyè and Pulleyblank, 2008; Walker, 2014) is shown in (2). An underlying nasalized vowel spreads its nasality to preceding oral vowels and glides.

\[(2) /\hat{u}j\hat{\bar{u}}l/ \rightarrow [\hat{u}\hat{j}\bar{u}]\], ‘praise(n.)’\]

The difference between progressive and regressive spreading corresponds to reading the input from left-to-right or right-to-left, respectively (Heinz and Lai, 2013). Regressive spreading cannot be modeled with OSL in a left-to-right fashion, because the output of the preceding vowels and glides depends on the presence or absence of a following nasal that could be an unbounded number of segments away. By reading from right-to-left, that nasal trigger will always be read before the target(s), making it akin to progressive spreading. Thus there are two overlapping but non-identical classes, which we call left(-to-right) OSL and right(-to-left) OSL.

There are other types of phonological maps that are neither ISL nor OSL. Consider the optional process of French a-deletion shown in (3) (Dell, 1973; Dell, 1980; Dell, 1985; Noske, 1993).

\[^{3}\text{To our knowledge no analysis of data complexity for OSTIA, OSTIA-D, and OSTIA-R has been completed (probably because they predate de la Higuera (1997)). Also, an analysis of the data complexity of OSTIA appears daunting.}\]
At issue is how this rule applies. There are two licit pronunciations of /ty davone/ ‘you became’ which are [ty dv̩n̩] and [ty dv̩n̩r̩]. The form * [ty dv̩n̩r̩] is considered ungrammatical. As Kaplan and Kay (1994) explain, these outputs can be understood as the rule in (3) applying left-to-right ([ty dv̩n̩r̩]), right-to-left ([ty dv̩n̩r̩]) or simultaneously (* [ty dv̩n̩r̩]). What matters is whether the left and right contexts of the rule match the input or output string: if both match the input it is simultaneous application, and if one side matches the input and the other the output it is left-to-right or right-to-left.

ISL functions always match contexts against the input and therefore they cannot model α-deletion. In this respect, ISL functions model simultaneous rule application. But there is also a problem with modeling the process as OSL, which is what to output when the α that will be deleted is read. Consider the input VCαCV. When the DSFST reads the α, it cannot decide what to output, because whether or not that α is deleted depends on whether or not the next two symbols in the input are CV. But since the DSFST is deterministic, it must make a decision at this point. It could postpone the decision and output λ. But that would require it to loop at the current state (Lemma 2), which in turn means it cannot distinguish VCαCV from VCββCV, a significant problem since only the former meets the context for deletion.

Thus the range of phonological processes that can be modeled with OSL functions is limited to those with one-sided contexts (e.g., either C__ or __D, the former being left OSL and the latter right OSL). In such cases the entire triggering context will be read before the potential target, so there is never a need to delay the decision about what to output. To summarize, phonological rules that apply simultaneously are ISL, and phonological rules with one-sided contexts that apply left-to-right or right-to-left are OSL.

In addition to iterative rules with two-sided contexts, long-distance processes like vowel harmony and consonant agreement and dissimilation are also excluded from the current analysis. While such process have been shown to be subsequential and therefore subregular (see Gainor et al. (2012; Luo (2014; Payne (2013; Heinz and Lai (2013)) they are neither ISL nor OSL because the target and triggering context are not within a fixed window of length k in either the input or output. An example is the long-distance nasal assimilation process in Kikongo (Rose and Walker, 2004), as in (4).

(4) /tu+nik+idi/ → [tunikini] ‘we ground’

In Kikongo, the alveolar stop in the suffix /-idi/ surfaces as a nasal when joined to a stem containing a nasal. Since stem nasals appear to occur arbitrarily far from the suffix, there is no k such that the target /d/ and the trigger /n/ are within a window of size k. Thus the process is neither ISL nor OSL.

8 Future Work

Processes like French α-deletion that have two-sided contexts, with one being on the output side, suggest a class that combines the ISL and OSL properties. We are tentatively calling this class ‘Input-Output SL’ and are currently working on its properties, FST characterization, and learning algorithm. For long-distance processes, we expect other functional subclasses will strongly characterize these. SL stringsets are just one region of the Subregular Hierarchy (Rogers and Pullum, 2011; Rogers et al., 2013), so we expect functional counterparts of the other regions can be defined. Some of these other regions model long-distance phonotactics (Heinz, 2007; Heinz, 2010; Rogers et al., 2010), so their functional counterparts may prove equally useful for modeling and learning long-distance phonology.

9 Conclusion

We have defined a subregular class of functions called the OSL functions and provided both language-theoretic and automata-theoretic characterizations. The structure of this class is sufficient to allow any k-OSL function to be efficiently learned from positive data. It was shown that the OSL functions—unlike the ISL functions—can model local iterative spreading processes. Future work will aim to combine the results for both ISL and OSL to model iterative processes with two-sided contexts.

Acknowledgments

We thank three reviewers for useful comments, especially the third, who caught a significant error in the first version of this paper.
References


Huan Luo. 2014. Long-distance consonant harmony and subsequentiality. Qualifying paper for the University of Delaware’s Linguistics PhD Program.


Amanda Payne. 2013. Dissimilation as a subsequential process. Qualifying paper for the University of Delaware’s Linguistics PhD Program.


