Learning k-ISL functions with phonological features

Magdalena Markowska and Jeffrey Heinz
Stony Brook University
Institute for Advanced Computational Science
Stony Brook, NY, USA

ABSTRACT

This study investigates the grammatical inference of phonological processes from *featural* representations, focusing on the class of k-Input Strictly Local (k-ISL) functions (Chandlee *et al.* 2014; Chandlee and Heinz 2018). We present the *Factor-by-Feature (FxF)* framework which decomposes the learning task into smaller ones along featural dimensions. The FxF framework is compatible with any grammatical inference algorithm designed to infer string-to-string functions in terms of finite-state transducers. The Structured Onward Subsequential Inference Algorithm (SOSFIA) is one such grammatical inference algorithm which has proven learnability results in linear time and data (Jardine *et al.* 2014). We empirically evaluate FxF on five phonological processes across four languages: English, Yawelmani, Chukchi, and Polish, representing cases of feature spreading, feature changing, and feature insertion. Comparisons are provided in each case between learning these processes over segments with SOSFIA alone and with SOSFIA in the FxF framework. To better measure data efficiency, we also introduce AMβA, a sampling algorithm that identifies smaller characteristic samples which, informally, are sets of data that guarantee successful generalization. Our results demonstrate that FxF consistently yields more compact and learnable data representations, significantly reducing the size of characteristic samples. These findings underscore the

Keywords: phonological representations, phonological features, grammatical inference, finite-state transducers, Input Strictly Local functions

importance of phonological representation and support linguistically informed learners for phonological grammars.

1 INTRODUCTION

This article shows how to incorporate phonological representations, in particular phonological features, into grammatical inference algorithms to make them more effective for the learning of phonological processes. This result is important not only because it uses representational primitives that are central to phonological theory, but also because the grammatical inference algorithms tend to come with some theoretical guarantees. For example, we can say with certainty that algorithm A, when given certain data D exemplifying any process P from a certain class of processes C, will return a grammar for P. In this way, the approach adopted here gives equal weight to understanding the general behavior of a learning algorithm (as expressed in theorems about its behavior) and to representational commitments of phonological theory.

Broadly speaking, this approach contrasts with other valid and interesting approaches to learning phonological processes. For example, Belth (2023) develops the Parsimonious Local Phonology (PLP) learner that learns phonological grammars formalized as a series of one or more phonological rules. While this program performs well on a variety of experiments, and employs phonological representations such as features, it lacks theoretical guarantees regarding its general behavior. It is not known, for instance, precisely what class of processes can be learned by such programs.

In the constraint-based tradition, the picture is somewhat different. Optimality-Theoretic grammars incorporating phonological representations can be learned with Recursive Constraint Demotion (RCD), an algorithm that does come with theoretical guarantees (Tesar and Smolensky 1998, 2000; Prince and Tesar 2004). However, it is still not known which class of processes is being learned. All that is known is that the class belongs to the factorial typology given by the set of constraints but this typology is relatively unconstrained in the

kinds of patterns it may include, even when the constraints themselves are significantly restricted (Lamont 2021, 2022).

The neural network (NN) tradition also includes interesting experimental results (Begus 2020; Prickett 2021). Nevertheless, it remains an active area of research whether such systems encode phonological representations such as features in some way, what kinds of processes such systems can express, and what kinds of processes such systems can learn from data.

On the other hand, the grammatical inference (GI) tradition develops algorithms whose principal aim is to understand the conditions under which a learner can generalize appropriately from data (de la Higuera 2010; Heinz *et al.* 2015; Heinz and Sempere 2016; Wieczorek 2017). This tradition answers questions like "What algorithm A (if any), when given certain data D exemplifying a process P drawn from a certain class of processes C, will invariably return a grammar for P?" However, some have argued that the utility of these algorithms is limited in the domain of phonology (Gildea and Jurafsky 1996).

The overall advantages and disadvantages of these approaches are summarized in Table 1. This summary should not be taken literally: whether a particular approach satisfies a given criterion is not as black-and-white as the table suggests. In reality, the cells would fall along a spectrum (i.e., various shades of gray). Nonetheless, the table reflects our overall understanding of the literature, and we believe the Yes/No entries capture the main strengths and weaknesses of each approach. A valuable research goal in any of these traditions is to develop results that turn "No" cells into "Yes" cells. This article shows how to turn the upper-right cell of Table 1 from "No" to "Yes." Future work will test the approach on the kinds of experiments used in earlier studies.

Table 1: Learning criteria and their inclusion in different traditions of learning phonological processes

Criteria	PLP	RCD	NN	GI
Uses phonological representations	Yes	Yes	No	No
Performs well on learning experiments	Yes	Yes	Yes	No
Theorems exist about learnable class	No	No	No	Yes
Theorems exist about learning behavior	No	Yes	No	Yes

1.1 Formal Grammars and Grammatical Inference

It has been known for decades that phonological processes can be described with finite-state machines (Johnson 1972; Kaplan and Kay 1994). Finite-state acceptors recognize exactly the regular languages, and finite-state transducers compute the corresponding regular relations (string-to-string mappings). That means that every regular language or relation can be implemented by some finite-state machine. Regular languages and relations thus provide an upper bound on the complexity of phonological grammars (Heinz 2011a).

Recent work in *subregular* phonology further shows that attested phonological patterns occupy highly structured proper subclasses of these regular languages and relations (Heinz 2010a, 2011b, 2018, 2025). In particular, many phonological processes can be modeled with *subsequential* functions, which are a proper subset of regular string-to-string mappings (Heinz and Lai 2013; Chandlee *et al.* 2014; Chandlee 2017; Luo 2017; Payne 2017; Chandlee *et al.* 2018), and can be instantiated with deterministic finite-state transducers (DFTs) (Sakarovitch 2009).[1] A wide range of attested processes fall into a subclass of those functions, particularly, the class of *k-Input Strictly Local* functions (Chandlee 2014; Chandlee and Heinz 2018). This class bounds the relevant contexts for phonological changes to memory windows of size k. These formal restrictions align with empirical typology and provide insights into computational properties of phonological processes. Extensions of this class have been used to account for nonlocal phonological processes (Burness and McMullin 2020; Burness *et al.* 2021).

Importantly, the subregular perspective also provides formal learnability results. Subsequential functions are identifiable in the limit (Gold 1967) in cubic time and data (Oncina *et al.* 1993). Despite this theoretical advance, the data samples sufficient for successful generalization are typically larger than – and qualitatively distinct from – realistic linguistic data (Gildea and Jurafsky 1996). In practice, human learners are exposed to limited input (Lignos and Yang 2016), yet they successfully acquire phonological grammars without explicit

[1] These transducers are also called *sequential* in the literature.

instructions (Yang 2013). This discrepancy between the learnability results and the actual language acquisition is one of the primary motivations for this research. More *data-efficient* learners may not only bridge the gap between theory and practice, but can also address the under-representation of low-resource languages in language technology (Wiemerslage *et al.* 2022; Nag *et al.* 2024).

Consequently, recent research in grammatical inference focuses on identifying subclasses of sequential functions with better bounds on time and data. Chandlee *et al.* (2014) show that the Input Strictly Local Function Learning Algorithm (ISLFLA) can learn k-ISL functions in quadratic time and data. Jardine *et al.* (2014) introduce the Structured Onward Subsequential Function Inference Algorithm (SOSFIA) that provides a recipe for learning particular subclasses of subsequential functions, including the k-ISL functions, in *linear* time and data. SOSFIA can also be used for learning phonological processes which are not k-ISL, but belong to other related classes, such as the Reverse Definite and the Tier-based (Reverse) Definite classes (Lambert and Heinz 2024). Such learners can therefore serve a dual role: they bring formal learnability results closer to the conditions of human language acquisition, and they remain effective when only small datasets are available for low-resource languages.

More technically, as discussed in the grammatical inference literature (de la Higuera 2010; Heinz *et al.* 2015; Heinz and Sempere 2016; Wieczorek 2017), a dataset D is a *characteristic sample* for a learning algorithm A and a target concept C provided that for any set of data S including D ($D \subseteq S$) it is the case that A returns an accurate representation of C given S. Informally, the presence of a characteristic sample in the data guarantees a successful generalization. Furthermore, if there is a function f, such that A runs in f time and data, then this means that the running time of A is bounded by $f(|S|)$ and that the size of a characteristic sample for any target concept C is bounded by $f(n)$ where n is the size of the representation of C. Since we are concerned with subsequential transducers, n is the size of the transducer. Thus, for k-ISL functions, SOSFIA is thought to have smaller characteristic samples than, for example, the Onward Subsequential Transducer Induction Algorithm (OSTIA; Oncina *et al.* 1993), because of the tighter theoretical bounds on time and data (linear for SOSFIA and cubic for OSTIA).

1.2 Contribution

This work approaches the problem of reducing requirements for successful generalization by incorporating phonological representations, particularly phonological features. As discussed in more detail below, phonological features are representational primitives widely adopted by phonological theories (Hayes 2009; Zsiga 2013; Bale and Reiss 2018). Building on the feature-based learning strategy introduced by Markowska and Heinz (2023), we introduce the *Factor-by-Feature* (FxF) framework that decomposes the learning problem, such that the target function is understood as a collection of functions, one for each phonological feature, each of which must be inferred. Inference is conducted over values associated with each feature, as opposed to fully specified segments. Such factorization allows a reduction in the size of the alphabet, which we show directly affects the size of the transducer, and therefore the size and quality of the characteristic sample. The FxF framework can be used with any learner suitable for phonological grammars. Here we instantiate FxF with SOSFIA (Jardine *et al.* 2014) and compare it directly to a baseline in which SOSFIA is applied to segmental representations of the data without featural factorization.

The FxF approach is empirically evaluated on four phonological alternations across four distinct languages: English vowel nasalization, Yawelmani vowel shortening, Chukchi final schwa epenthesis, and an opaque interaction of final devoicing and o-raising in Polish.[2] In each case, we compare the segment-based SOSFIA to the feature-based one. The results show that FxF consistently requires significantly fewer training examples to achieve the desired generalization. For a 2-ISL process with one feature change (such as English), we observe a reduction of up to 98% in the size of a characteristic sample. In more challenging cases, such as epenthesis and interaction of processes affecting multiple features, the reduction was at best 86% for Chukchi and 52% for Polish, given a limited input alphabet of size 7. These findings highlight the practical benefit of incorporating phonological representations into the learner and support the view that linguistically informed algorithms can be both formally rigorous and data-efficient.

[2] Code and materials are available at: `https://github.com/m-markowska/feature-learning-SOSFIA`.

This paper is organized as follows. Section 2 reviews the role of features in phonology and grammatical inference, and provides an overview of learners capable of inferring classes of sequential functions. Section 3 supplies the necessary formal definitions and notations. Section 4 introduces the factoring by feature idea and explains how phonological functions can be decomposed along featural dimensions. Section 5 provides theoretical explanation for why a learning strategy like FxF can, in principle, reduce the size of characteristic samples. Section 6 describes the algorithms central to this work. Section 7 presents the empirical evaluation of the approach in the aforementioned processes with and without the FxF approach. Section 8 offers further discussion of the novel framework and its limitations, as well as future work. Finally, Section 9 concludes with a summary of the contributions of the paper.

RELATED WORK 2

Importance of Phonological Features 2.1

Phonological features have played a central role in linguistic theory, tracing back to the seminal work of Roman Jakobson and Nikolai Trubetzkoy (Jakobson *et al.* 1951; Trubetzkoy 1969) and extensively developed in the subsequent theoretical framework, such as SPE (Chomsky and Halle 1968). Features group speech sounds into natural classes, which both simplifies the description of phonological rules and enables generalization across typologically diverse languages (Clements and Hume 1995). For example, a process which targets *voiced obstruents* can be characterized as a natural class with the representation ($\left[\begin{smallmatrix}+\\-\end{smallmatrix}\ \begin{smallmatrix}\text{voice}\\\text{sonorant}\end{smallmatrix}\right]$) rather than listing individual segments, such as [b d g v z ʒ d͡ʒ ɣ], within that class. Furthermore, features also make it possible to represent phonemic distinctions at various levels of representations (Kaye 1980; Rubach 2019). For example, in English, the distinction between aspirated and unaspirated voiceless stops is merely a surface one, while for native Thai speakers aspiration is a contrastive feature to distinguish the types of voiceless obstruents in the mental representation (Hyman 1975, pp. 26–27).

Beyond their role in stating phonological rules, there is independent evidence that features are needed to characterize the organization of phonological systems themselves. Typological studies of segmental inventories argue that cross-linguistic generalizations are best captured in terms of a set of distinctive features and some principles aimed at organizing them (e.g., feature economy, robustness, phonological enhancement), rather than in terms of segments (Avery and Idsardi 2001; Clements 2009). Complementary work in the contrastive hierarchy tradition posits that only contrastive features, which are arranged in a language-specific hierarchy, participate in phonological computation. Within this framework, such hierarchies are necessary to account for patterns such as neutralization or harmony in an empirically adequate and logically coherent way (Dresher 2009, pp. 11–35). In this view, it is not just the mere existence of features, but also their contrastive status, that is central to the structure of phonological grammars.

The importance of a featural level of representation is also corroborated by psycholinguistic evidence (Du and Durvasula 2025) and speech-error studies (Fromkin 1971), as well as by work on segment inventories, phonological alternations, and phonetic implementation (Cohn 2011). These pieces of evidence support the idea that feature-based representations play an important role in phonology, independently of any particular theory of learning. At the same time, the nature of features remains a subject of debate (Duanmu 2016). Traditional generative approaches argue that features are innate and are part of the learner's prior knowledge, on the grounds that some representational primitives are a logical precondition for any learning (e.g. Chomsky and Halle 1965, 1968; Fodor 1980; Reiss and Volenec 2022). We adopt this point of view here, while acknowledging that some researchers aim to infer features and natural classes from distributional patterns in the input (Mielke 2008; Mayer 2020). In practice, many computational models assume, as we do, that a feature system is 'baked' into phonological models (Gildea and Jurafsky 1996; Hayes and Wilson 2008).

Because phonological features have so much independent support, employing featural representations to learn phonological processes has ample precedents within the field of phonology. For example, Albright and Hayes (2002, 2003) provided a learning model

which represents segmental changes with those sets of features that are the smallest natural class consistent with them. More recently, Belth (2023, 2024) presents a rule-learning model which incorporates the Tolerance Principle (Yang 2016) and shows that it can generalize less conservatively. Despite the merits of these learning models and the simulations conducted with them, they do not have theoretical results with respect to their general learning behavior.

This paper adopts an alternative strategy: it begins with learning algorithms in the grammatical inference tradition, which come with some theoretical guarantees, and adapts them to incorporate features. In this way, this work bridges two communities: for the phonological community, it helps introduce them to results in grammatical inference, and for the grammatical inference community, it shows how to incorporate sub-symbolic structure into their algorithms.

Strother-Garcia *et al.* (2016) and Vu *et al.* (2018) show how model-theoretic approaches to formal languages enable one to naturally provide grammars which make reference to phonological features. Building on this work, Chandlee *et al.* (2019) and Rawski (2021) introduce the Bottom-Up Factor Inference Algorithm (BUFIA) which provably learns formal languages describable as the conjunctions of constraints, where the constraints are defined over arbitrary, but given, model-theoretic signatures. Recently, BUFIA has been shown to be practically effective for learning both phonotactic constraints (Swanson *et al.* to appear) and tonotactic constraints (Li 2025).

The line of research in this paper differs from this earlier research in two ways. First, it targets phonological processes (transformations of strings to strings) as opposed to phonotactics (well-formedness of strings). Second, the aforementioned works used logical representations of grammars, whereas this study uses finite-state representations of grammars. Related work by Yolyan (2025a,b) incorporates the ideas underlying BUFIA into a learning model for phonological processes described with the logical formalism known as Boolean Monadic Recursive Schemes (Bhaskar *et al.* 2020; Chandlee and Jardine 2021). A detailed comparison between that approach and the one adopted here is left for future research.

2.2 Subregular Classes in Phonology

This work stems from the idea that phonology is computationally simple. Heinz (2011a,b, 2018, 2025) provide an overview and arguments for this idea. On the stringset side (phonotactics), Heinz (2010a) argues that local phonotactics belongs to the Strictly Local class and that long-distance phonotactic patterns belong to the Strictly Piecewise class. As such, well-formedness depends only on contiguous substrings or non-contiguous subsequences. Subsequent logic-based characterizations (Rogers and Pullum 2011; Rogers *et al.* 2013) and the consideration of the Tier-Based Strictly Local class (Heinz *et al.* 2011; McMullin 2016; Lambert and Rogers 2020; Lambert 2023) further reinforce that phonotactic patterns occupy a very restricted complexity range. Furthermore, these classes, when parameterized, can be learned by simple mechanisms (Heinz 2010b; Heinz *et al.* 2012; Jardine and Heinz 2016; Jardine and McMullin 2017; Lambert *et al.* 2021). Additional related classes for phonotactics are studied by Lambert (to appear).

Of particular relevance to this study are subregular classes of string-to-string functions that can model various phonological processes. Long established results in computational linguistics show that phonological rules can be represented with subsequential functions and modeled with deterministic finite-state transducers (Kaplan and Kay 1994; Sakarovitch 2009). Chandlee (2014) and Chandlee and Heinz (2018) demonstrate that many phonological processes fall into small subclasses of sequential functions: Input Strictly Local (ISL) and Output Strictly Local (OSL). The outputs in ISL functions are determined based on a bounded window of the *input* context, while in OSL functions the outputs depend on a bounded window of the *output* context. More complex processes might require projection of relevant elements onto tiers or other formal devices (Jardine 2016; McCollum *et al.* 2020; Burness *et al.* 2021; Lambert 2022; Lambert and Heinz 2023, 2024).

The subregular classification of phonological mappings offers significant advantages for learnability. When a learner is equipped with the prior knowledge that the target function lies within a well-defined subregular class, the hypothesis space can be constrained accordingly,

which can enable efficient generalizations. Such restrictions not only reduce the risk of overfitting but also provide formal guarantees of convergence and correctness. In the next subsection, we provide a review of algorithms designed to identify classes of sequential functions.

2.3 *Subsequential function classes learners and their limitations*

Oncina *et al.* (1993) introduce OSTIA, which successfully identifies any total subsequential function in the limit, given a positive characteristic sample. OSTIA enforces *onwardness*, a property which guarantees that outputs are produced as soon as they are determined. It constructs a prefix tree transducer and merges compatible states, which allows for further generalization to unseen data. While the algorithm runs in polynomial time, its cubic time and data complexity can be computationally expensive in practice (Gildea and Jurafsky 1996).

Chandlee *et al.* (2014) present the ISLFLA algorithm that specifically learns the class of k-ISL functions. The learner, like OSTIA, builds a prefix tree transducer and merges states to produce a transducer with the appropriate structure. The counterpart for k-OSL functions was developed by Chandlee *et al.* (2015) and is called OSLFLA. It generalizes by constructing a finite-state transducer, iteratively exploring the states reachable from an initial state and associating transitions with the longest common prefix of the outputs observed in the training data. Both learners run in quadratic time and data. Thus, in practice, if one knows that the target process belongs to the k-ISL or k-OSL functions, then efficiency considerations would favor the use of ISLFLA and OSLFLA over OSTIA.

Jardine *et al.* (2014) introduce SOSFIA, which identifies particular subclasses of subsequential functions in *linear* time and data. SOSFIA assumes that the underlying structure of the transducer modeling the target function is known in advance. For many subregular classes, such as k-ISL functions, this is, in fact, the case: the class determines the structure of the transducer. Consequently, the learning problem is reduced to calculating the outputs for each transition. Due to its efficiency relative to other learners, we adopt SOSFIA as the basis for illustrating the Factor-by-Feature approach proposed in this paper. Further technical details on the algorithm are provided in Section 6.

The learners discussed above show how the structure of the hypothesis can facilitate successful generalization. As Heinz (2010a, p. 636) argues, whether a grammar employs featural representations or not is orthogonal to other formal properties of the grammar, such as whether it describes a string-to-string mapping that is sequential, ISL, or OSL. Thus, the focus of the above works was on the contributions such formal properties can make to learning. That said, the aforementioned learners all share a core limitation in practice. In particular, they appear to require large, highly specific characteristic samples. Gildea and Jurafsky (1996, p. 507) note that the characteristic sample "may require types of strings that are not found in the language for phonotactic or other reasons." For example, it may require a string containing a *ttt* sequence. The practical consequence is that actual data sets do not contain characteristic samples. For learners that construct transducers from the data where each encountered data point has its representative path, such as OSTIA or ISLFLA, missing data points lead to failure in generalization or incomplete functions. SOSFIA, on the other hand, technically refuses to output a transducer in the absence of a characteristic sample, though heuristics can be added to ensure that it outputs something.

Learners operating over symbols representing segments may not generalize across the segments that share some phonological feature specifications. Put differently, symbols such as [b], [d], [g] are seen as entirely separate units, and their crucial similarity of being voiced obstruents is lost. This contrasts with phonological theory in which rules and constraints are described in terms of natural classes, and not segments in isolation. The FxF framework directly addresses this issue by shifting the learning problem from segment-level to feature-level processes. By identifying the target function in terms of each individual feature, FxF allows for generalization across any segment sharing particular feature values, which in turn reduces the data and improves interpretability. In this way, it directly synthesizes representations motivated by phonological theory with grammatical inference algorithms that come with theoretical guarantees.

PRELIMINARIES 3

Let Σ denote a finite input alphabet of symbols and Σ^* the set of all strings over Σ of finite length. Such strings are called Σ-strings. The alphabetic symbols can represent either phonological segments, phonological feature values, or anything else that can be sequentially ordered. If w is a string, then w_i indicates the ith symbol in w and $|w|$ signifies the length of the string. λ is the empty string and $|\lambda| = 0$. For nonempty strings w, it follows that $w = w_1 w_2 \dots w_n$ with $n = |w|$. The *prefixes* of w are defined as follows: $\text{Pref}(w) = \{u \in \Sigma^* : uv = w, v \in \Sigma^*\}$. Similarly, the *suffixes* of w are $\text{Suff}(w) = \{v \in \Sigma^* : uv = w, u \in \Sigma^*\}$. Also, $\text{Suff}(w)^{k-1}$ ($\text{Pref}(w)^{k-1}$) is a suffix (prefix) of length $k-1$ for some $k \in \mathbb{N}$.

Given Σ-strings u and v, their concatenation is denoted as uv. If $w = uv$, $u^{-1}w = v$. Given two alphabets Σ, Δ, a Σ-string u, and Δ-string v of the same length, their direct product is a $\Sigma \times \Delta$-string $u \times v = (u_1, v_1)(u_2, v_2)\dots(u_n, v_n)$.

DEFINITION 1 *The longest common prefix (lcp) of a set of strings S, where $S \subseteq \Sigma^*$, is defined as follows:*

$$lcp(S) = x \text{ iff } x \in \bigcap_{w \in S} \text{Pref}(w), \forall x' \Big(x' \in \bigcap_{w \in S} \text{Pref}(w) \implies |x'| \leq |x| \Big)$$

Let Δ denote an output alphabet. Given a relation $t : \Sigma^* \times \Delta^*$, for all $w \in \Sigma^*$ and for all $v \in \Delta^*$, if $(w, v), (w, v') \in t$ implies $v = v'$, then t is a *function* and we write $t(w) = v$. A function is *total* iff for all $w \in \Sigma^*$ there exists $v \in \Delta^*$ s.t. $t(w) = v$.

For any function $f : \Sigma^* \to \Delta^*$ and $w \in \Sigma^*$ we define the *tails* of x with respect to f as $\text{tails}_f(x) = \{(y, v) : f(xy) = uv,\ u = lcp(f(x\Sigma^*))\}$. Informally, the tails of x with respect to f is the function identified with the input/output pairs (y, v) that extend from the input x and the lcp of all of possible outputs from input strings beginning with x $(x, lcp(f(x\Sigma^*))$. Two strings x and x' are *tail-equivalent* with respect to a function f iff $\text{tails}_f(x) = \text{tails}_f(x')$. We also denote it as follows: $x \sim_f x'$, where $\sim_f$ is the equivalence relation induced by tails_f which partitions Σ^*. A *subsequential function* is a function f that $\sim_f$ Σ^* into finitely many blocks (Choffrut 1977, 2003; Oncina and Garcia 1991).

3.1 Subsequential Finite-State Transducer

There are different ways to represent finite-state transducers that define subsequential functions (Oncina *et al.* 1993; Mohri 1997; Choffrut 2003). We use the delimited onward finite-state transducer (Jardine *et al.* 2014) because it associates every output with a transition, which simplifies presentation and analysis of the SOSFIA learning algorithm. Formally, onwardness relates the states in the transducer to the blocks induced by the tails equivalence relation. Informally, it ensures that outputs are produced as early as possible.

DEFINITION 2 *A delimited onward subsequential finite-state transducer (DFT) is a tuple* $(Q, \Sigma, \Delta, q_0, q_f, \delta)$ *where* Q *is a finite set of states;* Σ *and* Δ *are input and output alphabets, respectively;* $q_0 \in Q$ *is the* initial *state;* $q_f \in Q$ *is the* final state*; the transition relation is* $\delta \in Q \times (\Sigma \cup \{\rtimes, \ltimes\}) \times \Delta^* \times Q$*;* $\rtimes$ *and* $\ltimes$ *are the left and right boundary symbols respectively, and the following holds:*

i. *if* $(q, a, v, r) \in \delta$ *then* $q \neq q_f$ *and* $r \neq q_0$ *(there are no outgoing transitions from the final state nor any transitions incoming to the initial state),*

ii. *if* $(q, a, v, q_f) \in \delta$ *then* $a = \ltimes$ *and* $q \neq q_0$ *(all transitions to the final state have as input the right word boundary and do not originate in the initial state),*

iii. *if* $(q_0, a, v, r) \in \delta$ *then* $a = \rtimes$ *(all transitions out of the initial state have as input the left word boundary)*

iv. *if* $(q, \rtimes, v, r) \in \delta$ *then* $q = q_0$ *(all transitions which have as input the left word boundary originate in the initial state),*

v. *if* $((q, a, v, r), (q, a, u, s)) \in \delta$ *then* $(v = u)$ *and* $(r = s)$ *(determinism),*

vi. $(\forall q \in (Q \setminus q_0))[lcp\{v \in \Sigma^* | (\exists a \in \Sigma, r \in Q)[(q, a, v, r) \in \delta]\} = \lambda]$ *(onwardness).*

The function a DFT τ *recognizes can be defined by establishing a recurrence between input strings, states, and output strings. Informally, the idea is that given a state* q *and with an output string* y *already written, we can process an input string* x *letter by letter until its end, and return an*

updated output string y'. *Formally, this function has type* $\pi : Q \times \Delta^* \times \Sigma^* \to \Delta^*$ *and is defined as follows.*

$$\begin{aligned} \pi(q, y, \lambda) &= y \\ \pi(q, y, ax) &= \pi(q', yv, x) \textit{ provided } (q, a, v, q') \in \delta \end{aligned}$$

Finally, the function defined by τ *is* $\{(x, y) \in \Sigma^* \times \Delta^* \mid \pi(q_0, \lambda, \rtimes x \ltimes) = y\}$. *We write* $\tau(x) = y$ *iff* $(x, y) \in \tau$.

For additional details regarding these DFTs see Jardine *et al.* (2014).

Input Strictly Local (ISL) Functions 3.2

One class of subsequential functions that we focus on in this paper are Input Strictly k-Local (k-ISL) functions defined below in Definition 3 (Chandlee *et al.* 2014).

DEFINITION 3 *A function* f *is ISL iff there is a* $k \in \mathbb{N}$ *such that for all* $u_1, u_2 \in \Sigma^*$ *if* $\text{Suff}^{k-1}(u_1) = \text{Suff}^{k-1}(u_2)$ *then* $tails_f(u_1) = tails_f(u_2)$. *A function is* k*-ISL if it is ISL for some* k.

We limit our attention to total k-ISL functions. Informally, for every total ISL function f, there is a k marking the length of the widest substring required to define f, and there is a DFT representing f which is structured in a particular way: the current state of the machine is always determined by the previous $k-1$ symbols read in the input. Such DFTs are called k-ISL DFTs. Chandlee *et al.* (2014) provide an automata-theoretic characterization of ISL functions, Lambert and Heinz (2023) characterize them algebraically, and their relevance to phonology is discussed in Chandlee and Heinz 2018 and Chandlee *et al.* 2018.

DEFINITION 4 *A total* k*-ISL DFT can be constructed as follows:*

- $Q = \Sigma^{\leq k-1} \cup \{q_0, q_f\}$,
- $\exists v \in \Delta^*$ *s. t.* $(q_0, \rtimes, v, \lambda) \in \delta$.
- $\forall q \in Q \backslash \{q_0, q_f\}$, $\exists v \in \Delta^*$ *s. t.* $(q, \ltimes, v, q_f) \in \delta$.
- $\forall q \in Q \backslash \{q_0, q_f\}, \forall a \in \Sigma, \exists v \in \Delta^*$ *s. t.* $(q, a, v, \text{Suff}^{k-1}(qa)) \in \delta$.

Note that this construction may not be the smallest onward DFT recognizing a k-ISL function. An important insight resulting from this

construction of a k-ISL DFT is the dependency of $|Q|$ on the parameter k and the cardinality of Σ. In other words, the larger the k or the larger the alphabet Σ, the larger the state space. The cardinality of Q can be calculated as follows: $|Q| = \sum_{n=0}^{k-1} |\Sigma|^n + 2$.[3]

3.3 *Example: English Vowel Nasalization*

Let us consider a concrete example of a 2-ISL function, such as English vowel nasalization. In this process vowels are nasalized when followed by nasal consonants. Solé (1995), Chen (1997), and Proctor *et al.* (2013) show that nasalization is systematically conditioned and that listeners perceive nasalized vowels as distinct from their oral counterparts. Generative analyses of English phonology typically assume that vowels are underlyingly oral and undergo regressive nasalization when followed by tautosyllabic nasal consonants (Selkirk 1972; Hayes 2009). The traditional view, advocated by Krakow and Huffman (1989) and formalized by Krämer (2015), treats the alternation as a predictable, non-contrastive feature-spreading rule. We adopt this view in our study.[4] For the purpose of this study we do not introduce syllable boundaries to the data and we represent the process as follows:

(1) Vowel Nasalization (VN)
$[-\text{cons}] \longrightarrow [+\text{nasal}] \; / \; \_\_\_ \left[\begin{smallmatrix} +\text{cons} \\ +\text{nasal} \end{smallmatrix} \right]$

We now turn to explaining how this process can be modeled with a 2-ISL DFT, using a toy alphabet $\Sigma = \{a, m, t\}$. With this alphabet, the cardinality of Q is 6. A 2-ISL DFT representing a full inventory of English phonemes will quickly grow in size as every phoneme needs to be represented in a separate state. Figure 1 presents a 2-ISL DFT

[3] The number 2 includes the initial state q_0 and the final state q_f.

[4] Alternative analyses exist. In Optimality Theory (Prince and Smolensky 1993) under the principle of Richness of the Base (Kager 1999), the underlying form may be considered already nasalized. Some studies have argued that nasalization arises from gestural overlap between the articulatory movements of a vowel and a following nasal consonant (Cohn 1993; Krakow 1993; Fowler and Brown 2000), in which case, nasality can be interpreted as an automatic product of coarticulation and does not require reference to phonological processes per se.

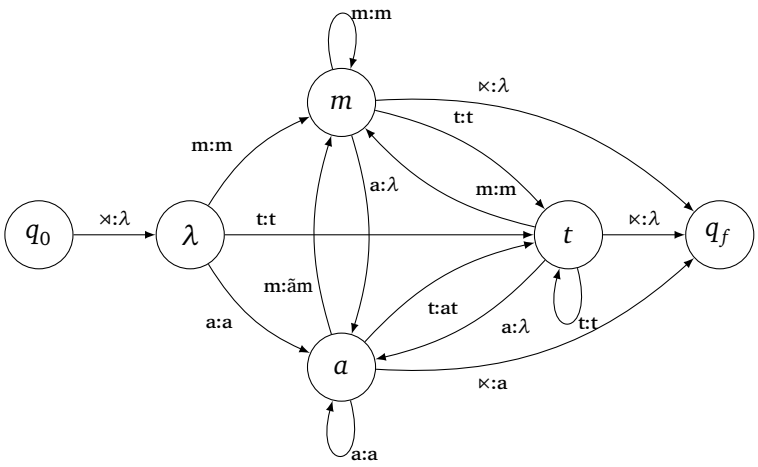

Figure 1: A 2-ISL DFT modeling vowel nasalization in English for $\Sigma = \{a, m, t\}$ and $\Delta = \Sigma \cup \{\tilde{a}\}$

for $|\Sigma| = 3$. For better readability, the transition from λ to the final state (q_f), the empty string transition, is omitted. The outputs on the transitions to and from the initial and final states are specified to be the empty string. The outputs on the transitions from the λ state to the m, a, and t states are m, λ, and t, respectively.

Consider how the DFT depicted in Figure 1 maps /mat/ to [mat] (as shown in Table 2) and /tam/ to [tãm] (as shown in Table 3).

Input:		⋊		m		a		t		⋉	
States:	q_0	→	λ	→	m	→	a	→	t	→	q_f
Output:		λ		m		λ		at		λ	

Table 2: How the DFT in Figure 1 maps /mat/ to [mat]

Input:		⋊		t		a		m		⋉	
States:	q_0	→	λ	→	t	→	a	→	m	→	q_f
Output:		λ		t		λ		ãm		λ	

Table 3: How the DFT in Figure 1 maps /tam/ to [tãm]

Because nasalization is conditioned by a right context, the transitions into the 'a' state from a distinct state outputs λ because at this point, it is unknown whether the following segment will be a nasal consonant or not so it is unknown whether to output 'a' or 'ã.' Once the segment following 'a' is read, the output produces either 'a,' 'at,' or 'ãm' appropriately. With the growth of the alphabet, it would be

observed that all vowel states behave in a similar manner, a generalization that *can* be captured with featural but not segmental representations. The following section provides details on how we incorporate phonological features into the learning process.

3.4 *Features*

In this work, we consider a feature ϕ to be a function from an alphabet Σ to the set of values $V = \{+, -, 0\}$, which indicate positive, negative, or a null value for the feature respectively. As an example, consider the feature system for English in Table 4, adapted from Hayes (2009, pp. 70–99), which specifies several features for the phonemes of English.[5]

Each feature ϕ can be lifted to a homomorphism from Σ^* to V^* in the natural way. For instance, $[\text{cor}](mat) = [\,-\,0\,+\,]$ and $[\text{cons}](mat) = [\,+\,-\,+\,]$. We call $\phi(x)$ the projection of x to the feature ϕ.

Given an ordered list of features $\Phi = (\phi_1, \phi_2, \ldots \phi_n)$, we interpret Φ as the direct product of the homomorphisms of its component parts. For instance, $[\,\begin{smallmatrix}\text{cor}\\ \text{cons}\end{smallmatrix}\,](mat) = \left[\begin{smallmatrix}-&0&+\\ +&-&+\end{smallmatrix}\right]$. As is common in the phonological literature, we write these strings of feature sets with square brackets instead of the more cumbersome $(-,+)(0,-)(+,+)$. Like before, we say $\Phi(x)$ is the projection of x to the features in Φ. Note that the domain of Φ is Σ^* and its co-domain is $(V^n)^*$.

Finally, given a sample S of input/output pairs and two sets of features Φ, Φ', let $\langle \Phi, \Phi' \rangle(S) = \{(\Phi(x), \Phi'(y)) : (x, y) \in S\}$. In other words, for any pair of strings (x, y) we can project the x string to a set of feature values from Φ and the y string to a set of feature values from Φ'.

Given a feature system F, we observe that there are maximally $|V|^{|F|}$ distinct representations (see Reiss 2012 for discussion). This is the *maximal size of the featural alphabet*. Throughout the remainder of this paper, we use Σ_Φ to denote the featural alphabet and Σ_{seg} to denote the segmental alphabet.

[5] Because vowels require fewer features to be fully specified, any feature that Hayes (2009) does not use for vowels is assigned the value 0 in our system, so the feature [coronal] for a vowel is represented as [0 coronal]. To keep the notation concise, we abbreviate feature names to their first few letters, so [cons] stands for [consonantal] and [cor] stands for [coronal].

Table 4: Feature chart for English phonemes

Segment	æ	ɑ	ɔ	ɪ	ɛ	ʌ	ʊ	u	i	m	p	b	f	v	θ	ð	n	t	s	z	l	t͡ʃ	d͡ʒ	ŋ	k	g
high	−	−	−	+	−	−	+	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	+
low	+	+	−	−	−	−	−	−	−	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−	−	−
tense	0	0	−	−	−	−	−	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
front	+	−	−	+	+	−	−	−	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
back	−	+	+	−	−	+	+	+	−	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
round	−	−	+	−	−	−	+	+	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
long	−	−	−	−	−	−	−	−	−	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cons	−	−	−	−	−	−	−	−	−	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
son	+	+	+	+	+	+	+	+	+	+	−	−	−	−	−	−	+	−	−	−	+	−	−	+	−	−
cont	+	+	+	+	+	+	+	+	+	−	−	−	+	+	+	+	+	−	+	+	+	−	−	−	−	−
delrel	0	0	0	0	0	0	0	0	0	0	−	−	+	+	+	+	0	−	+	+	0	+	+	0	−	−
approx	0	0	0	0	0	0	0	0	0	−	−	−	−	−	−	−	+	−	−	−	+	−	−	−	−	−
nasal	−	−	−	−	−	−	−	−	−	+	−	−	−	−	−	−	+	−	−	−	−	−	−	+	−	−
voice	+	+	+	+	+	+	+	+	+	+	−	+	−	+	−	+	+	−	−	+	+	−	+	+	−	+
lab	0	0	0	0	0	0	0	0	0	+	+	+	+	+	−	−	−	−	−	−	−	−	−	−	−	−
labdent	0	0	0	0	0	0	0	0	0	−	−	−	+	+	−	−	−	−	−	−	−	−	−	−	−	−
cor	0	0	0	0	0	0	0	0	0	−	−	−	−	−	+	+	+	+	+	+	+	+	+	−	−	−
ant	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	+	+	+	+	+	−	−	0	0	0
distr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	−	−	−	−	−	+	+	0	0	0
strid	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−	−	−	−	+	+	−	+	+	0	0	0
lat	0	0	0	0	0	0	0	0	0	−	−	−	−	−	−	−	+	−	−	−	+	−	−	−	−	−
dor	0	0	0	0	0	0	0	0	0	−	−	−	−	−	−	−	−	−	−	−	−	−	−	+	+	+

4 DECOMPOSING FUNCTIONS WITH FEATURES

We aim to decompose the target sequential function $f : \Sigma^* \rightarrow \Delta^*$ into n-many sequential functions, one for each feature ϕ. In this work, each ϕ is individually represented with a k-ISL DFT, $\tau_{(\Phi,\phi)}$, where ϕ is the feature whose values are being output by the machine, and Φ is a set of features being used as input symbols by the machine. Put differently, each $\tau_{(\Phi,\phi)}$ can be thought of as aiming to predict a particular feature ϕ of the output using only the feature set Φ in the input. The feature combination Φ may vary across the n-many transducers.

For instance, consider the prior example of English vowel nasalization. Table 4 provides 22 features and so this process will be represented by 22 transducers, one for each feature. In this work, we assume each of these will be represented with a 2-ISL DFT, though we discuss in the Conclusion why we believe this assumption can be relaxed in future work. All but one of these features, [nasal], is always faithful to its underlying representation. Consequently, for a faithful feature ϕ, the transducer $\tau_{(\{\phi\},\phi)}$ will be the 2-ISL DFT whose input and output alphabets correspond to the feature values, and which represents the identity function. Specifically, for each transition, an input symbol not equal to $\rtimes$ or $\ltimes$ is an element of V and the output is an element of V^*. The output on the transitions with an input symbol $\sigma \in \{\rtimes, \ltimes\}$ is the empty string λ.

In English vowel nasalization, the feature [nasal] is not always faithful. Furthermore, its surface value cannot be predicted from the feature [nasal] alone. However, the features [nasal] and [consonantal] together are sufficient to determine its output value. Thus the feature [nasal] can be modeled with the 2-ISL DFT $\tau_{(\left[\begin{smallmatrix}\text{nas}\\ \text{cons}\end{smallmatrix}\right],\text{nasal})}$. The input alphabet (and thus the states) of this DFT corresponds to the set of feature value combinations of [nas, cons] in the input.

While there are 3^2 logical combinations available, in practice we only use those combinations that are present in a particular dataset. Since both of those particular features have only binary values $\{+, -\}$, this transducer could be composed, in theory, of up to 2^2 states, i.e. $\{\left[\begin{smallmatrix}+\\ +\end{smallmatrix}\right], \left[\begin{smallmatrix}-\\ +\end{smallmatrix}\right], \left[\begin{smallmatrix}-\\ -\end{smallmatrix}\right], \left[\begin{smallmatrix}+\\ -\end{smallmatrix}\right]\}$, where the upper symbol in each pair represents values for feature [nasal], and the lower one represents values for

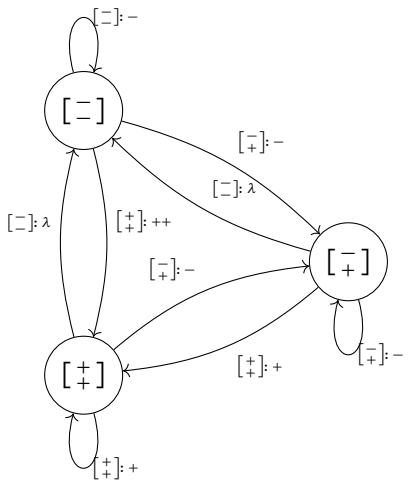

Figure 2:
A 2-ISL DFT$_{([\,^{\text{nasal}}_{\text{cons}}\,],\text{nasal})}$
for English vowel nasalization

[consonantal]. As such, the state $[\,^{+}_{+}\,]$ corresponds to a nasal consonant, $[\,^{-}_{+}\,]$ corresponds to an oral consonant, and so on. However, in this analysis of English vowel nasalization, there are no underlying nasal vowels; hence the state $[\,^{+}_{-}\,]$ will be omitted from this transducer.

Figure 2 represents $\tau_{([\,^{\text{nas}}_{\text{cons}}\,],\text{nasal})}$ for vowel nasalization in English with the initial, final and lambda states omitted for better readability. Also, note the difference between the input and output notations: the square brackets around the inputs indicate a collection of values for a single segment. The outputs are strings of values for the feature [nasal] only. For example, $[\,^{+}_{+}\,] : ++$ means that a $[\,^{+\ \text{nas}}_{+\ \text{cons}}\,]$ symbol is being read and two [+nasal] symbols are being output.

This example may appear to suggest that the feature-based model of English nasalization is more complex than the segment-based model. The feature-based model consists of 22 DFTs, whereas the segment-based model contains a single DFT. Furthermore, as each of the DFTs is a 2-ISL DFT with $|\Sigma_\Phi| \leq 3$, they roughly have the same number of states. Thus the total number of states in the feature-based model is roughly 22 times that of the segment-based model. However, it is important to notice that the segment-based DFT in Figure 1 is built only for 3 segments: {a, m, t}. The crucial way the transducer operating over segments differs from the one operating over features is that as the size of the segmental inventory grows, the number of states will increase in the segment-based model, but not in the feature-based model. This is the particular reason why learning sequential functions

over features requires less data. This point is discussed in greater detail in the next section.

Finally, we need to explain precisely how the feature-based model with its n-many DFTs (one for each $\phi \in F$) produces an output from an input like /tam/. Recall the notation $\tau_{(\Phi,\phi)}$ for each of these DFTs, where Φ is the set of features determining the input alphabet and ϕ is the feature whose values are being output. Essentially, an input string x is projected according to Φ and then processed by $\tau_{(\Phi,\phi)}$, producing a string of feature values for ϕ. This happens for each $\tau_{(\Phi,\phi)}$ in the feature-based model, and thus n-many strings of feature values are produced. The feature values across strings are then combined by the direct product operation in order to recover the segmental units. This workflow is schematized in Figure 3.

5 WHY FEATURAL DECOMPOSITION SHOULD FACILITATE LEARNING

This section argues on theoretical grounds that learning the feature-based representations will require less data. For concreteness, we make the argument with k-ISL DFT transducers.

Jardine *et al.* (2014) prove that the size of a characteristic sample for a target DFT τ is $O(|\tau|)$ (Lemma 18). In other words, the size of the characteristic sample is a linear function of the size of the transducer. The size of a DFT τ is measured by the number of its states and the lengths of its outputs on its transitions (Definition 5).

DEFINITION 5 *The size of a DFT* $\tau = (Q, \Sigma, \Delta, q_0, q_f, \delta)$ *is* $|Q| + \sum_{(q,a,v,r)\in\delta} |v|$.

Following Definition 4, the number of states in a k-ISL DFT is simply the cardinality of $\Sigma^{\leq k-1} + 2$. Because the size of a k-ISL DFT transducer is dependent on $|\Sigma|$ and k, reducing one of these variables can reduce the size of the DFT. Table 5 presents an example of how the state space grows in terms of $|\Sigma|$ and k. As k increases, the number of states increases exponentially. For $k = 3$, if the alphabet consists of 5 segments, the number of states will grow to 33, while for $|\Sigma| = 10$, the number of states will increase to 113.

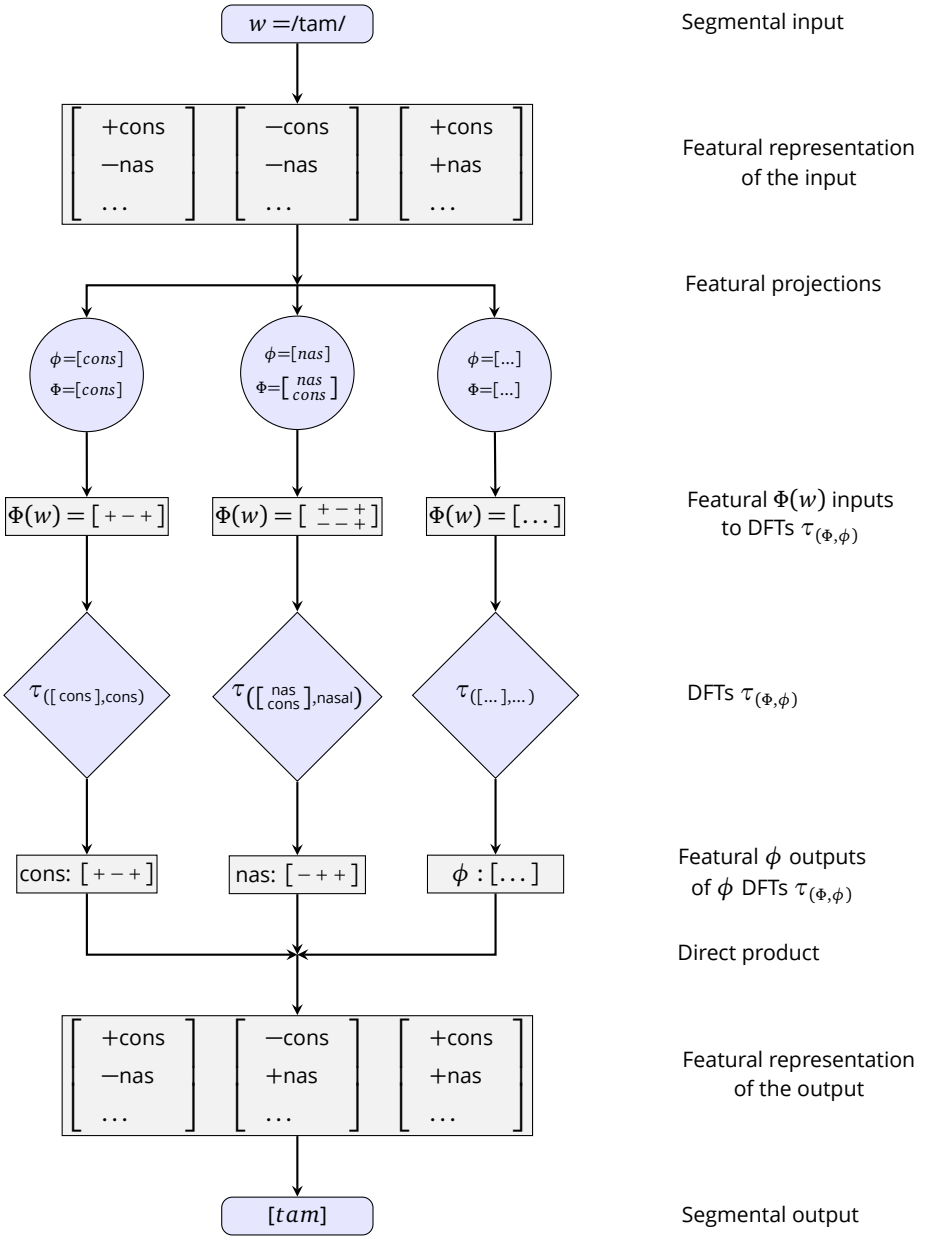

Figure 3: The generation process by which a feature-based model generates outputs from inputs

Table 5: The number of states in k-ISL DFTs

$\lvert\Sigma\rvert$	$k=2$	$k=3$	$k=4$
5	8	33	158
10	13	113	1,113
25	28	153	3,278
50	53	2,553	127,553
100	103	10,103	1,010,103

What do these numbers mean for segment-based models vis-à-vis feature-based models? In a segment-based model, Σ is the alphabet of segments. In a feature-based model, Σ is the number of feature value combinations used in a particular feature machine.

Consider the example discussed earlier for English vowel nasalization. For concreteness, assume a variety of English with 40 phonemes. It follows that the segmental model would consist of a single 2-ISL DFT with 43 states. Now consider the featural model. This consists of 22 2-ISL DFTs. For faithful features like [consonantal], there are only 2 feature values $\{+,-\}$, which constitute the input alphabet. This [consonantal] DFT would have 5 states. For a vowel feature like [high], there are 3 feature values $\{+,-,0\}$. Thus, the [high] DFT has 6 states. For the [nasal] DFT depicted in Figure 2, there are also 6 states, because only three combinations of the features [nasal, consonant] are present on the input side in any data sample (recall that the combination $[\,{}^{+}_{-}\,]$ is omitted since the underlying vowels are not nasal).

As mentioned, the size of a characteristic sample to learn these DFTs is linear in the size of the DFT. Each of the feature DFTs is a fraction of the size of the segmental DFT, and it follows that the sample needed to generalize correctly is correspondingly a fraction of the size of the characteristic sample to learn the segmental DFT. In other words, a smaller machine results in a smaller characteristic sample. Section 7 provides empirical evidence supporting this hypothesis.

There are some caveats worth discussing, which helps motivate the empirical analysis in Section 7. The total size of the feature-based model is the sum of the sizes of the 22 DFTs. If each DFT had six states, the total size of the feature-based model would thus be 132, almost three times the size of the segmental model. It is natural to wonder whether the feature-based model is really a smaller model.

There are two reasons why the comparison in the above example can be misleading. First, this example is particularly simple in order to help explain the basic concepts. Processes with larger k windows require much larger DFTs; in fact, the number of states grows exponentially with respect to k. For instance, suppose that there is a language with 50 phonemes described with 20 binary features (so each $\phi \in F$ has only 2 values) and consider a 3-ISL process where each feature value can be predicted from a combination of two binary features (so $|\Sigma_\Phi| = 4$). A segment-based model needs 2,553 states (Table 5). Each DFT in the feature model would have at most $20 + 3$ states, and thus the entire model contains $20 \times 23 = 460$ states. In other words, as the k value increases, feature-based models can be significantly more compact.

One potential objection to this line of reasoning is that as the number of feature combinations grows, the size of Σ in the feature-based model can grow as well. For example, if there was one feature value that required 10 binary features to predict, that could potentially mean an alphabet of size 2^{10}. This objection is valid to some extent. We acknowledge that in the worst case, the feature-based model may be no better than the segment-based model, and could even be worse. However, we also do not anticipate that the worst case appears very often. For example, we are not aware of any phonological processes that would require 10 features interacting to predict the value of a feature. Feature systems are also generally 'sparse' in the sense that one can usually find fewer than 10 features to uniquely pick out a phoneme. In other words, the specter of the worst case may be more theoretical than found in actual practice.

The second reason for comparing the total size of the segmental model to the feature-based model is that the size of the characteristic sample for the feature-based model may not be the sum of the sizes of the characteristic samples of the individual DFTs that make it up. Those characteristic samples may overlap quite significantly.

In short, the questions raised by this theoretical analysis motivate the empirical investigation in the next section. We believe the results there vindicate the approach advocated here.

6 THE ALGORITHMS

This section describes three algorithms central to this study. First, we discuss the SOSFIA algorithm introduced by Jardine *et al.* (2014). Next we present the Factor-by-Feature (FxF) algorithm which uses SOSFIA. Once again, we emphasize that the FxF algorithm is a broader concept that can work with *any* inference algorithm. Third, we present an algorithm that selects *a near-minimal sample* from the characteristic sample for SOSFIA. We call this algorithm A Minimal Sample Search Algorithm (AMβA).

6.1 *SOSFIA*

SOSFIA (see Algorithm 1) takes as its arguments a finite sample of input-output pairs $S \subset \Sigma^* \times \Delta^*$, and an output-empty DFT $\tau_\square$, i.e. a transducer where for every $(q, a, v, r) \in \delta$, $v = \square$, where $\square$ represents a 'blank' that needs to be filled in. Each output-empty DFT corresponds

Algorithm 1: Structured Onward Subsequential Function Inference Algorithm (SOSFIA, Jardine *et al.* 2014)

Data: A sample $S \subset \rtimes\Sigma^* \ltimes \times\Delta^*$, an output-empty DFT $\tau_\square = (Q, \Sigma, \Delta = \{\square\}, q_0, q_f, \delta)$
Result: $\tau_\square$ as a DFT with filled transitions

```
F ← empty_Queue
Push (F, (q_0, λ))
mark (q_0)
while F ≠ ∅ do
    (q, w) ← Shift_First(F)
    for all σ ∈ Σ ∪ {⋊, ⋉} (in lexicographic order) do
        for all δ_i = (q, σ, □, q′) ∈ δ do
            if ∃ σ′ ≠ σ such that (q, σ′, u, q″) ∈ δ then
                Change δ_i to (q, σ, v, r), where v = min_change_S(w, σ)
            else
                Change δ_i to (q, σ, λ, r)
            end if
            if q′ is not marked then
                Push (F, (q′, wσ))
            end if
        end for
    end for
end while
```

to a class of sequential functions which vary only in what strings are assigned to the blanks. Many important subregular classes of functions can in fact be represented by such a DFT, such as the k-ISL functions. Jardine *et al.* (2014) prove that the algorithm successfully infers the class of sequential functions represented by $\tau_{\square}$ given a sufficient data sample.

SOSFIA conducts a *breadth-first* search through the output-empty DFT considering the shortest path that led to that state. In the case of the k-ISL DFTs, the shortest input prefix that leads to the 'ma' state is '⋊ma'. SOSFIA stores in a queue untreated states together with their shortest prefixes. The next step identifies the outputs on the transitions using two functions `common_out` and `min_change`, defined below.

DEFINITION 6 *The* `common_out` *of an input prefix w in a training sample S is the longest common prefix (lcp) of all the strings that start with w in S:*

$$\texttt{common\_out}_S(w) = lcp(\{u \in \Sigma^* : \exists v \; s.t. \; (wv, u) \in S\}).$$

DEFINITION 7 *The* `min_change` *from a string w to a string $w\sigma$ in a training sample S:*

$$\texttt{min\_change}_S(\sigma, w) = \begin{cases} \texttt{common\_out}_S(\sigma) & \text{if } w = \lambda \\ \texttt{common\_out}_S(w)^{-1}\texttt{common\_out}_S(w\sigma) & \text{otherwise} \end{cases}$$

Given a transition $(q, a, \square, r)$, SOSFIA calculates the lcp of all the outputs associated with the input strings beginning with the shortest prefix to q, as well as the lcp of all the outputs associated with the input strings beginning with the shortest prefix to r by way of q and a. Then the minimum change between `common_out`(q) and `common_out`(r) is determined and replaces the blank square in the $(q, a, \square, r)$ transition. The procedure is repeated until the queue tracking untreated states is empty.

6.2 *Factor-by-Feature (FxF)*

The FxF algorithm decomposes sequential data into subsegmental units (features) and learns functions for those individual units

separately. Given a set of features F of cardinality n, a sample S (input-output pairs with segmental representations), and a k value, the FxF algorithm iterates through the features $\phi \in F$ and for each ϕ it finds a set of features Φ such that the k-ISL DFT $\tau_{(\Phi,\phi)}$ accurately predicts those feature values in the outputs of the sample. The final output of FxF is a grammar G containing n-many $\tau_{\square(\Phi,\phi)}$, one for each $\phi \in F$. The pseudocode is presented in Algorithm 2.

Algorithm 2: Factor-by-Feature (FxF)

Data: A sample $S \subset \rtimes\Sigma^* \ltimes \times\Delta^*$, a feature set F, and a non-negative integer k
Result: A set G containing n-many k-ISL DFTs, one for each $\phi \in F$
$G \leftarrow \emptyset$
for all $\phi \in F$ (in lexicographic order) **do**
$\tau_{(\Phi,\phi)} \leftarrow \texttt{FeatSearch}(k, S, F, \phi, [\{\phi\}])$
$G \leftarrow G \cup \{\tau_{(\Phi,\phi)}\}$
end for
return G

As discussed earlier, it is often the case that information from more than one feature is required to determine the output feature values of a feature ϕ. `FeatSearch` (Algorithm 3) searches for a featural combination which makes accurate predictions for ϕ. It takes as arguments a non-negative integer k, the sample S, the feature system F, the target feature ϕ and a list of feature sets `FeatLIST`, which is initialized with one item in the list: the singleton feature set $\{\phi\}$.

The algorithm `FeatSearch` dequeues the first feature combination Φ from `FeatLIST` and evaluates it. It checks if the projected sample $S_{(\Phi,\phi)}$ is functional using the `is_functional` function (Definition 8). If it is, a new k-ISL DFT $\tau_{(\Phi,\phi)}$ is constructed. Definition 4 specifies the structure of a k-ISL DFT $\tau_{(\Phi,\phi)}$. At this step, we initialize an output-empty $\tau_{\square(\Phi,\phi)}$, i.e. one whose output on each transition is empty. More formally, $\forall(q, a, v, r) \in \delta^*[v = \square]$. The newly constructed DFT and $S_{(\Phi,\phi)}$ are then passed as arguments to SOSFIA and `FeatSearch` returns the output of SOSFIA.

DEFINITION 8 *A data sample S is functional under the projection functions Φ and Φ' iff for all $(u, v), (u', v') \in S$, if $\Phi(u) = \Phi'(u')$ then $\Phi(v) = \Phi'(v')$.*

Algorithm 3: FeatSearch

Data: A nonnegative integer k, a sample $S \subset \rtimes\Sigma^* \ltimes \times\Delta^*$, a feature set F, a feature ϕ, and a list of sets of features FeatLIST
Result: A DFT $\tau_{(\Phi,\phi)}$

```
while FeatLIST is not empty do
    Φ ← dequeue(FeatLIST)
    m ← |Φ|
    if is_functional(S_(Φ,φ)) then
        τ_□ ← build output-empty k-ISL DFT with Φ and φ
        return SOSFIA(τ_□, S_(Φ,φ))
    end if
end while
NewFeatLIST ← feat_combo(φ, m + 1, F)
return FeatSearch(k, S, F, φ, NewFeatLIST)
```

If, on the other hand, $S_{(\Phi,\phi)}$ is not functional, `FeatSearch` generates a new list of feature combinations of size $m+1$ with the `feat_combo` function (Definition 9) and recursively calls itself with this new list appended to the rear of the queue, searching for a sufficient featural combination.

DEFINITION 9 `feat_combo`(ϕ, m, F) *(Algorithm 4) produces a list of feature sets where each feature set contains* ϕ *and is of size* m*. Formally,* `feat_combo`(ϕ, m, F) *returns the set* $\{\Phi \subseteq F \mid \phi \in \Phi, |\Phi| = m\}$ *and puts its elements in a queue.*

Algorithm 4: feat_combo

Data: A feature ϕ, a positive integer m, and a feature set F
Result: A queue FeatLIST of feature sets Φ such that $\phi \in \Phi$ and $|\Phi| = m$

```
FeatLIST ← empty queue
AllCombos ← COMBINATIONS(F, m)
for all Φ ∈ AllCombos do
    if φ ∈ Φ then
        enqueue(FeatLIST, Φ)
    end if
end for
return FeatLIST
```

Algorithm `feat_combo` generates candidate feature sets by enumerating `COMBINATIONS`(F, m), i.e., the collection of all m-element subsets of F (combinations without repetition, where order is irrelevant), and retaining only those Φ such that $\phi \in \Phi$. In this way it is

similar to the BUFIA algorithm (Chandlee *et al.* 2019). It is this subroutine that also makes FxF run in time exponential in $|F|$, the number of available features, in the worst case. This is because if all features in F are necessary to predict some feature ϕ then `feat_combo` will have to traverse as many subsets as there are in the powerset of $F/\{\phi\}$ before reaching F itself.[6]

However, in the feature system for English, the maximum number of features needed to distinguish each phoneme from every other phoneme is far less than the cardinality of F. BUFIA also has a run time that is exponential in the worst case, but it has proven effective in practice (Swanson *et al.* to appear; Li 2025). For these reasons, we do not let the worst-case, exponential time analysis deter our investigation into the practical utility of FxF.

6.3 *Learning vowel nasalization over features: an example*

We illustrate FxF(SOSFIA) with the vowel nasalization example. Consider the sample S with pairs (mæn, mæ̃n) and (mææ, mææ̃). Let F be the feature system in Table 4 and $k = 2$. Then FxF(SOSFIA) initializes an empty grammar G, and considers the features in F one by one.

Suppose the first feature it considers is ϕ = [nasal]. It proceeds to run `FeatSearch` with FeatLIST initialized to $[\{\phi\}]$. It dequeues FeatLIST, setting $\Phi = \{\phi\}$ and $m = 1$. It then calculates the projected sample $S_{([\text{nasal}],\text{nasal})}$ to be ([+−+], [+++]) and ([+−+], [+−+]). As such, `is_functional` outputs False, and `feat_combo` generates all feature sets of size 2, which include [nasal], and places them on NewFeatLIST. `FeatSearch` calls itself again with NewFeatList.

On this next round, the queue FeatLIST contains more than one feature set, and it checks them one at a time. For example, if Φ

[6] The other aspects of FxF run in polynomial time. Finding the featural projection of the data is linear in the size of the data. Checking the functionality of a projected sample can be done in quadratic time by checking pairs of data points. Literally building the k-ISL DFTs is a polynomial of degree k for a given featural alphabet $|\Sigma_\Phi|$. However, for finitely many featural alphabets (for instance (V^i) for $1 \leq i \leq n$) these k-ISL DFTs could be stored in advance and just retrieved from memory at the cost of a (large) constant. Finally, SOSFIA itself runs in time linear in the data.

= [nasal, voice], then `is_functional` would again return False since the projected sample $S_{(\left[\begin{smallmatrix}\text{nasal}\\\text{voice}\end{smallmatrix}\right],\text{nasal})}$ is not functional. This is because [voice] does not provide sufficient information to predict the right outputs for [nasal]. Particularly, it does not separate vowels from consonants. Therefore, Φ would be re-assigned to the next element in the queue, and the process would continue.

Eventually, `FeatSearch` dequeues Φ = [nasal, cons]. The projected sample $S_{(\left[\begin{smallmatrix}\text{nasal}\\\text{cons}\end{smallmatrix}\right],\text{nasal})}$ is functional. Consequently, `FeatSearch` constructs an output-empty, k-ISL DFT $\tau_{\square(\left[\begin{smallmatrix}\text{nasal}\\\text{cons}\end{smallmatrix}\right],\text{nasal})}$ and passes it with $S_{(\left[\begin{smallmatrix}\text{nasal}\\\text{cons}\end{smallmatrix}\right],\text{nasal})}$ to SOSFIA, returns the output, and terminates.

Interestingly, any feature combination which distinguishes consonants from vowels is sufficient to make the right predictions for the feature [nasal]. For example, the feature [labiodental] assigns a value 0 to all vowels and either + or − to consonants. Consequently, the projected sample $S_{(\left[\begin{smallmatrix}\text{nasal}\\\text{labdent}\end{smallmatrix}\right],\text{nasal})}$ is functional and would also lead to SOSFIA outputting a transducer. `FeatSearch` terminates with the first feature combination that works. Thus, the order in which the feature sets Φ are considered does matter, though we abstract away from this choice in this description of the algorithm. We return to this point in the experimental and discussion sections later.

Finally, if none of the feature sets containing [nasal] of length $|\Phi|$ result in `is_functional` outputting True, the length m is increased by one, and the procedure repeats. With the feature [nasal] completed, FxF(SOSFIA) moves on to the next feature, and continues to do so until transducers are obtained for each feature.[7]

To summarize, FxF(SOSFIA) factors the problem of learning a phonological process along featural dimensions. For each feature it searches successively more complex feature combinations until it finds one that generalizes successfully given the other constraints on learning (such as the k-ISL structure). While in the worst case, the problem reaches a stage where it is essentially learning over segmental representations, the idea is that, in practice, the decomposition to features allows correct generalizations to be found more quickly with less data. To establish this point empirically requires a way of measuring the

[7] Alternatively, FxF could be run in parallel for each feature $\phi \in F$.

amount of data that is sufficient for FxF(SOSFIA) to succeed and to compare it to the amount of data that is sufficient for SOSFIA over segmental representations to succeed. That is the purpose of the next algorithm.

6.4 *Characteristic sample and a minimal sample search algorithm (AMβA)*

This section describes SOSFIA's characteristic sample (henceforth, CS) that guarantees success in learning k-ISL functions and provides a simple heuristic method for finding a near-minimal characteristic sample that is a subset of the CS provided by Jardine *et al.* (2014). This heuristic will be used as a measuring stick in the experimental case studies when investigating how much data is sufficient for a successful generalization.

Jardine *et al.* (2014) present formal concepts for a class of functions $\mathbb{T}$, a class of representations (grammars) $\mathbb{R}$, a learning algorithm, and a characteristic sample.

DEFINITION 10 *Let $\mathbb{T}$ be a class of functions represented by some class of representations $\mathbb{R}$. A sample S for a function $t \in \mathbb{T}$ is a finite set of data for which $(w, v) \in S$ iff $t(w) = v$. A $(\mathbb{T}, \mathbb{R})$-learning algorithm $\mathscr{A}$ is a program that takes a sample for a function $t \in T$ and outputs a representation from $\mathbb{R}$.*

DEFINITION 11 *Let $\mathscr{L}$ be a naming, total function that maps $\mathbb{R}$ onto $\mathbb{T}$. For a $(\mathbb{T}, \mathbb{R})$-learning algorithm $\mathscr{A}$, a sample CS is a characteristic sample of a representation $r \in \mathbb{R}$ if for all samples S for $\mathscr{L}(r)$ s.t. $CS \subseteq S$, $\mathscr{A}$ returns r.*

In this context, k-ISL functions are represented with k-ISL DFTs, and SOSFIA outputs a representation r isomorphic to the original k-ISL DFT that generated the CS. By referring to the properties of k-ISL DFTs and SOSFIA's method of inferring the underlying k-ISL functions, we show that the minimum size of the longest string $s \in \langle \Phi, \Phi' \rangle(CS) = 2k-1$.

The core idea is that a characteristic sample exposes all possible paths in the transducer to ensure the learning algorithm has sufficient information. In Lemma 16, Jardine *et al.* (2014) show that for each

transition $(q, a, v, r) \in \delta$ in a DFT, it is sufficient if a sample includes a set of strings of the form pas where these symbols are defined as follows.[8]

i p is the shortest input that leads from the start state to state q;

ii a is the input symbol on the transition;

iii s is a shortest string that leads from state r to any state in the machine.

PROPOSITION 1 *For k-ISL functions and for SOSFIA, a sample which includes all strings up to length $2k-1$ is characteristic.*

PROOF From the definition of k-ISL functions, the size of a context that can affect a change is k-long. Therefore, it is necessary to consider k-long substrings from *every* state $q \in Q \setminus \{q_f\}$ in the k-ISL DFT for the function. Let the shortest string to reach q from q_0 be p_q. We are interested in $\max\{p_q \mid q \in Q \setminus \{q_f\}\}$. The furthest non-final state q from q_0 is reachable by a string of length equal to $k-1$. In other words, the depth of the k-ISL machine (excluding q_f) is $k-1$. Hence, the sum of the depth of a k-ISL DFT ($k-1$) and the memory window (k) is equal to $2k-1$.

By Lemma 16 in Jardine *et al.* 2014, it follows that if S contains all strings up to length $2k-1$ then it is characteristic. □

Proposition 1 provides a characteristic sample for k-ISL functions, but it does not necessarily provide the smallest such sample. Next, we present a procedure for searching for a near-minimal CS: A Minimal Sample Search Algorithm (AMβA) (Algorithm 5). Since AMβA is a greedy, randomized procedure, there is no guarantee that the sample it finds is *the* minimal one. Hence, the output of AMβA is some sufficient subset of the sample in Proposition 1, which will be referred to as *near-minimal CS*.

AMβA is built on top of FxF. It takes three arguments: an original, segmental-based DFT τ representing the target process, a sample S generated with τ, and a feature set F. It selects 10 input-output pairs[9] at random from S and stores them in a variable called

[8] There is another condition to Lemma 16, but due to the properties of k-ISL DFTs, it does not apply since the structure of the k-ISL DFTs precludes it.

[9] We chose 10 as a balance between convenience and fine-grained analysis.

Algorithm 5: A Minimal Sample Search Algorithm (AMβA)

Data: A segment-based k-ISL DFT τ, a sample $S \subset \rtimes\Sigma^* \ltimes \times\Delta^*$, a feature set F
Result: A near-minimal sufficient sample

```
min_sample ← ∅
min_sample ← min_sample ∪ random(S, 10)
G ← FxF(min_sample, F, k)
while ¬match(G, τ) do
    S ← S \ min_sample
    min_sample ← min_sample ∪ random(S, 10)
    G ← FxF(min_sample, F, k)
end while
return min_sample
```

`min_sample`. The selected pairs are subsequently provided as an argument to FxF(SOSFIA), which ultimately returns a grammar G, which is a collection of k-ISL DFTs, one for each feature in F. AMβA then calls the `match` function (Algorithm 6) to contrast the output of FxF(SOSFIA) with the original τ. For each ϕ in F, the `match` function transforms the original τ by collapsing transitions and states by projecting the output labels under ϕ and by projecting the input labels and state names under the set Φ associated with ϕ in G. This method generates a collection of n distinct DFTs $\tau_{(\Phi,\phi)}$, each corresponding to a feature $\phi \in F$. This set is then evaluated against the output of FxF(SOSFIA) for the same features F. If the compared sets are identical, it means that FxF learned a representation for the target function with `min_sample` and thus AMβA returns it. If not, the procedure continues sampling without replacement until the min_sample is large enough to infer the target function, i.e. until the sample is *sufficient*.

Algorithm 6: `match`

Data: A learned DFT G, a target DFT τ
Result: Bool

```
δ_t ← transitions in δ ∈ τ
δ_l ← transitions in δ ∈ G
for all (q, σ, v, r) ∈ δ_t do
    if (q, σ, z, r) ∈ δ_l and v ≠ z then
        return False
    end if
end for
return True
```

EMPIRICAL ARGUMENTS 7

This section presents four case studies evaluating the performance of FxF(SOSFIA) and SOSFIA in learning four distinct sequential functions, each representing a phonological process attested in natural language. The processes include: vowel nasalization in English (2-ISL), vowel shortening in Yawelmani (3-ISL), schwa epenthesis in Chukchi (3-ISL), and an interaction between final devoicing and ɔ-raising in Polish (2-ISL and 3-ISL, respectively).

Generally, the objective of the case studies is to evaluate the performance of the novel algorithm, FxF(SOSFIA). In particular, they test the hypothesis that FxF(SOSFIA) requires significantly less training data compared to SOSFIA. Moreover, they also investigate the near-minimal samples produced by AMβA by comparing outputs for both algorithms against the originally generated characteristic samples (CS). Due to the inherent randomness in AMβA's minimal sample selection, each experiment was repeated ten times. The reported results include both the mean and standard deviation across the runs.

The characteristic samples were constructed using a subset of phonemes from the respective languages. Ideally, the entire phonemic inventories would be used. However, full data generation proved computationally expensive due to the size of the resulting CSs. For example, Yawelmani's inventory includes 43 segments, and modeling vowel shortening as a 3-ISL function yields a characteristic sample of size $|CS| = 150{,}508{,}643$ (see Proposition 1). To keep experiments tractable, the phonemic inventories were reduced so that $|CS|$ did not exceed 20,000. However, this practical decision was not without consequences, as discussed in later sections.

Representative k-ISL functions 7.1

This section presents five phonological processes attested across four natural languages. The processes were deliberately selected to illustrate variation in the value of k as well as in the number and type of phonological features involved in their computation. As with the case of English vowel nasalization, we acknowledge that alternative analyses and debates exist for some of the phenomena discussed. Our aim

is not to argue for any one analysis in particular but instead to use concrete examples to meet our objectives. These examples should be understood as representative instances of broader classes of processes that fall within the expressive range of k-ISL functions.

7.1.1 Vowel nasalization in English

Vowel nasalization was introduced and discussed in Section 3.3. As mentioned, the process has been expressed with a phonological rule like the one shown in (1), repeated below.

(2) Vowel Nasalization (VN)
$[-\text{cons}] \longrightarrow [+\text{nasal}] \; / \; \_\_\_ \left[\begin{smallmatrix} +\text{cons} \\ +\text{nasal} \end{smallmatrix} \right]$

By modeling vowel nasalization in this way, we test whether a learner can infer a contextually conditioned alternation that depends on the feature composition of a following segment. In particular, we expect that the feature [nasal] will not be sufficient on its own to learn the generalization and will require additional information from other features, such as [consonantal], which condition the alternation. The phonemes used for the experiments are presented in Table 4.

7.1.2 Vowel Shortening in Yawelmani

Yawelmani is a dialect of Yokuts, an Indigenous North American language spoken in California. Its phonemic inventory consists of consonants exhibiting contrastive aspiration and glottalization, and vowels with an underlying length contrast (Hockett 1967; Kenstowicz 1994). The 7 phonemes used to generate the dataset are summarized in Table 6.

Yawelmani exhibits a process of vowel shortening, where underlying long vowels are shortened on the surface. This alternation can be observed, for instance, in imperative and non-future verb stems (Kenstowicz 1994, p. 108).

(3)

future	**imperative**	**non-future**	**gloss**
wo:n-en	won-ko	won-hin	‘hide’
la:n-en	lan-ka	lan-hin	‘hear’
me:k-en	mek-ka	mek-hin	‘swallow’

Segment	o	i	o:	i:	t^h	g	n
high	−	+	−	+	0	+	0
low	−	−	−	−	0	−	0
tense	−	+	−	+	0	0	0
front	−	+	−	+	0	0	0
back	+	−	+	−	0	0	0
round	+	−	+	−	−	−	−
long	−	−	+	+	0	0	0
cons	−	−	−	−	+	+	+
son	+	+	+	+	−	−	+
cont	+	+	+	+	−	−	−
delrel	0	0	0	0	−	−	0
approx	0	0	0	0	−	−	−
nasal	−	−	−	−	−	−	+
voice	+	+	+	+	−	+	+
lab	0	0	0	0	−	−	−
labdent	0	0	0	0	−	−	−
cor	0	0	0	0	+	−	+
ant	0	0	0	0	−	0	+
distr	0	0	0	0	−	0	−
strid	0	0	0	0	−	0	−
lat	0	0	0	0	−	−	−
dor	0	0	0	0	−	+	−
constrgl	0	0	0	0	+	−	−
spreadgl	0	0	0	0	−	−	−

Table 6:
Feature chart for Yawelmani phonemes used in experiments

Kenstowicz and Kisseberth (1979, p. 84) propose that the condition for vowel shortening is two consecutive consonants.

(4) Vowel Shortening (VS)
$\left[\begin{smallmatrix} -\text{cons} \\ +\text{long} \end{smallmatrix} \right] \longrightarrow [-\text{long}] \; / \; \_\_\_ \; [+\text{cons}][+\text{cons}]$

We adopt Kenstowicz and Kisseberth's rule in (4), since it provides a clear view into the contrast between 2-ISL and 3-ISL function learning, where the change affects only a single feature.[10] We expect

[10] We acknowledge that there are alternative analyses of vowel shortening in Yawelmani. For instance, Archangeli (1991) analyzes these alternations as an

Table 7:
Feature chart for Chukchi phonemes used in experiments

Segment	i	o	m	p	t	ɬ	s
high	+	−	0	0	0	0	0
low	−	−	0	0	0	0	0
tense	+	+	0	0	0	0	0
front	+	−	0	0	0	0	0
back	−	+	0	0	0	0	0
round	−	+	−	−	−	−	−
long	−	−	0	0	0	0	0
cons	−	−	+	+	+	+	+
son	+	+	+	−	−	−	−
cont	+	+	−	−	−	+	+
delrel	0	0	0	−	−	+	+
approx	0	0	−	−	−	−	−
nasal	−	−	+	−	−	−	−
voice	+	+	+	−	−	−	−
lab	0	0	+	+	−	−	−
labdent	0	0	−	−	−	−	−
cor	0	0	−	−	+	+	+
ant	0	0	0	0	+	+	+
distr	0	0	0	0	−	−	−
strid	0	0	0	0	−	−	+
lat	0	0	−	−	−	+	−
dor	0	0	−	−	−	−	−

that [long] alone does not carry sufficient information to infer the underlying function on its own.

7.1.3 Schwa epenthesis in Chukchi

Chukchi is an endangered Chukotko-Kamchatkan language spoken in Siberia. From its phonemic inventory of five vowels and 13 consonants (Dunn 1999), seven were chosen for the experiments (Table 7).

Chukchi was selected for this study because it exemplifies a different type of phonological pattern, namely, final ə-epenthesis (see

emergent consequence of prosodic well-formedness, and Blevins (2004) argues that many length alternations are morphological or lexical in origin.

examples in (5)). This case is particularly interesting because, unlike in the previous processes, this transformation affects every single feature, and predicting sufficient featural combinations for learning is not an immediate and trivial task. In English, for example, it was evident that [nasal] depended on contextual information from [cons]; here, on the other hand, the specifications required for successful learning of individual features are far less transparent.

Krause (1980) identifies three distinct environments in which /ə/ may be inserted, typically to break up biconsonantal or triconsonantal clusters. For the purposes of this paper, we only focus on the insertion of schwa to break up word-final bi-consonantal clusters. Examples of this phenomenon are shown below.

(5)

noun-ABS.SG	**noun-ABS.PL**	**gloss**
miməl	mimlət	'water'
lewət	lewtət	'mouth'
qepəl	qeplət	'ball'

In these examples, the underlying representations arguably are /miml, lewt, qepl/ for the singular forms and /mimlt, lewtt, qeplt/ for the plural forms. A rule like the one in (6) applies to break up the word-final consonant cluster.

(6) Final epenthesis (FE)
$\text{ø} \longrightarrow \text{ə} \;/\; [+\text{cons}]\_\_[+\text{cons}] \ltimes$

We acknowledge that the distribution of epenthetic schwa in Chukchi is more complex than modeled in (6), as it may interact with additional phonological processes, including final vowel deletion (Dunn 1999). Our goal, however, is to test the effectiveness of FxF(SOSFIA) on this type of process. That is the reason why we restrict the process to one environment: the insertion of /ə/ between two consonants at a word boundary (Krause 1980, p. 53).

Final devoicing and o-raising in Polish 7.1.4

The final phonological phenomena that we consider involve the interaction of two strictly local processes. Rule interaction has a rich history in phonological theory (Kiparsky 1973; Baković 2007; Chandlee *et al.* 2018; Baković and Blumenfeld 2024).

In this study, we focus on an *opaque* interaction between two processes in Polish: word-final obstruent devoicing and ɔ-raising. Data in (7) presents selected cases of /ɔ/ raising to [u] when followed by voiced obstruents and approximants while remaining the same in the context of a following nasal.

(7)

singular	**plural**	**gloss**
ruk	rɔg-i	'horn'
sɔk	sɔk-i	'juice'
ʐwup	ʐwɔb-ɨ	'crib'
snɔp	snɔp-ɨ	'sheaf'
bur	bɔr-ɨ	'forest'
dzwɔn	dzwɔn-ɨ	'bell'
dɔm	dɔm-ɨ	'house'

The interaction is called opaque because there are forms like [ruk] 'horn, sg.' and [ʐwup] 'crib, sg.' where ɔ-raising has applied, yet the following sound in the surface form is devoiced. Typical analyses, discussed below, account for this by applying ɔ-raising before word-final obstruent devoicing.

Similarly to the previous cases, we acknowledge that the ɔ-raising process is complex and has been the subject of various analyses (see Bethin 1978; Buckley 2001; Gussmann 2007). Here we adapt the SPE-style analysis of the rule described in Kenstowicz (1994, p. 77). To correctly derive the surface forms, ɔ-raising (9) must precede final obstruent devoicing (8). This creates a counterbleeding type of order, as the reverse order would prevent ɔ-raising from applying.

(8) Final Devoicing (FD)

$$[-\text{sonorant}] \longrightarrow [-\text{voice}] \;/\; \_\_\_ \ltimes$$

(9) ɔ-raising (OR)

$$\begin{bmatrix} -\text{cons} \\ +\text{back} \\ -\text{low} \end{bmatrix} \longrightarrow [+\text{high}] \;/\; \_\_\_ \begin{bmatrix} +\text{cons} \\ +\text{voice} \\ -\text{nasal} \end{bmatrix} \ltimes$$

For example, consider the underlying representation of 'horn' /rog/. OR applies first yielding an intermediate form [rug]. Then FD applies yielding the surface form [ruk]. On the other hand, if FD applied first it would yield [rok], to which OR would not apply because the final

obstruent of this intermediate form is voiceless, predicting an incorrect surface form.

FD is a 2-ISL function, while OR is 3-ISL. Their ordered composition (OR then FD) can be modeled with a 3-ISL function; call it CB (for counterbleeding). The finite-state representation of CB does not literally execute OR and then FD; for instance, there is no intermediate representation [rug] that appears in any execution of CB (see Beesley and Karttunen 2003 for additional discussion regarding this point). Interestingly, as will become apparent in the following section, learning over features naturally separates these processes. Chandlee *et al.* (2018) provide additional discussion regarding how k-ISL DFTs can model a range of processes which interact opaquely.

The phonemes chosen for the experiments are presented in Table 8.

Feature	a	ɔ	ɔ̃	u	b	t	d	n	ɕ	ʐ
voice	+	+	+	+	+	−	+	+	−	+
son	+	+	+	+	−	−	−	+	−	−
cons	−	−	−	−	+	+	+	+	+	+
cont	+	+	+	+	−	−	−	−	+	+
cor	0	0	0	0	−	+	+	+	+	+
ant	0	0	0	0	0	+	+	+	−	−
dor	0	0	0	0	−	−	−	−	−	−
lab	0	0	0	0	+	−	−	−	−	−
distr	0	0	0	0	0	−	−	−	−	−
delrel	0	0	0	0	−	−	−	0	+	+
nasal	−	−	+	−	−	−	−	+	−	−
lat	0	0	0	0	−	−	−	−	−	−
high	−	−	−	+	0	0	0	0	0	0
back	+	+	+	+	0	0	0	0	0	0
low	+	−	−	−	0	0	0	0	0	0
tense	−	−	−	+	0	0	0	0	0	0
round	−	+	+	+	0	0	0	0	0	0

Table 8: Feature chart for Polish phonemes used in experiments

7.2 Data generation

For each of the four k-ISL functions, a corresponding k-ISL DFT is constructed, and a characteristic sample consisting of underlying-surface form pairs is generated accordingly. Following Proposition 1, a set of all logically possible n-grams over Σ^* up to length $2k-1$ is created to serve as the set of potential underlying forms.[11] The constructed k-ISL DFTs are then used to generate the corresponding surface forms (outputs). These same transducers are also used to evaluate the results.

As previously mentioned, a constraint was imposed on the total size of each characteristic sample (CS) to ensure that the number of generated input–output pairs did not exceed 20,000. Since the size of a CS, |CS|, depends on both the size of the alphabet ($|\Sigma|$) and the function's k value, these parameters were adjusted accordingly. For the 2-ISL function, $|\Sigma| = 26$, resulting in a CS of 18,287 pairs. For the 3-ISL functions, a reduced alphabet size of $|\Sigma| = 7$ was used, yielding CSs of 17,206 input-output pairs. We ran additional experiments on Polish, where the size of the input alphabet was increased to 10 to further showcase the difference between the growth of CS when learning over segments vs. features (see Table 18 for details).

We introduce terminology corresponding to each of the types of samples generated. There are five sample types that perform various roles, which are listed below.

- a sample (**CS**): a characteristic sample over segmental representations
- a near-minimal *segmental* sample (**M**): a near-minimal sample randomly drawn from CS with AMβA calling SOSFIA.
- a feature-based sample ($\mathbf{S}_{(\Phi,\phi)}$): a sample extracted from CS whose inputs and outputs are projected according to Φ and ϕ, respectively, for each feature $\phi \in F$. For example, $\mathrm{S}_{([\text{high}],\text{high})}$ is a sample extracted from CS to predict [high] from [high], while

[11] The generated data is constructed to be extensionally consistent with the four functions, but it is not intended to represent attested or naturalistic linguistic patterns. Learning with naturalistic data will be addressed in the next stage of this research.

$S_{(\left[\begin{smallmatrix}\text{high}\\ \text{round}\end{smallmatrix}\right],\text{high})}$ means that two features [high] and [round] are projected to predict [high].

- a near-minimal feature sample ($\mathbf{M}_{(\Phi,\phi)}$): a near-minimal feature sample randomly drawn from $S_{(\Phi,\phi)}$ with AMβA calling the original SOSFIA algorithm.
- a synthesis of $M_{(\Phi,\phi)}$ for all $\phi \in F$ (**MF**): a segmental sample reconstructed by combining the minimal feature-based samples $M_{(\Phi,\phi)}$ for each feature $\phi \in F$. To estimate how many full segmental input-output pairs can be formed from these minimized components, we identify compatible combinations of feature vectors that jointly satisfy subsets of the segmental transformations observed in CS. For example, the following feature-based data points

$$\begin{aligned}
&\dots\\
&([\begin{smallmatrix}-+-\\ ++-\end{smallmatrix}],[++-]) \in S_{([\text{nasal, cons}],\text{cons})},\\
&([0+-],[0+-]) \in S_{(\text{cor,cor})},\\
&([0--],[0--]) \in S_{(\text{lat,lat})},\\
&\dots
\end{aligned}$$

  are all subcomponents of a segmental pair $(\text{ant}, \tilde{\text{ant}}) \in \text{CS}$. The synthesized set $\mathbf{M}_\mathbf{F}$ reflects an *upper bound* on the number of segmental input–output pairs that can be reconstructed from the featural data.

7.3 *Results*

This section presents a summary and interpretation of the results. First, as expected, all four functions were successfully learned from the characteristic sample, both with SOSFIA and with FxF(SOSFIA). Second, when the data sample input to the learning algorithms was reduced with AMβA, FxF(SOSFIA) consistently outperformed the segmental baseline SOSFIA in terms of the amount of data sufficient for correct generalization. The advantage of FxF was especially pronounced in cases involving a single feature change and in the epenthesis process. The Polish case, which involved the interaction of multiple features, required additional justification and further experimentation to confirm the effectiveness of the approach. Third, the results demonstrate

that the data reduction obtained by AMβA for FxF(SOSFIA) is more substantial than what AMβA obtains for SOSFIA. For the feature-based FxF(SOSFIA) the reductions in size of the CS ranged from 98% in the best case to 52% in the worst. By contrast, for the segment-based SOSFIA, AMβA only achieved a reduction ranging from 66% to just 0.15%. We report here only the results most relevant to the main discussion; full experimental details and quantitative data are provided in Appendices A, B, and C. [12]

In the ensuing discussion of the learning results, we refer to features in F as either *sufficient* or *insufficient*. A feature ϕ is termed insufficient if its values alone do not provide enough information to infer the target function. Formally, this means the sample $S_{(\{\phi\},\phi)}$ is not functional, and thus `FeatSearch` must add additional features to yield a functional sample. A feature ϕ is termed sufficient if it is not insufficient (and so the sample $S_{(\{\phi\},\phi)}$ is functional).

As mentioned in Section 6.3, it is often true that multiple feature combinations are able to successfully predict a feature ϕ. While `FeatSearch` is defined to stop once the first one is found, our implementation returned all feature sets with the smallest cardinality capable of predicting ϕ since we were curious about them. As such, when calculating the synthesized sample M_F, we used the combination which yielded the smallest functional sample for each insufficient feature (or randomly selected one if there was more than one equally smallest sample). While this technique was adequate to demonstrate the core concept proposed here, future research will explore more linguistically informed methods for selecting appropriate featural combinations in cases where individual features are insufficient for learning (see Section 8 for more details).

7.3.1 English

Of the 22 features used in the English vowel nasalization case study, only one feature, particularly [nasal], was found to be insufficient, as

[12] Appendix A provides sizes of the extracted samples $|S_{(\Phi,\phi)}|$, averaged $|M_{(\Phi,\phi)}|$, and standard deviation for all *sufficient* and selected *insufficient* features, while Appendix B provides $|M_{(\Phi,\phi)}|$ and M_F for each of the 10 runs, separately. Appendix C provides the sizes of $S_{(\Phi,\phi)}$ for all the combinations with *insufficient* features.

expected. The remaining 21 features were individually sufficient and yielded functional samples on their own. The projected samples $S_{(\Phi,\phi)}$ for these features varied in size depending on the number of values ϕ takes. For instance, the binary feature $\phi = [\text{voice}]$ (with values $\{+,-\}$, as shown in Table 4) yielded a sample $S_{(\text{voice,voice})}$ comprising 14 input-output pairs. In contrast, the ternary feature [cor] produced $S_{(\text{cor,cor})}$ with 39 input-output pairs. These samples were successfully reduced using AMβA to average sizes of $M_{(\text{voice,voice})} = 9$ and $M_{(\text{cor,cor})} = 23$, respectively. Table 19 provides the average values of $M_{(\Phi,\phi)}$ across all features, and Table 23 provides complete results corresponding to each individual feature.

We identified 10 feature combinations yielding functional samples for the insufficient feature [nasal], whose sizes also varied depending on the size of the resulting featural alphabets Σ_Φ. As was already presented in Figure 2, the combination of features [nasal, cons] resulted in the input alphabet $\Sigma_{\left[\begin{smallmatrix}\text{nasal}\\ \text{cons}\end{smallmatrix}\right]} = \{[\begin{smallmatrix}-\\-\end{smallmatrix}], [\begin{smallmatrix}-\\+\end{smallmatrix}], [\begin{smallmatrix}+\\+\end{smallmatrix}]\}$. Recall that the remaining logical feature value $[\begin{smallmatrix}+\\-\end{smallmatrix}]$ was not considered as we assume English does not have underlying nasal vowels.

Accordingly, the projected sample $S_{\left(\left[\begin{smallmatrix}\text{nasal}\\ \text{cons}\end{smallmatrix}\right],\text{nasal}\right)}$ contained 39 pairs. Other combinations involving [nasal] produced larger sample sizes, particularly 84 and 155 pairs, depending on whether the featural alphabet contained four or five elements, respectively. In all cases, AMβA was able to reduce the samples substantially. Since the pair [nasal, cons] yielded the smallest functional sample, we used this combination to construct the synthesized featural sample M_F.

Table 9 presents a sample of cardinalities of the extracted samples for features ϕ (specifically, [voice], [coronal], and [nasal]). The Feature sets Φ, in the first column, here as well as in the corresponding tables for the remaining test cases, indicate what features were used

Feature set Φ	$\lvert S_{(\Phi,\phi)}\rvert$	Avg. $\lvert M_{(\Phi,\phi)}\rvert$	SD
[**voice**]	14	9	2.66
[**cor**]	39	23	4.76
[**nasal**, cons]	39	31	3.89
[**nasal**, front]	84	68	10.9
[**nasal**, approx]	155	117	16.22

Table 9:
Selected representative samples $\lvert S_{(\Phi,\ \phi)}\rvert$, Avg. $\lvert M_{(\Phi,\ \phi)}\rvert$ and standard deviation (SD) for VN in English for features [voice], [coronal], and [nasal]

for input projections for ϕ. The bolded feature represents the specific feature ϕ whose values are predicted. When only a single bolded feature appears, such as [**voice**], it means that the values of that feature alone were sufficient to define the characteristic sample for the corresponding feature. In cases where multiple features appear, as in [**nasal**, cons], the bolded feature (ϕ) is the (insufficient) predicted feature, while the full set in brackets represents Φ, the set of features that give a functional sample for ϕ.

Given the data samples $M_{(\Phi,\phi)}$ selected by AMβA for each $\phi \in F$, we reconstructed the corresponding segmental sample M_F. This enabled direct comparison with the segmental characteristic sample CS and its averaged AMβA-reduced counterpart M (see Table 10).

Table 10: Averaged segmental sample $|M|$ and averaged synthesized featural $|M_F|$ and their corresponding SDs in English

$\|CS\|$	Segments		Features	
	Avg. $\|M\|$	SD	Avg. $\|M_F\|$	SD
18,278	6,157	650.21	188	10.06

The average size of a reduced characteristic sample M derived from the segmental representation was 6,157. In contrast, the FxF learner identified significantly smaller successful samples, with M_F averaging only 188 elements. Starting from an original sample of $|CS| = 18{,}278$, segmental learning achieved a 66% reduction, whereas featural learning achieved a 98% reduction. This substantial difference demonstrates that featural decomposition in the FxF framework yields more compact and data-efficient generalizations.

7.3.2 Yawelmani

This case study examines vowel shortening in Yawelmani, a process that, like English vowel nasalization, targets a single feature: [long]. What differs is that the environment for the change is larger and thus requires more memory.

Contrary to our expectations, *all* features were sufficient for learning the vowel shortening function, including [long] in isolation. This finding reflects a representational asymmetry: the values assumed for the feature [long] themselves encode distributional distinctions that parallel the relevant contextual split. In our dataset, [long] takes on

ternary values: {+, –, 0}, where {+, –} are exclusively associated with vowels, and 0 with consonants. As a result, the learner can indirectly infer the necessary contextual distinctions purely from this feature's values. This outcome shows that not only the featural decomposition but also the range and interpretation of values can play a decisive role in learnability.

Table 11 presents results from AMβA's reductions of individual feature-based samples $S_{(\Phi,\phi)}$. In the best-performing case, the sample for [cor] was reduced by 59%. The smallest reduction was observed for the target feature [long], which is expected, as it is the only feature that changes and, therefore, must be represented with particular data that cover the necessary paths in the transducer.

Feature set Φ	$\lvert S_{(\Phi,\phi)}\rvert$	Avg. $\lvert M_{(\Phi,\phi)}\rvert$	SD
[**voice**]	54	33	10.11
[**round**]	62	30	7.39
[**ant**]	309	136	19.28
[**cor**]	336	139	24.05
[**long**]	363	317	49.02

Table 11:
Selected representative samples $\lvert S_{(\Phi,\ \phi)}\rvert$, Avg. $\lvert M_{(\Phi,\ \phi)}\rvert$ and standard deviation (SD) for VS in Yawelmani for features [voice], [round], [anterior], [coronal], and [long]

Interestingly, the projected sample sizes $S_{(\Phi,\phi)}$ varied even among features with identical value sets. For example, both [voice] and [round] are binary features with values {+, –}, yet their sample sizes differed. This disparity is attributable to the distribution of feature values among the segments. Because $S_{(\Phi,\phi)}$ is projected from CS, the features whose values were less evenly distributed among the segments resulted in smaller projected samples. For example, in our data, only one of the seven segments was marked as [–voice], whereas two segments were [–round]; this skew resulted in a smaller sample for [voice].

Overall sample sizes are summarized in Table 12. The segmental sample M, derived without feature factorization, had an average

$\lvert$**CS**$\rvert$	Segments		Features	
	Avg. $\lvert M\rvert$	SD	Avg. $\lvert M_F\rvert$	SD
17,206	16,583	702.06	952	49.90

Table 12:
Averaged segmental sample $\lvert M\rvert$ and averaged synthesized featural $\lvert M\rvert_F$ and their corresponding SDs in Yawelmani

size of 16,583 (4% reduction). In contrast, the synthesized featural sample M_F required only 952 examples on average. The larger transducer sizes observed in the Yawelmani case, driven by the larger k value, contributed to the higher sample size, which was expected as discussed in Section 5. Nevertheless, the difference between reduction over segmental vs. featural is much more pronounced than in the English case study.

7.3.3 Chukchi

The Chukchi epenthesis function presents a different learning scenario, where each feature is affected due to the inserted segment. Surprisingly, of the 22 features used to represent Chukchi segments, 13 were *sufficient* on their own. These tended to be features that, similar to the situation in Yawelmani, implicitly distinguished vowels from consonants via their value patterns. For instance, [labial] assigns the value 0 to vowels and either + or − to consonants, providing a natural partition of the inventory. The remaining 9 features were *insufficient* individually, but when paired with complementary features that supplied the missing contrast, they successfully learned the target function. For example, while [sonorant] alone does not separate vowels from consonants, its combination with [nasal] provides enough information to infer the correct generalization.

Sample sizes for the projected datasets $S_{(\Phi,\phi)}$ varied widely, and in many cases were substantially larger than those in the English and Yawelmani experiments. For example, the combination $S_{(\left[\begin{smallmatrix}\text{strid}\\ \text{dor}\end{smallmatrix}\right],\text{strid})}$ produced 1,236 input-output pairs due to the resulting 4-element alphabet $\Sigma_{\left[\begin{smallmatrix}\text{strid}\\ \text{dor}\end{smallmatrix}\right]}$. Table 13 shows the reductions obtained with AMβA.

Table 13: Selected representative samples $|S_{(\Phi,\phi)}|$, Avg. $|M_{(\Phi,\phi)}|$ and standard deviation (SD) for FE in Chukchi for features [tense], [high], [round], [labial], [continuant], and [strident]

Feature set Φ	$\lvert S_{(\Phi,\phi)}\rvert$	Avg. $\lvert M_{(\Phi,\phi)}\rvert$	SD
[**tense**]	62	32	5.69
[**high**]	309	165	30.34
[**round**, high]	309	225	30.28
[**lab**]	363	313	8.88
[**cont**, tense]	363	306	18.02
[**strid**, dor]	1,236	955	58.46

Despite the challenge posed by the global nature of the epenthesis transformation, FxF learning remained robust.

$\lvert CS\rvert$	Segments		Features	
	Avg. $\lvert M\rvert$	SD	Avg. $\lvert M_F\rvert$	SD
17,206	17,181	28.87	2,331	46.27

Table 14:
Averaged segmental sample $|M|$ and averaged synthesized featural $|M|_F$ and their corresponding SDs in Chukchi

Table 14 summarizes the overall segmental and featural sample sizes. The reduced characteristic segmental sample M remained nearly identical in size to the original CS (avg. $|M| = 17{,}181$), showing that little to no compression was achieved without factorization. By contrast, the FxF approach yielded a significantly smaller sample of 2,331 input-output pairs on average. Although this is larger than the feature-based sample size obtained for the Yawelmani function (avg. $M_F = 952$), the difference reflects the more complex nature of the process (feature insertion vs. feature changing). In this context, the observed 86% reduction remains a strong result, demonstrating the scalability and effectiveness of featural decomposition.

Polish 7.3.4

The Polish case study presents a 3-ISL function involving two phonological processes in a counterbleeding relation. While the featural learner remained successful, the degree of sample reduction was significantly less than what was found with the English, Yawelmani, and Chukchi case studies. This was not due to the compositional nature of the function, but rather to the complexity of one of its components, namely, the transformation of /ɔ/ to [u], which required coordinated information from multiple features.

In this alternation, features such as [high] and [tense] were insufficient on their own and required combination with three additional features to produce functional samples. Unlike the processes that target broader classes (e.g. English [+nasal, –cons]) or Yawelmani (e.g., [+long, –cons]), the ɔ-raising process in Polish targets a single segment, which necessitates distinctions between similar segments, here vowels. For instance, [low] is needed alongside [high]

to exclude vowel 'a', which does not undergo change. Contextual triggers, on the other hand, could be successfully encoded by features like [voice] and [sonorant].

Table 15: Selected representative samples $|S_{(\Phi,\phi)}|$, Avg. $|M_{(\Phi,\phi)}|$ and standard deviation (SD) for FD and OR in Polish for $|\Sigma_{\text{seg}}| = 7$, for features [nasal], [sonorant], [round], [voice], [high], [tense]

Feature set Φ	$\lvert S_{(\Phi,\phi)}\rvert$	Avg. $\lvert M_{(\Phi,\phi)}\rvert$	SD
[**nasal**]	54	29	14.92
[**son**]	62	27	12.77
[**round**]	336	137	51.76
[**voice**, son]	336	214	10.73
[**high**, voice, son, low]	8,250	7,119	907.70
[**tense**, voice, son, round]	8,250	7,372	806.83

Table 15 above presents a subset of representative feature combinations for $|\Sigma| = 7$. While most projected featural samples $S_{(\Phi,\phi)}$ were reduced by AMβA, the reduction was less prominent for features that required extensive combinations. In particular, the input alphabet for feature [high] ($V_{[\text{high,voice,son,low}]}$) was reduced only marginally, reflecting the difficulty of compressing transformations involving singleton segments. As a result, the projected sample contained 8,250 input-output pairs and was reduced only by approximately 14%.

Because the reduced samples $M_{(\Phi,\phi)}$ were significantly large, the averaged M_F samples remained large as well. Nevertheless, when compared to reduction over segmental data representation, FxF still comes out ahead. Table 16 shows that the FxF procedure achieved a 52% reduction with AMβA, as opposed to the reduction with AMβA obtained for segmental representations, which was 12%.

To evaluate the robustness of the feature-based learner under increasing inventory size, we incrementally expanded the phoneme set by one symbol at a time ($|\Sigma_{\text{seg}}| = 8, 9, 10$) and re-ran the experiments for each resulting alphabet size. Table 17 shows that although the projected $S_{(\Phi,\phi)}$ samples grew in size, AMβA continued to yield con-

$\lvert CS\rvert$	Segments		Features	
	Avg. $\lvert M\rvert$	SD	Avg. $\lvert M_F\rvert$	SD
17,206	15,169	2,063.50	8,235	208.39

Table 16:
Averaged segmental sample $|M|$ and averaged synthesized featural $|M_F|$ and their corresponding SDs in Polish

Feature set Φ	$\lvert\Sigma_{\text{seg}}\rvert = 8$		$\lvert\Sigma_{\text{seg}}\rvert = 9$		$\lvert\Sigma_{\text{seg}}\rvert = 10$	
	$\lvert S_{(\Phi,\phi)}\rvert$	$\lvert M_{(\Phi,\phi)}\rvert$	$\lvert S_{(\Phi,\phi)}\rvert$	$\lvert M_{(\Phi,\phi)}\rvert$	$\lvert S_{(\Phi,\phi)}\rvert$	$\lvert M_{(\Phi,\phi)}\rvert$
[**nasal**]	54	43	62	29	62	43
[**son**]	62	32	62	34	62	23
[**round**]	336	162	336	136	336	153
[**voice** son]	363	241	363	223	363	235
[**high** voice son low]	8,446	7,416	–	–	–	–
[**tense** voice son round]	8,446	6,656	–	–	–	–
[**high** voice nasal low]	–	–	17,892	13,122	17,892	8,122
[**tense** voice nasal round]	–	–	17,892	12,482	17,892	17,842

Table 17:
Selected representative samples $|S_{(\Phi,\phi)}|$ and $|M_{(\Phi,\phi)}|$ FD and OR in Polish with increasing $|\Sigma_{\text{seg}}|$ of 8, 9 and 10 phonemes

siderable reductions. Table 18 reports the overall segmental and featural reductions at the full-sample level. While the segmental learner benefited modestly from data pruning (at best 21% reduction with $|\Sigma_{\text{seg}}| = 10$), the feature-based learner achieved far more efficient compression. Particularly, with the largest alphabet, the reduction peaked at 82%. These results suggest that even in cases involving tightly constrained alternations and rare, singleton targets, the FxF approach remains scalable and resilient to inventory growth.

Table 18: Samples $|S|$, $|M|$, and $|M_F|$ for increasing $|\Sigma_{\text{seg}}|$ of 8, 9, and 10 phonemes in Polish

$\|\Sigma_{\text{seg}}\|$	$\|CS\|$	Segments $\|M\|$	Features $\|M_F\|$
8	33,352	30,344	8,283
9	59,868	51,161	16,272
10	101,110	79,540	17,945

This is particularly noticeable for feature [high] in the $|\Sigma_{\text{seg}}| = 10$ condition. While the segmental alphabet is of size 10, the features used to predict high yield an inventory of 7. We suspect that given the full inventory of Polish phonemes, the projected sample sizes for [high] will remain approximately the same as when $|\Sigma_{\text{seg}}| = 10$. More generally, although increasing $|\Sigma_{\text{seg}}|$ will continue to enlarge the segmental sample, the featural alphabet $|\Sigma_\Phi|$ sizes eventually saturate: once the inventory already realizes all relevant combinations of feature values, adding additional segments does not introduce any new featural patterns, and thus does not increase $|\Sigma_\Phi|$.

8 DISCUSSION AND FUTURE WORK

The empirical studies presented in Section 7 demonstrate that the Factor-by-Feature (FxF) approach significantly enhances the learnability of k-ISL functions by reducing the size of the required training samples. By introducing a systematic featural decomposition, FxF enables learners to utilize the inherent structure of phonological systems, which in turn reduces the complexity and size of the training sample requirement. The success of this approach can be consistently observed across a range of phonological processes, where the feature-based learner outperforms the baseline segmental learner in data efficiency.

A key contribution of FxF is the modularization that it brings. This approach allows factoring the learning problem into smaller, feature-specific subproblems, which leads to an efficient reduction of the underlying transducers. For processes involving a single feature alternation (e.g., English vowel nasalization, Yawelmani vowel shortening), the improvement is especially pronounced, with sample size reductions of up to 98%. Even in more complex settings like Polish, where

rule interaction and singleton transformations make learning more challenging, FxF still achieved a substantial reduction (over 50%) relative to the original sample size.

Another important contribution is the stability of featural sample sizes under increased alphabet size. As shown in the Polish experiments, the size of featural samples grew much more slowly than segmental ones, highlighting the scalability of the approach. This is crucial for practical applications, especially when working with large phonemic inventories or naturalistic data.

It is also worth noting that the alternating features in Polish naturally separate the two processes: in ɔ-raising, the change is carried by features [tense] and [high] (with [voice] and [sonorant] appearing only in the triggering context), whereas in word-final devoicing [voice] itself is the feature that changes. This indicates that there are circumstances where FxF can track distinct processes, which invites future investigation.

Taken together, these findings support the hypothesis that phonological learning benefits not only from prior knowledge about the type of function (e.g. k-ISL), but also from the representational flexibility allowed by featural decomposition. The FxF method leverages this flexibility to construct more compact and learnable representations. This offers a promising direction for computational models of phonological acquisition and phonological processes in low-resource languages while remaining interpretable and grounded in grammatical-inference methods with theoretical guarantees.

Despite the advantages of incorporating featural representation in learning classes of sequential functions, the FxF approach could be further advanced in various ways. First, while the present study holds the memory window k constant and assumes the structure of the underlying transducer, the empirical results suggest that k can, in many cases, be minimized for individual features. Particularly, the features that do not reflect any changes between the inputs and outputs could, in theory, be learned assuming a much simpler underlying structure, i.e., 1-ISL DFT.

The second challenge for the FxF approach is the expensive combinatorial search for useful feature combinations. This is particularly true when many features are needed to correctly represent the process. One way to address this issue would be to incorporate structure into

the feature space, such as *feature geometry* (Clements 1985). A more structured representation of the feature system may allow for a more efficient and linguistically plausible traversal through the combinatorial feature space.

A useful extension to this search would be to consider the size of Σ_Φ for particular combinations Φ. For example, as can be seen in Table 29 in Appendix C, the choice of featural combination for the insufficient feature [nasal] has a significant impact on sample sizes. Particularly, the $S_{\left[\begin{smallmatrix}\textbf{nasal}\\ \textbf{low}\end{smallmatrix}\right]}$ sample has 336 pairs projected from the CS sample, while $S_{\left[\begin{smallmatrix}\textbf{nasal}\\ \textbf{cor}\end{smallmatrix}\right]}$ has 1300 pairs. In this study, we manually picked the combinations that provided the most efficient functional samples. Future improvements should incorporate this option into the `FeatSearch` algorithm.

Third, the current work consistently assumes left-to-right processing of the input string. Although it is known that the direction of processing does not affect the structure of a k-ISL DFT, the potential effects of directionality on learnability have not yet been explored. For most of the functions discussed here, the necessary context for change was on the right, which requires the use of 'waiting transitions', i.e. transitions that output λ (the empty string), postponing output until the relevant context falls within the k window. If, instead, the processing direction was reversed to *right-to-left*, the relevant context would be seen before the target symbol, eliminating the need for such waiting transitions. Such a change in directionality can thus reduce the number of transitions whose outputs differ from the corresponding inputs. Under the default assumption of input-output identity at the beginning of the learning process, switching the direction of processing may lead to a further reduction of characteristic samples.

Fourth, the current study is limited to k-ISL functions, which, while capturing a broad class of phonological processes, do not cover *all* types of attested phenomena. Extending the framework to capture more complex processes, such as long-distance or tonal, will require adapting algorithms that handle tier-based representations, such as k-OTSL functions (Burness and McMullin 2019; Burness *et al.* 2021), or the tier definite and reverse definite classes (Lambert and Heinz 2024). Investigating these extensions is necessary for building models that are both typologically comprehensive and empirically grounded.

At the same time, because the FxF framework is implemented in terms of deterministic finite-state transducers, it is natural to ask how this architecture relates to learning approaches that operate directly over symbolic rules, such as Wicentowski's model of multilingual inflectional morphology (Wicentowski 2002) or Belth's (2023; 2024) learning-based accounts of phonological processes and tiers. We leave a systematic comparison with these rule-based learners for future work.

Finally, the data in this study was synthetically generated. To more rigorously evaluate the FxF approach, future work should incorporate naturalistic data. An immediate first step would be to apply the model to problem sets from phonology textbooks, which provide more realistic yet curated examples that respect different components of phonological grammar, such as phonotactic constraints. Ultimately, extending the analysis to include data found in the corpora of child-directed speech will allow for a more thorough assessment of the approach's plausibility as a model of human language acquisition or language variation.

Taken together, these future directions point toward the need for a flexible and adaptive model that can adjust its parameters as needed for particular functions. As the FxF framework is inherently modular and compatible with various learners, it seems to be well suited to support such adaptability. Future extensions could lead to a learner that can determine whether another feature needs to be added to enrich the input, adjust the k value as needed (beginning from the default assumption that $k = 1$ for each feature $\phi \in F$), and select the most efficient direction of processing. Furthermore, it could also incorporate mechanisms for projecting relevant elements onto tiers where necessary, for example, to capture long-distance processes without compromising *locality*. Developing such a model could further reduce the characteristic sample size, making FxF-based learning especially suitable for low-resource languages, which remain underrepresented in language technology research and applications.

9 CONCLUSION

This work provides a way to apply grammatical inference algorithms to the learning of phonological alternations in a manner that remains faithful to the representations employed by phonological theory. The paper not only provides such a synthesis but also demonstrates that it comes with tangible benefits: the amount of data sufficient to learn a phonological process is reduced, in many cases significantly, when compared to the same grammatical inference algorithm (SOSFIA) operating over segments.

The FxF framework factors the learning of phonological processes into n-many processes, one for each feature, and aims to learn the processes that govern each individual feature. The overall grammar is represented by n-many DFTs, which process segmental input in parallel according to their featural makeup. The outputs of these DFTs are combined via direct product to recover the segmental output.

Provided that the featural combinations required to successfully predict an individual feature are small, the state space of the finite-state transducer to be learned will likewise be small. The reduction in model size is mirrored by the reduction of the characteristic sample required for successful inference. As the empirical results show, these smaller machines more than compensate for the fact that there are now n-many transducers to learn instead of a single one. In this way, this work provides an additional argument for phonological features, beyond the ones already adduced by phonologists and reviewed in Section 2: phonological features can facilitate the learning of phonological processes.

Empirically, we showed that a 2-ISL function with one feature changing, such as English vowel nasalization, can be learned from fewer than 200 input-output pairs, whereas the segment-based learner required over 6,000 examples. Even for processes of a different nature, such as segment insertion affecting every feature, FxF provides substantial benefits. This was exemplified by the final ə-epenthesis in Chukchi, where the feature-based learner inferred the function with approximately 86% less data than the segment-based learners, which required nearly the entire characteristic sample generated for that process.

While the present study focuses solely on a subclass of subsequential functions, the limitations and directions for future research identified in the previous section show the broader relevance of this approach to learning phonological processes. Future empirical studies can bring us closer to learning models that are linguistically grounded and interpretable, come with theoretical guarantees, are adaptable to low-resource settings, and are suitable for modeling language acquisition.

ACKNOWLEDGMENTS

We thank the audiences at the International Conference on Grammatical Inference (ICGI) 2023 and the North American Phonology Conference (NAPhC) 2025 for their helpful feedback. We are grateful to Jordan Kodner and Thomas Graf for insightful comments, and to the Mathematical Linguistics Reading Group at Stony Brook University for valuable discussions. We also thank the participants of the Subregular Phonology workshop (Rutgers University, 2024). This work was supported by NSF grant #2416183.

A APPENDIX: ALL AVERAGED SAMPLES

Table 19: Samples $|S_{(\Phi,\phi)}|$, Avg. $|M_{(\Phi,\phi)}|$ and standard deviation (SD) for all *sufficient* features and selected one for the *insufficient* feature for VN in English

Feature set Φ	$\lvert S_{(\Phi,\phi)}\rvert$	Avg. $\lvert M_{(\Phi,\phi)}\rvert$	SD
$\begin{bmatrix}\textbf{nasal}\\ \text{cons}\end{bmatrix}$	39	31	3.89
[**high**]	39	24	4.52
[**low**]	39	23	7.51
[**tense**]	39	25	9.42
[**front**]	39	20	8.09
[**back**]	39	24	5.65
[**round**]	14	14	0.67
[**long**]	14	8	1.79
[**cons**]	14	9	2.02
[**son**]	14	10	1.66
[**cont**]	14	9	3.47
[**delrel**]	39	25	4.99
[**approx**]	39	26	9.29
[**voice**]	14	9	2.66
[**lab**]	39	24	4.72
[**labdent**]	39	26	4.72
[**cor**]	39	23	4.76
[**ant**]	39	24	5.80
[**distr**]	39	26	3.92
[**strid**]	39	24	8.50
[**lat**]	39	28	4.06
[**dor**]	39	25	5.38

Feature set Φ	$\lvert S_{(\Phi,\phi)}\rvert$	Avg. $\lvert M_{(\Phi,\phi)}\rvert$	SD
[**high**]	363	148	27.13
[**low**]	62	32	6.92
[**tense**]	363	159	32.15
[**front**]	363	146	30.43
[**back**]	363	167	23.66
[**round**]	62	30	7.39
[**long**]	363	317	49.02
[**cons**]	62	36	5.47
[**son**]	62	39	7.77
[**cont**]	62	34	6.19
[**delrel**]	62	34	7.75
[**approx**]	62	33	7.67
[**nasal**]	54	41	6.68
[**voice**]	54	33	10.11
[**lab**]	62	35	4.17
[**labdent**]	62	33	4.51
[**cor**]	336	139	24.05
[**ant**]	309	136	19.28
[**distr**]	62	32	8.09
[**strid**]	62	30	6.75
[**lat**]	62	31	4.92
[**dor**]	336	169	17.89
[**constrgl**]	336	157	32.91
[**spreadgl**]	62	35	7.51

Table 20:
Samples $\lvert S_{(\Phi,\ \phi)}\rvert$, Avg. $\lvert M_{(\Phi,\ \phi)}\rvert$ and standard deviation (SD) for all *sufficient* features for VS in Yawelmani

Table 21:
Samples $|S_{(\Phi,\phi)}|$, Avg. $|M_{(\Phi,\phi)}|$ and standard deviation (SD) for all *sufficient* features and selected one for each *insufficient* feature for FE in Chukchi

Feature set Φ	$\lvert S_{(\Phi,\phi)}\rvert$	Avg. $\lvert M_{(\Phi,\phi)}\rvert$	SD
[**high**]	309	165	30.34
[**low**]	62	38	5.45
[**tense**]	62	32	5.69
[**front**]	309	166	31.76
[**back**]	309	174	22.55
[**round** high]	309	225	30.28
[**long**]	62	32	5.54
[**cons**]	62	31	4.95
[**son** low]	336	260	12.77
[**cont** tense]	363	306	18.02
[**delrel** long]	1,236	924	57.82
[**approx**]	62	34	6.24
[**nasal** son]	336	290	17.26
[**voice** cons]	363	305	21.22
[**lab**]	363	313	8.88
[**labdent**]	62	34	5.46
[**cor**]	363	305	17.30
[**ant** labdent]	1,236	922	61.73
[**distr** cor]	1,236	801	14.98
[**strid** dor]	1,236	955	58.46
[**lat**]	336	274	12.90
[**dor**]	62	32	4.55

Feature set Φ	$\lvert S_{(\Phi,\phi)}\rvert$	Avg. $\lvert M_{(\Phi,\phi)}\rvert$	SD
[**voice**, son]	336	214	10.73
[**son**]	62	27	12.77
[**cons**]	62	28	14.37
[**cont**]	62	26	7.70
[**cor**]	62	35	6.34
[**ant**]	336	157	17.27
[**dor**]	62	32	4.33
[**lab**]	62	25	14.26
[**distr**]	62	32	12.28
[**delrel**]	336	147	38.97
[**nasal**]	54	29	14.92
[**lat**]	62	31	15.93
[**high**, voice, son, low]	8,250	7,119	907.70
[**back**]	62	29	12.12
[**low**]	336	153	60.16
[**tense**, voice, son, round]	8,250	7,372	806.83
[**round**]	336	137	51.76

Table 22:
Samples $\lvert S_{(\Phi,\phi)}\rvert$, Avg. $\lvert M_{(\Phi,\phi)}\rvert$ and standard deviation (SD) for all *sufficient* features and selected one for the *insufficient* features for FD and OR in Polish

B APPENDIX: ALL INDIVIDUAL SAMPLES

Table 23: Cardinalities $|M_{(\Phi,\phi)}|$ and $|M_F|$ for all *sufficient* features and selected one for the *insufficient* feature, across 10 runs, for VN in English

Feature set Φ	$\lvert M_{(\Phi,\phi)}\rvert$									
	1	2	3	4	5	6	7	8	9	10
$\begin{bmatrix}\textbf{nasal}\\ \text{cons}\end{bmatrix}$	37	34	24	31	31	33	32	25	30	30
[**high**]	19	16	30	20	23	28	29	24	24	25
[**low**]	19	35	27	23	26	12	30	23	26	11
[**tense**]	32	29	27	18	33	33	3	21	32	23
[**front**]	21	32	6	23	8	26	23	19	27	19
[**back**]	29	30	17	26	22	22	17	20	34	24
[**round**]	11	8	8	12	8	7	6	5	8	11
[**long**]	5	7	7	9	10	7	7	9	11	9
[**cons**]	9	9	8	12	10	10	5	8	7	11
[**son**]	10	12	12	9	10	7	9	11	8	11
[**cont**]	11	8	5	11	12	10	13	3	6	9
[**delrel**]	25	20	17	23	28	33	28	28	21	30
[**approx**]	33	27	29	22	23	35	31	29	2	24
[**voice**]	7	10	11	9	6	13	6	10	5	11
[**lab**]	28	28	31	23	23	26	16	24	17	23
[**labdent**]	19	38	28	27	28	21	34	24	16	21
[**cor**]	14	25	22	18	21	25	22	32	23	25
[**ant**]	21	16	22	18	27	23	35	27	28	18
[**distr**]	29	26	31	19	27	26	21	26	31	24
[**strid**]	30	33	3	26	25	24	31	20	29	25
[**lat**]	23	26	23	26	32	23	33	27	32	31
[**dor**]	24	24	25	32	21	26	35	23	16	28
$\lvert M_F\rvert$	189	206	177	179	196	198	190	174	184	186

Table 24: Cardinalities $|M_{(\Phi,\phi)}|$ and $|M_F|$ for all *sufficient* features, across 10 runs, for VS in Yawelmani

Feature set Φ	$\lvert M_{(\Phi,\phi)}\rvert$									
	1	2	3	4	5	6	7	8	9	10
[**high**]	106	151	127	146	140	184	119	174	145	187
[**low**]	38	33	27	37	31	38	40	29	29	17
[**tense**]	143	199	165	172	183	179	96	120	149	185
[**front**]	113	130	194	141	116	159	161	107	155	188
[**back**]	169	174	157	119	180	187	200	166	140	181
[**round**]	40	23	38	31	24	30	35	16	34	28
[**long**]	354	341	335	298	236	224	358	351	332	344
[**cons**]	33	32	37	33	46	42	35	40	27	37
[**son**]	52	36	42	27	44	32	46	38	31	44
[**cont**]	38	39	22	36	33	39	25	38	37	29
[**delrel**]	40	28	24	37	29	32	26	46	29	44
[**nasal**]	41	40	23	24	28	32	46	37	33	28
[**approx**]	40	32	44	52	28	41	44	43	39	42
[**voice**]	24	37	31	45	26	22	51	20	35	36
[**lab**]	32	39	32	36	32	37	38	28	41	31
[**labdent**]	31	31	37	26	41	32	33	38	31	29
[**cor**]	153	132	117	128	179	159	97	151	154	124
[**ant**]	128	119	132	152	104	158	148	124	131	166
[**distr**]	27	45	35	22	25	33	27	44	34	24
[**strid**]	26	21	35	23	33	31	36	41	35	23
[**lat**]	23	32	35	26	25	33	39	33	33	31
[**dor**]	142	143	181	189	169	168	164	168	199	167
[**constrgl**]	128	140	137	217	132	145	159	195	133	200
[**spreadgl**]	920	943	942	967	868	910	972	994	952	1052
$\lvert M_F\rvert$	920	943	942	967	868	910	972	994	952	1052

Table 25: Cardinalities $|M_{(\Phi,\phi)}|$ and $|M_F|$ for all *sufficient* features and selected one for each *insufficient* feature, across 10 runs, for FE in Chukchi

Feature set Φ	$\lvert M_{(\Phi,\phi)}\rvert$									
	1	2	3	4	5	6	7	8	9	10
[**high**]	193	146	174	176	125	170	228	154	137	144
[**low**]	30	40	44	32	45	39	44	37	36	32
[**tense**]	32	22	39	32	32	26	29	32	41	36
[**front**]	123	172	191	183	144	206	147	193	186	114
[**back**]	190	190	188	174	160	204	134	152	194	157
[**round** high]	256	186	215	222	239	254	209	170	260	236
[**long**]	28	34	42	29	30	36	22	32	37	34
[**cons**]	37	33	31	28	31	32	32	32	19	36
[**son** low]	223	259	203	253	273	283	283	271	267	287
[**cont** tense]	277	315	317	323	317	293	313	307	275	323
[**delrel** long]	1023	933	873	923	863	853	913	963	893	1003
[**approx**]	26	32	38	45	28	27	38	38	39	32
[**nasal** son]	277	298	283	267	297	331	281	285	293	285
[**voice** cons]	277	319	259	311	295	317	323	321	311	313
[**lab**]	315	311	323	311	321	317	297	305	303	323
[**labdent**]	39	44	31	29	35	33	28	26	36	36
[**cor**]	315	285	313	267	299	311	321	307	315	321
[**ant** labdent]	873	873	893	963	993	923	1053	883	873	893
[**distr** cor]	813	813	813	784	784	784	813	813	784	813
[**strid** dor]	963	893	1013	1083	953	933	963	953	893	903
[**lat**]	279	249	257	267	287	287	285	275	275	281
[**dor**]	34	37	26	32	30	40	31	37	30	27
$\lvert M_F\rvert$	2,380	2,293	2,320	2,373	2,301	2,308	2,372	2,365	2,238	2,356

Table 26: Cardinalities $|M_{(\Phi,\phi)}|$ and $|M_F|$ for all *sufficient* features and selected one for each *insufficient* feature, across 10 runs, for FD and OR in Polish

Feature set Φ	$\lvert M_{(\Phi,\phi)}\rvert$									
	1	2	3	4	5	6	7	8	9	10
[**voice**, son]	223	201	219	196	201	217	223	217	225	221
[**son**]	39	34	26	28	45	25	9	30	33	3
[**cons**]	39	42	34	44	25	28	33	32	3	4
[**cont**]	24	25	26	7	27	32	31	25	31	35
[**cor**]	28	40	43	32	33	36	40	34	22	39
[**ant**]	165	152	126	160	170	140	168	158	187	145
[**dor**]	28	39	34	26	26	35	31	33	36	33
[**lab**]	27	16	3	36	3	31	27	41	25	44
[**distr**]	39	38	3	23	41	48	37	33	31	29
[**delrel**]	200	159	124	114	117	141	151	133	102	224
[**nasal**]	3	42	42	29	37	43	3	36	27	30
[**lat**]	33	3	46	30	47	31	3	34	36	45
[**high**, voice, son, low]	7,310	6,560	8,180	6,060	7,780	7,620	6,620	8,220	5,500	7,340
[**back**]	23	4	37	35	22	42	22	46	27	27
[**low**]	145	169	165	187	157	125	4	152	198	230
[**tense**, voice, son, round]	6,870	8,220	7,780	6,100	7,180	8,130	8,100	6,300	8,140	6,900
[**round**]	144	163	150	161	178	4	126	172	166	104
$\lvert M_F\rvert$	8,185	8,371	8,321	7,669	8,263	8,350	8,336	8,333	8,319	8,200

Table 27: Cardinalities $|S_{(\Phi,\phi)}|$ and $|M_{(\Phi,\phi)}|$ for all *sufficient* features and selected one for each *insufficient* feature, for Σ_{seg} with 8, 9 and 10 phonemes, for FD and OR in Polish

Feature set Φ	$\lvert\Sigma_{\text{seg}}\rvert = 8$		$\lvert\Sigma_{\text{seg}}\rvert = 9$		$\lvert\Sigma_{\text{seg}}\rvert = 10$	
	$\lvert S_{(\Phi,\phi)}\rvert$	$\lvert M_{(\Phi,\phi)}\rvert$	$\lvert S_{(\Phi,\phi)}\rvert$	$\lvert tM_{(\Phi,\phi)}\rvert$	$\lvert S_{(\Phi,\phi)}\rvert$	$\lvert M_{(\Phi,\phi)}\rvert$
[**voice**, son]	363	241	363	223	363	235
[**son**]	62	32	62	34	62	23
[**cons**]	62	38	62	41	62	31
[**cont**]	62	40	62	27	62	33
[**cor**]	62	7	62	34	336	4
[**ant**]	363	4	363	114	363	135
[**dor**]	62	33	62	3	62	43
[**lab**]	62	47	62	41	336	4
[**distr**]	62	34	62	25	62	29
[**delrel**]	363	141	363	4	363	5
[**nasal**]	54	43	62	29	62	43
[**lat**]	62	32	62	28	62	39
[**high**, voice, son, low]	8,446	7,416	–	–	–	–
[**high**, voice, nasal, low]	–	–	17,892	13,122	17,892	8,122
[**back**]	62	24	62	32	62	37
[**low**]	336	192	336	119	336	182
[**tense**, voice, son, round]	8,446	6,656	–	–	–	–
[**tense**, voice, nasal, round]	–	–	17,892	12,482	17,892	17,843
[**round**]	336	162	336	136	336	153
$\lvert M_F\rvert$	33,352	8,283	59,868	16,272	101,110	17,945

APPENDIX: CARDINALITIES OF ALL $|S_{(\Phi,\phi)}|$ FOR THE *insufficient* FEATURES

C

Φ	$\lvert S_{(\Phi,\phi)}\rvert$
[**nasal** cons]	39
[**nasal** approx]	155
[**nasal** lat]	155
[**nasal** lab]	155
[**nasal** labdent]	84
[**nasal** cor]	155
[**nasal** dor]	155
[**nasal** front]	84
[**nasal** back]	84
[**nasal** long]	39

Table 28:
Cardinality $|S_{(\Phi,\text{nasal})}|$ for Φ combinations with the *insufficient* feature [nasal] and various features in English

Table 29: Cardinalities $|S_{(\Phi,\phi)}|$ for all Φ combinations with insufficient features [round], [son], [cont], [delrel], and [nasal] and various features in Chukchi

[round]		**[son]**		**[cont]**		**[delrel]**		**[nasal]**	
Φ	$\lvert S_{(\Phi,\text{round})}\rvert$	Φ	$\lvert S_{(\Phi,\text{son})}\rvert$	Φ	$\lvert S_{(\Phi,\text{cont})}\rvert$	Φ	$\lvert S_{(\Phi,\text{delrel})}\rvert$	Φ	$\lvert S_{(\Phi,\text{nasal})}\rvert$
[**round** low]	309	[**son** high]	1172	[**cont** high]	1236	[**delrel** high]	3405	[**nasal** high]	1172
[**round** tense]	309	[**son** tense]	336	[**cont** low]	363	[**delrel** low]	1236	[**nasal** low]	336
[**round** front]	309	[**son** front]	1172	[**cont** front]	1236	[**delrel** tense]	1236	[**nasal** tense]	336
[**round** back]	309	[**son** back]	1172	[**cont** back]	1236	[**delrel** front]	3405	[**nasal** front]	1172
[**round** long]	309	[**son** long]	336	[**cont** long]	363	[**delrel** back]	3405	[**nasal** back]	1172
[**round** cons]	309	[**son** cons]	336	[**cont** cons]	363	[**delrel** long]	1236	[**nasal** long]	336
[**round** approx]	309	[**son** cont]	1300	[**cont** son]	1300	[**delrel** cons]	1236	[**nasal** cons]	336
[**round** lab]	1236	[**son** approx]	336	[**cont** delrel]	3530	[**delrel** approx]	1236	[**nasal** delrel]	1236
[**round** labdent]	309	[**son** nasal]	336	[**cont** approx]	363	[**delrel** nasal]	1236	[**nasal** approx]	336
[**round** cor]	1236	[**son** lab]	1300	[**cont** lab]	3530	[**delrel** lab]	3530	[**nasal** lab]	1300
[**round** lat]	1172	[**son** labdent]	336	[**cont** labdent]	363	[**delrel** labdent]	1236	[**nasal** labdent]	336
[**round** dor]	309	[**son** cor]	1300	[**cont** cor]	3530	[**delrel** cor]	3530	[**nasal** cor]	1300
		[**son** lat]	1236	[**cont** lat]	1236	[**delrel** lat]	3530	[**nasal** lat]	1236
		[**son** dor]	336	[**cont** dor]	363	[**delrel** dor]	1236	[**nasal** dor]	336

Table 30: Cardinalities $|S_{(\Phi,\phi)}|$ for all Φ combinations with insufficient features [voice], [ant], [distr], and [strid] and various features in Chukchi

[voice]		**[ant]**		**[distr]**		**[strid]**	
Φ	$\lvert S_{(\Phi,\text{voice})}\rvert$	Φ	$\lvert S_{(\Phi,\text{ant})}\rvert$	Φ	$\lvert S_{(\Phi,\text{distr})}\rvert$	Φ	$\lvert S_{(\Phi,\text{strid})}\rvert$
[**voice** high]	1236	[**ant** high]	3405	[**distr** high]	3405	[**strid** high]	3405
[**voice** low]	363	[**ant** low]	1236	[**distr** low]	1236	[**strid** low]	1236
[**voice** tense]	363	[**ant** tense]	1236	[**distr** tense]	1236	[**strid** tense]	1236
[**voice** front]	1236	[**ant** front]	3405	[**distr** front]	3405	[**strid** front]	3405
[**voice** back]	1236	[**ant** back]	3405	[**distr** back]	3405	[**strid** back]	3405
[**voice** approx]	363	[**ant** long]	1236	[**distr** long]	1236	[**strid** long]	1236
[**voice** lab]	1300	[**ant** cons]	1236	[**distr** cons]	1236	[**strid** cons]	1236
[**voice** labdent]	363	[**ant** approx]	1236	[**distr** approx]	1236	[**strid** approx]	1236
[**voice** cor]	1300	[**ant** lab]	1236	[**distr** lab]	1236	[**strid** lab]	1236
[**voice** lat]	1300	[**ant** cor]	1236	[**distr** labdent]	1236	[**strid** labdent]	1236
[**voice** dor]	363	[**ant** lat]	1236	[**distr** lat]	1236	[**strid** lat]	1236
		[**ant** dor]	1236	[**distr** dor]	1236	[**strid** cor]	1236

Table 31:
Cardinalities $|S_{(\Phi,\phi)}|$ for all Φ combinations with insufficient features [voice], [high], [tense] and various features in Polish

Φ	Cardinality
[**voice**, son]	336
[**voice**, delrel]	1172
[**high**, voice, **son**, **round**]	8250
[**high**, voice, **son**, **low**]	8250
[**tense**, voice, **son**, **low**]	8250
[**tense**, voice, **son**, **round**]	8250

REFERENCES

Adam ALBRIGHT and Bruce HAYES (2002), Modeling English past tense intuitions with minimal generalization, in *Proceedings of the Sixth Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON 6)*, pp. 58–69, Association for Computational Linguistics, Philadelphia, PA.

Adam ALBRIGHT and Bruce HAYES (2003), Rules vs. analogy in English past tenses: A computational/experimental study, *Cognition*, 90:119–161.

Diana ARCHANGELI (1991), Syllabification and prosodic templates in Yawelmani, *Natural Language & Linguistic Theory*, 9(2):231–283.

Peter AVERY and William IDSARDI (2001), Laryngeal dimensions, completion and enhancement, in T. A. HALL and U. KLEINHANZ, editors, *Studies in distinctive feature theory*, pp. 41–70, Mouton de Gruyter.

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

Eric BAKOVIĆ and Lev BLUMENFELD (2024), A formal typology of process interactions, *Phonological Data and Analysis*, 6(3):1–43, doi:10.3765/pda.v6art3.83, `https://phondata.org/index.php/pda/article/view/83`.

Alan BALE and Charles REISS (2018), *Phonology: A formal introduction*, The MIT Press.

Kenneth R. BEESLEY and Lauri KARTTUNEN (2003), *Finite state morphology*, CSLI Publications.

Gasper BEGUS (2020), Modeling unsupervised phonetic and phonological learning in generative adversarial phonology, in Allyson ETTINGER, Gaja JAROSZ, and Joe PATER, editors, *Proceedings of the Society for Computation in Linguistics 2020*, pp. 38–48, Association for Computational Linguistics, New York, New York, `https://aclanthology.org/2020.scil-1.5/`.

Caleb BELTH (2023), A learning-based account of local phonological processes, *Phonology*, 40(1–2):1–33.

Caleb BELTH (2024), A learning-based account of phonological tiers, *Linguistic Inquiry*, pp. 1–63.

Christina BETHIN (1978), *Phonological rules in the nominative singular and the genitive plural of the Slavic substantive declension*, Ph.D. thesis, University of Illinois at Urbana-Champaign.

Siddharth BHASKAR, Jane CHANDLEE, Adam JARDINE, and Christopher OAKDEN (2020), Boolean monadic recursive schemes as a logical characterization of the subsequential functions, in Alberto LEPORATI, Carlos

Martín-Vide, Dana Shapira, and Claudio Zandron, editors, *Language and Automata Theory and Applications – LATA 2020*, Lecture Notes in Computer Science, pp. 157–169, Springer.

Juliette Blevins (2004), A reconsideration of Yokuts vowels, *International Journal of American Linguistics*, 70(1):33–51, doi:10.1086/425845.

Eugene Buckley (2001), Polish o-raising and phonological explanation, manuscript, University of Pennsylvania.

Phillip Burness and Kevin McMullin (2019), Efficient learning of output tier-based strictly 2-local functions, in Philippe de Groote, Frank Drewes, and Gerald Penn, editors, *Proceedings of the 16th Meeting on the Mathematics of Language*, pp. 78–90, Association for Computational Linguistics, Toronto, Canada, doi:10.18653/v1/W19-5707, https://aclanthology.org/W19-5707.

Phillip Burness and Kevin McMullin (2020), Multi-tiered strictly local functions, in Garrett Nicolai, Kyle Gorman, and Ryan Cotterell, editors, *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pp. 245–255, Association for Computational Linguistics, Online, doi:10.18653/v1/2020.sigmorphon-1.29, https://aclanthology.org/2020.sigmorphon-1.29/.

Phillip Alexander Burness, Kevin James McMullin, and Jane Chandlee (2021), Long-distance phonological processes as tier-based strictly local functions, *Glossa*, 6(1):1–37.

Jane Chandlee (2014), *Strictly local phonological processes*, Ph.D. thesis, University of Delaware.

Jane Chandlee (2017), Computational locality in morphological maps, *Morphology*, 27(4):599–641.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz (2014), Learning strictly local subsequential functions, *Transactions of the Association for Computational Linguistics*, 2:491–504, doi:10.1162/tacl_a_00198, https://aclanthology.org/Q14-1038.

Jane Chandlee, Remi Eyraud, Jeffrey Heinz, Adam Jardine, and Jonathan Rawski (2019), Learning with partially ordered representations, in Philippe de Groote, Frank Drewes, and Gerald Penn, editors, *Proceedings of the 16th Meeting on the Mathematics of Language*, pp. 91–101, Association for Computational Linguistics, Toronto, Canada, doi:10.18653/v1/W19-5708, https://aclanthology.org/W19-5708.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz (2015), Output strictly local functions, in *Proceedings of the 14th Meeting on the Mathematics of Language (MoL 2015)*, pp. 112–125, Association for Computational Linguistics, Chicago, USA.

Jane Chandlee and Jeffrey Heinz (2018), Strict locality and phonological maps, *Linguistic Inquiry*, 49(1):23–60.

Jane Chandlee, Jeffrey Heinz, and Adam Jardine (2018), Input strictly local opaque maps, *Phonology*, 35(2):171–205.

Jane Chandlee and Adam Jardine (2021), Computational universals in linguistic theory: Using recursive programs for phonological analysis, *Language*, 93:485–519.

Marilyn Y. Chen (1997), Acoustic correlates of English and French nasalized vowels, *Journal of the Acoustical Society of America*, 102(4):2360–2370.

Christian Choffrut (1977), Une caractérisation des fonctions séquentielles et des fonctions sous-séquentielles en tant que relations rationnelles, *Theoretical Computer Science*, 5(3):325–337, ISSN 0304-3975, doi:https://doi.org/10.1016/0304-3975(77)90049-4, http://www.sciencedirect.com/science/article/pii/0304397577900494.

Christian Choffrut (2003), Minimizing subsequential transducers: A survey, *Theoretical Computer Science*, 292(1):131–143, doi:http://dx.doi.org/10.1016/S0304-3975(01)00219-5.

Noam Chomsky and Morris Halle (1965), Some controversial questions in phonological theory, *Journal of Linguistics*, 1:97–138.

Noam Chomsky and Morris Halle (1968), *The sound pattern of English*, Harper & Row.

George. N. Clements (1985), The geometry of phonological features, *Phonology Yearbook*, 2:225–252.

George. N. Clements (2009), The role of features in phonological inventories, in Eric Raimy and Charles E. Cairns, editors, *Contemporary views on architecture and representations in phonology*, pp. 19–68, MIT Press, doi:10.7551/mitpress/9780262182706.003.0002.

George N. Clements and Elizabeth V. Hume (1995), The internal organization of speech sounds, in John Goldsmith, editor, *The Handbook of Phonological Theory*, pp. 245–306, Blackwell.

Abigail C. Cohn (1993), Nasalisation in English: Phonology or phonetics, *Phonology*, 10:43–81.

Abigail C. Cohn (2011), Features, segments, and the sources of phonological primitives, https://api.semanticscholar.org/CorpusID:13303504.

Colin de la Higuera (2010), *Grammatical inference: Learning automata and grammars*, Cambridge University Press.

B. Elan Dresher (2009), *The contrastive hierarchy in phonology*, number 121 in Cambridge Studies in Linguistics, Cambridge University Press.

Naiyan Du and Karthik Durvasula (2025), *Psycholinguistics and phonology: The forgotten foundations of generative phonology*, Cambridge University Press, doi:https://doi.org/10.1017/9781009347631.

San Duanmu (2016), *A theory of phonological features*, Oxford University Press.

Michael Dunn (1999), *A grammar of Chukchi*, Ph.D. thesis, Australian National University.

Jerry A. Fodor (1980), Fixation of belief and concept acquisition, in Massimo Piattelli-Palmarini, editor, *Language and learning: The debate between Jean Piaget and Noam Chomsky*, pp. 142–162, Harvard University Press.

Carol A. Fowler and Julie M. Brown (2000), Perceptual parsing of acoustic consequences of velum lowering from information for vowels, *Perception & Psychophysics*, 62(1):21–32.

Victoria A. Fromkin (1971), The non-anomalous nature of anomalous utterances, *Language*, 47(1):27–52.

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(5):447–474.

Edmund Gussmann (2007), *The phonology of Polish*, Oxford University Press.

Bruce Hayes (2009), *Introductory phonology*, Wiley-Blackwell.

Bruce Hayes and Colin Wilson (2008), A maximum entropy model of phonotactics and phonotactic learning, *Linguistic Inquiry*, 39:379–440.

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

Jeffrey Heinz (2010b), String extension learning, in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 897–906, Association for Computational Linguistics, Uppsala, Sweden.

Jeffrey Heinz (2011a), Computational phonology part I: Foundations, *Language and Linguistics Compass*, 5(4):140–152.

Jeffrey Heinz (2011b), Computational phonology part II: Grammars, learning, and the future, *Language and Linguistics Compass*, 5(4):153–168.

Jeffrey Heinz (2018), The computational nature of phonological generalizations, in Larry Hyman and Frans Plank, editors, *Phonological typology*, Phonetics and Phonology, chapter 5, pp. 126–195, De Gruyter Mouton.

Jeffrey Heinz (2025), The logical structure of phonological generalizations, *Phonological Studies*, 28:69–80.

Jeffrey Heinz, Colin de la Higuera, and Menno van Zaanen (2015), *Grammatical inference for computational linguistics*, Synthesis Lectures on Human Language Technologies, Morgan and Claypool.

Jeffrey Heinz, Anna Kasprzik, and Timo Kötzing (2012), Learning with lattice-structured hypothesis spaces, *Theoretical Computer Science*, 457:111–127.

Jeffrey Heinz and Regine Lai (2013), Vowel harmony and subsequentiality, in András Kornai and Marco Kuhlmann, editors, *Proceedings of the 13th Meeting on Mathematics of Language*, Sofia, Bulgaria.

Jeffrey Heinz, Chetan Rawal, and Herbert G. Tanner (2011), Tier-based strictly local constraints for phonology, in Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 58–64, Association for Computational Linguistics, Portland, Oregon, USA, https://aclanthology.org/P11-2011/.

Jeffrey Heinz and José Sempere, editors (2016), *Topics in grammatical inference*, Springer-Verlag.

Charles F. Hockett (1967), The Yawelmani basic verb, *Language*, 43(1):208–222.

Larry Hyman (1975), *Phonology: Theory and analysis*, Holt, Rinehart and Winston.

Roman Jakobson, Gunnar Fant, and Morris Halle (1951), *Preliminaries to speech analysis*, MIT Press.

Adam Jardine (2016), Computationally, tone is different, *Phonology*, 33(2):247–283, doi:10.1017/S0952675716000129.

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

Adam Jardine and Jeffrey Heinz (2016), Learning tier-based strictly 2-local languages, *Transactions of the Association for Computational Linguistics*, 4:87–98, ISSN 2307-387X, doi:10.1162/tacl_a_00085, https://doi.org/10.1162/tacl_a_00085.

Adam Jardine and Kevin McMullin (2017), Efficient learning of tier-based strictly k-local languages, in Frank Drewes, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications, 11th International Conference*, Lecture Notes in Computer Science, pp. 64–76, Springer.

C. Douglas Johnson (1972), *Formal aspects of phonological description*, Mouton.

René Kager (1999), *Optimality theory*, Cambridge University Press.

Ronald M. Kaplan and Martin Kay (1994), Regular models of phonological rule systems, *Computational Linguistics*, 20:331–378.

Jonathan KAYE (1980), The mystery of the tenth vowel, *Journal of Linguistic research*, 1:1–14.

Michael KENSTOWICZ (1994), *Phonology in generative grammar*, Blackwell Publishing.

Michael KENSTOWICZ and Charles KISSEBERTH (1979), *Generative phonology: Description and theory*, Academic Press.

Paul KIPARSKY (1973), Abstractness, opacity and global rules, in O. FUJIMURA, editor, *Three dimensions of linguistic theory*, pp. 57–86, Tokyo: TEC, part 2 of "Phonological representations".

Rena A. KRAKOW (1993), Nonsegmental influences on velum movement patterns: Syllables, sentences, stress, and speaking rate, in Marie K. HUFFMAN and Rena A. KRAKOW, editors, *Nasals, nasalization, and the velum*, pp. 87–116, Academic Press.

Rena A. KRAKOW and Michael K. HUFFMAN (1989), Temporal coordination in speech production: The nasalization of vowels in English, *Journal of Phonetics*, 17:277–284.

Scott Russell KRAUSE (1980), *Topics in Chukchee phonology and morphology*, Ph.D. thesis, University of Illinois at Urbana-Champaign.

Martin KRÄMER (2015), *The phonology of vowel harmony*, Cambridge University Press.

Dakotah LAMBERT (2022), *Unifying classification schemes for languages and processes with attention to locality and relativizations thereof*, Ph.D. thesis, Stony Brook University, `https://vvulpes0.github.io/PDF/dissertation.pdf`.

Dakotah LAMBERT (2023), Relativized adjacency, *Journal of Logic Language and Information*, 32:707–731, doi:https://doi.org/10.1007/s10849-023-09398-x.

Dakotah LAMBERT (to appear), Multitier phonotactics, *Phonology*.

Dakotah LAMBERT and Jeffrey HEINZ (2023), An algebraic characterization of total input strictly local functions, in *Proceedings of the Society for Computation in Linguistics*, volume 6, pp. 25–34, doi:https://doi.org/10.7275/Q54B-MG07.

Dakotah LAMBERT and Jeffrey HEINZ (2024), Algebraic reanalysis of phonological processes described as output-oriented, in *Proceedings of the Society for Computation in Linguistics*, volume 7, pp. 129–138, doi:https://doi.org/10.7275/scil.2137.

Dakotah LAMBERT, Jonathan RAWSKI, and Jeffrey HEINZ (2021), Typology emerges from simplicity in representations and learning, *Journal of Language Modelling*, 9(1):151–194.

Dakotah LAMBERT and James ROGERS (2020), Tier-based strictly local stringsets: Perspectives from model and automata theory, in *Proceedings of the Society for Computation in Linguistics*, volume 3, pp. 330–337, New Orleans, Louisiana.

Andrew LAMONT (2021), Optimizing over subsequences generates context-sensitive languages, *Transactions of the Association for Computational Linguistics*, 9:528–537.

Andrew LAMONT (2022), Optimality theory implements complex functions with simple constraints, *Phonology*, 38(4):729–740.

Han LI (2025), Learning tonotactic patterns over autosegmental representations, *Proceedings of the Annual Meetings on Phonology*, 1(1), doi:10.7275/amphonology.3034, `https://doi.org/10.7275/amphonology.3034`.

Constantine LIGNOS and Charles YANG (2016), *Morphology and language acquisition*, p. 765–791, Cambridge Handbooks in Language and Linguistics, Cambridge University Press, doi:10.1017/9781139814720.027.

Huan LUO (2017), Long-distance consonant agreement and subsequentiality, *Glossa*, 2(1):52.

Magdalena MARKOWSKA and Jeffrey HEINZ (2023), Empirical and theoretical arguments for using properties of letters for the learning of sequential functions, in François COSTE, Faissal OUARDI, and Guillaume RABUSSEAU, editors, *Proceedings of 16th edition of the International Conference on Grammatical Inference*, volume 217 of *Proceedings of Machine Learning Research*, pp. 270–274.

Connor MAYER (2020), An algorithm for learning phonological classes from distributional similarity, *Phonology*, 37:91–131, doi:10.1017/S0952675720000056.

Adam G. MCCOLLUM, Eric BAKOVIĆ, Anna MAI, and Eric MEINHARDT (2020), Unbounded circumambient patterns in segmental phonology, *Phonology*, 37(2):215–255, doi:10.1017/S095267572000010X.

Kevin MCMULLIN (2016), *Tier-based locality in long-distance phonotactics: Learnability and typology*, Ph.D. thesis, University of British Columbia.

Jeff MIELKE (2008), *The emergence of distinctive features*, Oxford University Press.

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

Arijit NAG, Soumen CHAKRABARTI, Animesh MUKHERJEE, and Niloy GANGULY (2024), Efficient continual pre-training of LLMs for low-resource languages, *arXiv preprint arXiv:2412.10244*.

Jose ONCINA and Pedro GARCIA (1991), Inductive learning of subsequential functions, Technical Report DSIC II-34, University Politécnia de Valencia.

José ONCINA, Pedro GARCÍA, and Enrique VIDAL (1993), Learning subsequential transducers for pattern recognition tasks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:448–458.

Amanda PAYNE (2017), All dissimilation is computationally subsequential, *Language*, 4(3):e353–e371.

Brandon PRICKETT (2021), *Learning phonology with sequence-to-sequence neural networks*, Ph.D. thesis, University of Massachusetts.

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

Alan PRINCE and Bruce TESAR (2004), Fixing priorities: Constraints in phonological acquisition, in *Fixing priorities: Constraints in phonological acquisition*, Cambridge University Press.

Michael PROCTOR, Louis GOLDSTEIN, Adam LAMMERT, Dani BYRD, Asterios TOUTIOS, and Shrikanth NARAYANAN (2013), Velic coordination in French nasals: A real-time magnetic resonance imaging study, in *Proceedings of Interspeech*, pp. 577–581.

Jonathan RAWSKI (2021), *Structure and learning in natural language*, Ph.D. thesis, Stony Brook University.

Charles REISS (2012), Towards a bottom-up approach to phonological typology, in A. DI SCIULLO, editor, *Towards a biolinguistic understanding of grammar*, pp. 169–191, John Benjamins.

Charles REISS and Veno VOLENEC (2022), Conquer primal fear: Phonological features are innate and substance free, *Canadian Journal of Linguistics*, 67(4):581–610.

James ROGERS, Jeffrey HEINZ, Margaret FERO, Jeremy HURST, Dakotah LAMBERT, and Sean WIBEL (2013), Cognitive and sub-regular complexity, in *Formal Grammar*, number 8036 in Lecture Notes in Computer Science, pp. 90–108, Springer.

James ROGERS and Geoffrey K. PULLUM (2011), Aural pattern recognition experiments and the subregular hierarchy, in Christian RETORÉ, editor, *Logical aspects of computational linguistics*, number 6735 in Lecture Notes in Computer Science, pp. 188–206, Springer, doi:10.1007/978-3-642-21254-3_11.

Jerzy RUBACH (2019), Surface velar palatalization in Polish, *Natural Language & Linguistic Theory*, 37:1421–1462, doi:10.1007/s11049-019-09443-8.

Jacques SAKAROVITCH (2009), *Elements of automata theory*, Cambridge University Press, translated by Reuben Thomas from the 2003 edition published by Vuibert, Paris.

Elisabeth O. SELKIRK (1972), *The phrase phonology of English and French*, Ph.D. thesis, MIT.

Maria-Josep SOLÉ (1995), Spatio-temporal patterns of velo-pharyngeal action in phonetic and phonological nasalization, *Language and Speech*, 38(1):1–23.

Kristina Strother-Garcia, Jeffrey Heinz, and Hyun Jin Hwangbo (2016), Using model theory for grammatical inference: A case study from phonology, in Sicco Verwer, Menno van Zaanen, and Rick Smetsers, editors, *Proceedings of The 13th International Conference on Grammatical Inference*, number 57 in Proceedings of Machine Learning Research, pp. 66–78.

Logan Swanson, Jeffrey Heinz, and Jon Rawski (to appear), Phonotactic learning and constraint selection without statistics, *Linguistic Inquiry*.

Bruce Tesar and Paul Smolensky (1998), Learnability in optimality theory, *Linguistic Inquiry*, 29(2):229–268.

Bruce Tesar and Paul Smolensky (2000), *Learnability in optimality theory*, MIT Press.

Nikolai Sergeevich Trubetzkoy (1969), *Principles of phonology*, University of California Press, Berkeley, translation of *Grundzüge der Phonologie*.

Mai Ha Vu, Ashkan Zehfroosh, Kristina Strother-Garcia, Michael Sebok, Jeffrey Heinz, and Herbert G. Tanner (2018), Statistical relational learning with unconventional string models, *Frontiers in Robotics and AI*, 5(76):1–26, doi:https://doi.org/10.3389/frobt.2018.00076.

Richard Wicentowski (2002), *Modeling and learning multilingual inflectional morphology in a minimally supervised framework*, Ph.D. thesis, Johns Hopkins University, Baltimore, MD.

Wojciech Wieczorek (2017), *Grammatical inference: Algorithms, routines and applications*, Springer.

Adam Wiemerslage, Aaron Mueller, Garrett Nicolai, Miikka Silfverberg, and David Yarowsky (2022), Morphological processing of low-resource languages: Where we are and what's next, *arXiv preprint arXiv:2203.08909*.

Charles Yang (2013), Who's afraid of George Kingsley Zipf? or: Do children and chimps have language?, *Significance*, 10(6):29–34, doi:10.1111/j.1740-9713.2013.00708.x.

Charles Yang (2016), *The price of productivity*, MIT Press.

Tatevik Yolyan (2025a), A framework for learning phonological maps as logical transductions, in *Proceedings of the 18th Meeting of the Mathematics of Language*, forthcoming.

Tatevik Yolyan (2025b), *Phonological expressivity and learning via boolean monadic recursive schemes*, Ph.D. thesis, Rutgers University.

Elizabeth Zsiga (2013), *The sounds of language: An introduction to phonetics and phonology*, Wiley-Blackwell.

Magdalena Markowska
0009-0004-7692-1583
magdalena.markowska@stonybrook.edu
Stony Brook University Institute for Advanced Computational Science
Stony Brook, NY, USA

Jeffrey Heinz
0000-0002-5954-3195
jeffrey.heinz@stonybrook.edu
Stony Brook University Institute for Advanced Computational Science
Stony Brook, NY, USA

Magdalena Markowska and Jeffrey Heinz (2026), Learning k-ISL functions with phonological features, *Journal of Language Modelling*, 14(1):1–80
https://dx.doi.org/10.15398/jlm.v14i1.493