Input Strictly Local Opaque Maps

Jane Chandlee, Adam Jardine, Jeffrey Heinz

to appear in *Phonology*

September 21, 2017

Abstract

This paper gives a computational characterization of opaque interactions in phonology. Specifically, a range of opaque interactions are shown to be Input Strictly Local (ISL) maps (Chandlee, 2014), which are string-to-string functions that determine output based only on contiguous sequences of input symbols. Examples from Baković (2007)'s extended typology of counterfeeding, counterbleeding, self-destructive feeding, non-gratuitous feeding, and cross-derivational feeding, as well as a case of fed counterfeeding from Kavitskaya and Staroverov (2010), are all given ISL analyses, showing that these interactions can be computed based on sequences of bounded length in the input. It is discussed how ISL maps are restrictive in their typological predictions, have guaranteed learning results, and have known methods for generation and recognition, and thus compare favorably to rule-based and constraint-based approaches to these interactions.

1 Introduction

Opacity has long been a topic of interest in phonology (Kiparsky, 1971, 1973), but it took on new importance with the shift from rule-based to constraint-based theories of the phonological grammar. The inability of classic Optimality Theory (Prince and Smolensky, 1993; Kager, 1999; Prince and Smolensky, 2004) to straightforwardly account for cases traditionally analyzed with ordered rules (Idsardi, 1998, 2000) led to a range of proposals for how to model seemingly non-optimal cases within the general OT framework. Such proposals include correspondence theory (McCarthy and Prince, 1996), output-output faithfulness (Benua, 1997), sympathy (McCarthy, 1999), and candidate chains (McCarthy, 2007), among others. An alternative approach has been to call into question the psychological reality of opaque generalizations (Sanders, 2003; Kawahara, 2015), under the assumption that if they are not sychronically productive then the theory need not account for them anyway. In other words, the lack of a mechanism for accounting for opaque interactions would be a feature, not a shortcoming, of a theory of the synchronic phonological grammar. Whether or not opacity is 'real' is an empirical question, one we do not aim to settle here. Readers interested in this question may consult McCarthy

(2007) and references therein for a review of the evidence against denying opacity as a distinct and psychologically real phenomenon.

Opacity is of interest from a computational perspective because of the complications it has posed for modern theories of generative phonology and because of claims that it is more 'complex' or 'difficult to learn' (as claimed in Kiparsky, 1971, 1973). Thus, in this paper we investigate the computational nature of opacity and show that in fact opaque interactions are computationally just as simple as single processes.

Specifically, we study the typology of opaque generalizations provided by Baković (2007) in the light of Chandlee's (2014) theory of local, subregular functions. We show that the variety of opaque generalizations discussed by Baković, and an additional case by Kavitskaya and Staroverov (2010), have the property of being Input Strictly Local (ISL), the definition of which is given in §2. Baković's typology includes a variety of opaque cases, including ones analyzed as counterfeeding (both on-environment and on-focus), counterbleeding, self-destructive feeding, non-gratuitous feeding, and cross-derivational feeding. Kavitskaya and Staroverov (2010) call their case fed counterfeeding. The fact that all such generalizations are ISL functions shows they are not necessarily more complex than single processes, and also comes with strong learnability results—the ISL class is known to be efficiently learnable from positive data (Chandlee et al., 2014; Jardine et al., 2014), in the sense of de la Higuera (1997) (a paradigm which can be thought of as adding efficiency guarantees to the paradigm introduced in Gold (1967)).

It should be noted that in the spirit of inclusiveness, we do not attempt to define opacity formally, as any such definition will likely exclude one or more of the above generalizations that others have called "opaque."[1] In addition, it is not clear how to construct a formal definition in a theory-neutral way—in fact, Baković (2007) notes that cross-derivational feeding in Lithuanian may or may not be opaque depending on how one views the generalization (for more on this see §3.2). Without such a definition, we cannot establish a formal, provable relationship between opaque maps and ISL ones, and that is not the point of the paper. Rather, our goal is to demonstrate that the various types of interactions researchers have identified as opaque all share the computational property of being ISL. In other words, opacity does not necessarily disrupt ISL-ness.

In this paper, the only assumption we make is that phonological grammars relate underlying representations to surface representations. Following Tesar (2014), we use the term 'map' to describe this phonological transformation.[2] We then establish that, in terms of a well-defined computational property, a variety of specific reported instances of opacity are no different from single processes (i.e., maps without interactions).

Because our analyses focus on the computational nature of the maps themselves, the analyses are *independent* of traditional phonological formalisms. This approach has precedent in the literature (see Baković, 2013; Tesar, 2014). This focus on the maps themselves means the conclusions we draw are not about the nature of phonological rules or phonological constraint-based grammars, but of the input-output map that all such

[1]One potential candidate is Tesar (2014)'s notion of output-drivenness, but Tesar himself rejects this, saying that "the two properties are fundamentally distinct" (p. 4).

[2]Throughout the paper we will use the more general terms 'input' and 'output' instead of UR and SR, since individual phonological maps based on rule representations are embedded in a larger grammar such that the input/output of a given map may not correspond to the UR/SR.

formalisms aim to describe. This focus notwithstanding, there are significant ramifications for theories of phonology, which we discuss in §4. There we argue that these results make the case that a theory of phonological grammar which embraces subregular properties like ISL has more support from generation, recognition, typology and learnability than a theory which embraces optimization.

This paper is organized as follows. Section 2 presents the needed background for the analyses of opaque interactions presented in §3. In §4 we compare our approach to other types of phonological theories (i.e., rule- and constraint-based) before concluding in §5.

2 Background

The foundational premise to the computational analyses presented in this paper is that phonological grammars posit transformations: they ultimately describe a function or map that takes URs as inputs and computes corresponding SRs as output.

2.1 Intensional and extensional descriptions

It will be helpful to make clear a distinction between grammars, on the one hand, and the objects grammars describe on the other. Grammars are *intensional* descriptions of objects, and the objects themselves are *extensional*. The objects may be infinite in size, but the intensional descriptions must be finite in size. These distinctions are common in mathematics. Lines on a cartesian plane are sets of infinitely many points and have many intensional descriptions. For example, they can be described with an equation of finite size of the form $y = ax + b$; they can also be described with any two distinct points on the line such as (x, y) and (x', y'). However the equations and points are not the lines themselves, but descriptions of them. They are *intensional* descriptions (grammars), while the infinite set of points is the *extension* of these descriptions. While these are familiar concepts in mathematics and perhaps hardly worth mentioning, it is important to keep this distinction in the forefront of one's mind when thinking about grammars, since the distinction is equally valid. What is the extension that a grammar describes?

To make the point concrete in phonology, consider a language where word final vowels delete. There are different grammars which can describe this fact. One could write the SPE-style rewrite rule shown in (1):

(1) $\text{V} \rightarrow \varnothing$ / __ #

In an OT grammar one might (in part) employ the ranking shown below in (2):

(2) Final-C $\gg$ Max-V

What are the extensions of these grammars? We would argue that, like lines on cartesian planes, the extension can be understood as an infinite set of points, with the left coordinate being the input form and the right coordinate being the output form. This infinite set of points includes pairs like (kapa, kap), (tiki, tik), and (plabo, plab). We assume this extension also includes points like (rek, rek), (malarakalark, malarakalark), and (b,b).

This is consonant with the principle of the rich base (Prince and Smolensky, 2004) and that when a rule does not apply its output is the same as its input. We can now ask what *kind* of set of points is this? More specifically, what computational properties explain the nature of the sets of points that represent phonological maps? In the next section we define one such property: input strict locality.

2.2 Input Strictly Local functions

We will first define ISL functions through a metaphor, and then we'll provide a more precise, formal characterization. First, the metaphor. You are a computer asked to return the surface representation y of some underlying representation x. You begin this process by reading from the beginning of x and moving through the string left to right. As you read x, and perhaps before you finish reading x, you begin writing the output y also from left to right. The key idea behind Input Strictly Local functions is this: there is some positive integer k such that what you write at any given point in this process only depends on the last k symbols you read in x. Figure 1 illustrates this point with $k = 2$.

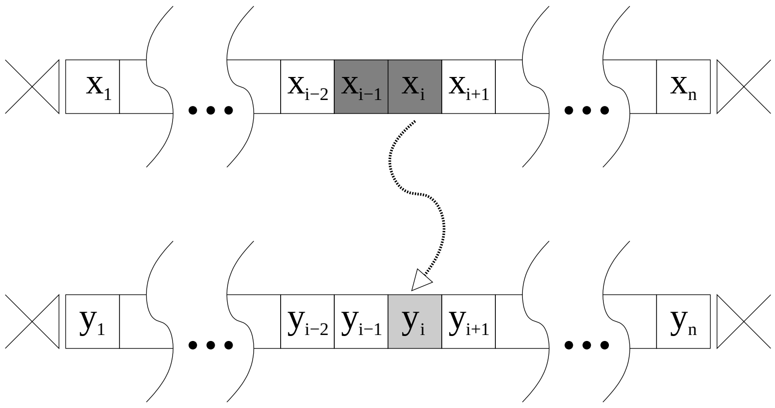


Figure 1: A schematic illustrating the Markovian nature of Input Strictly k-Local functions. For every Input Strictly 2-Local function, the output string y_i of each input element x_i depends only on x_i and x_{i-1}. In other words, the lightly shaded cell only depends on the darkly shaded cells.

The metaphor of reading and writing left to right and paying attention to only the most recent k symbols in the input is useful, but it is inaccurate in one important way. As it turns out, it does not matter if the string is read left-to-right or right-to-left. All that matters is that if the input x is n symbols long (so $x = x_1x_2 \dots x_n$ with each x_i a letter), we can think of the output y as the concatenation of n strings (so $y = y_1y_2 \dots y_n$ with each y_i a string) where each y_i only depends on the string $x_{i-k+1} \dots x_i$ (which is of length k). The order in which each y_i is computed does not really matter.

One may think that ISL functions can only describe surface representations at least as long as the underlying representations they are computed from because there are n-many y_i strings. However, do not forget that y_i can also be the *empty* string, which is the unique string of length zero (which we denote λ). Not only does this make ISL functions able to model deletion processes, it also allows them to describe phonological processes with both left and right contexts. Examples like this are provided later in the paper.

2.3 Finite-state characterization of ISL functions

ISL functions have been characterized in terms of automata theory, formal language theory (Chandlee, 2014; Chandlee et al., 2014), and first order logic (Chandlee and Lindell, to appear). Here we provide the finite-state transducer (FST) characterization. The original definition, which provides an abstract, grammar-independent characterization, is provided in an appendix.

What is a FST? Figure 2 shows a FST for the example of word final deletion discussed earlier. A FST includes a finite set of states (shown as circles), with one state being designated the start state (the one with the unlabeled incoming arrow) and some set of states designated as final states (circles with double perimeters). It also includes a set of transitions between states that determine 1) how the state changes when reading some input letter and 2) what output is written; transitions are shown as arrows between states with the label $a : w$ meaning letter a is read and string w is written.

Figure 2 is a FST whose extension is a map for word final vowel deletion. An input x is mapped to an output y only if there is a path from state 1 (the start state) to state 2 (in this FST the only final state) which reads a sequence corresponding to x and writes a sequence corresponding to y. In this FST, C and V serve as variables such that any input segment in the class of consonants/vowels can follow a transition marked C/V.

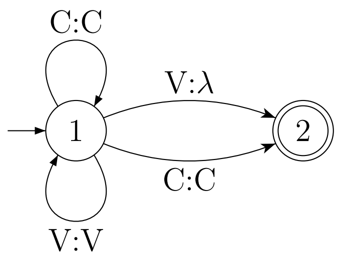


Figure 2: FST for final vowel deletion

For example, the input string CVC is mapped to CVC because the following path in (3) is a valid path in the FST (i.e., starts in the start state and ends in a final state).

(3)
$$\begin{array}{rccccccc}
\text{Input:} & & \text{C} & & \text{V} & & \text{C} & \\
\text{States:} & 1 & \longrightarrow & 1 & \longrightarrow & 1 & \longrightarrow & 2 \\
\text{Output:} & & \text{C} & & \text{V} & & \text{C} &
\end{array}$$

And here is a path which computes the output of CVCV.

(4)
$$\begin{array}{rccccccccc}
\text{Input:} & & \text{C} & & \text{V} & & \text{C} & & \text{V} & \\
\text{States:} & 1 & \longrightarrow & 1 & \longrightarrow & 1 & \longrightarrow & 1 & \longrightarrow & 2 \\
\text{Output:} & & \text{C} & & \text{V} & & \text{C} & & \lambda &
\end{array}$$

Note that any path which ends in state 1 is not valid. Also, note that an input like CVV will map to CV so only one final vowel is deleted. Thus, this map is identical to the extension of the SPE-style rule in (1) applying simultaneously, and it is not identical to the extension of the OT grammar in (2), which deletes all final vowels. To make an FST which deletes all final vowels one would have to add a transition from state 2 to itself with the label V:λ. This FST would now match the extension of the OT grammar in (2), as well as the extension of the SPE-style rule (1) applying right-to-left or persistently.

One kind of FST is necessary and sufficient to describe ISL functions. Figure 3 shows an ISL FST with the same extension as the FST in Figure 2. Recall that a map is ISL if there is a positive integer k such that the i^{th} part of the output string depends only on the last k letters up to and including the i^{th} symbol read. This is encoded in a FST in the states: basically, each state corresponds to the last $k-1$ letters read in the input, which means there must be a state for each $k-1$ sequence of input that *could* be read.[3] This in turn means the ISL FST will have more states than the non-ISL version, so for readability the FST in Figure 3 assumes a reduced alphabet with one consonant and one vowel: {t, a}.[4]

Instead of indexing states with numbers as we did in Figure 2, it makes sense to label states with the sequence they correspond to. Another difference is that both the input and the output strings explicitly encode beginning and end markers, using the symbols $\rtimes$ and $\ltimes$, respectively.[5] In Figure 3, k equals 2, so five states are needed to keep track of possible input sequences up to length $k-1$ or 1. These states are λ, t, a, $\rtimes$, and $\ltimes$.

The transitions are organized in a very simple, particular way. Basically, if the FST is in a state corresponding to the sequence t and the letter a is read then the FST transitions to the state which corresponds to the last $k-1$ letters of ta. When $k=2$ as in Figure 3, this means whenever a is read the FST transitions to state a. Whenever t is read it transitions to state t. There is only one final state (labeled $\ltimes$) and every transition which reads the end marker $\ltimes$ goes to this final state.

To see how this FST works, the valid path in (5) shows the mapping of *tat* to *tat*.

(5)
$$\begin{array}{lcccccccccc} \text{Input:} & & \rtimes & & t & & a & & t & & \ltimes \\ \text{States:} & \lambda & \longrightarrow & \rtimes & \longrightarrow & \text{t} & \longrightarrow & \text{a} & \longrightarrow & \text{t} & \longrightarrow & \ltimes \\ \text{Output:} & & \rtimes & & t & & \lambda & & at & & \ltimes \end{array}$$

The valid path in (6) shows how the FST maps *tata* to *tat*.

(6)
$$\begin{array}{lcccccccccccc} \text{Input:} & & \rtimes & & t & & a & & t & & a & & \ltimes \\ \text{States:} & \lambda & \longrightarrow & \rtimes & \longrightarrow & \text{t} & \longrightarrow & \text{a} & \longrightarrow & \text{t} & \longrightarrow & \text{a} & \longrightarrow & \ltimes \\ \text{Output:} & & \rtimes & & t & & \lambda & & at & & \lambda & & \ltimes \end{array}$$

[3]If the input read so far is less than k letters then the state just represents the string read so far (there is an exception involving the final state, discussed below).

[4]A reviewer questions whether an ISL FST with an alphabet that includes all consonants and vowels is in fact equivalent to the representations in (1) and (2) and to the FST in Figure 2, which all use standard natural class variables. It would be equivalent because it would have the same extension.

[5]The technical motivation for explicitly encoding the beginnings and ends of strings is explained in Jardine et al. (2014).

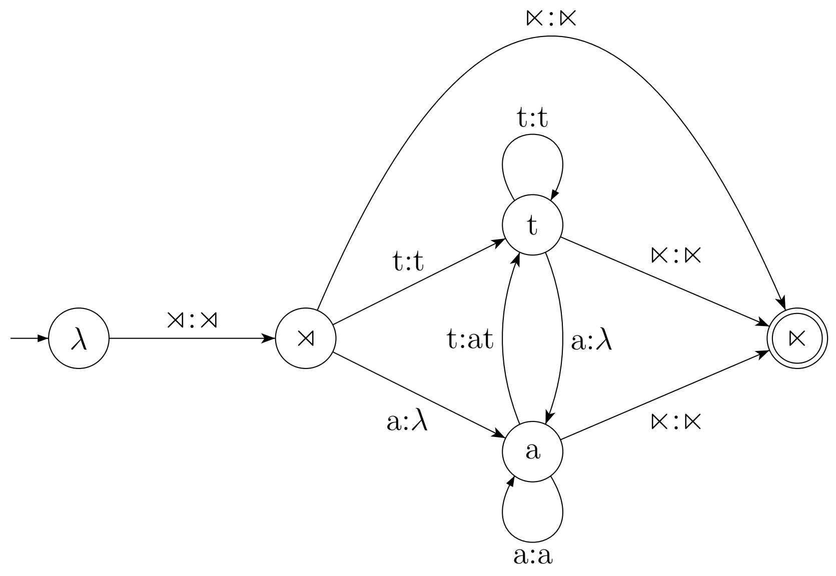


Figure 3: The ISL FST for final vowel deletion

These examples illustrate how the states and the ability to write the empty string λ allow the ISL FST to deterministically decide whether to write a given vowel or not. When an *a* is read, it is not written out right away. This is because it is unknown at that point, as the string is processed left to right, whether that *a* is word-final or not. If it is word-final, nothing further is written and the FST moves to the final state upon reaching the end marker. If it is not, the first symbol of the next part of the output is always *a*.

2.4 Properties of ISL functions

ISL functions have several important properties that are important to the study of phonology. First, Chandlee (2014) and Chandlee and Heinz (to appear) discuss how all phonological processes which can be described with a rule A$\rightarrow$ B / C __ D applying simultaneously are ISL functions provided only finitely many strings match the structural description (or equivalently, provided there is a longest string in CAD). This excludes long-distance phenomena (such as consonant harmony) that involve segments interacting over an arbitrary number of intervening segments.

Second, Chandlee and Heinz (to appear) also explain that ISL functions do not necessarily correspond to individual processes. Maps that include multiple phenomena (such as multiple ordered rules in rule-base grammars or many interacting constraints in OT grammars) can still be ISL functions. A precise accounting of which such maps are ISL and which are not is an outstanding goal. This paper makes a contribution to this goal by showing that a variety of opaque maps are ISL.

Third, ISL functions are a proper subclass not only of the regular relations, but also of the class of *subsequential* functions. Regular relations are those definable with FSTs of the sort illustrated in Figure 2. Subsequential functions are definable by *deterministic* FSTs (see Chandlee and Heinz (2016) for further discussion). Thus, the ISL functions are an extremely restrictive class in terms of the kind of computations they can represent. The importance of this restriction to phonological theory will be discussed further in section 2.5.

Fourth, ISL functions have a clear cognitive interpretation. Rogers et al. (2013) explain how different subregular classes of formal languages can be understood in terms of the different kinds of memory they require (and hence distinguish). The kind of memory needed to compute a ISL function is fleeting: it is only necessary to scan the input word w and remember the most recent substring of length k. Everything else can be thrown out. It is not necessary to compare a substring in one part of w with a substring in another part of w. Nor is it necessary to remember the number of times particular substrings occurred in w. This is how ISL functions are like the Strictly Local (SL) formal languages discussed by Rogers and Pullum (2011); Rogers et al. (2013). Cognitively, this is one of the simplest forms of memory.

Fifth, k-ISL functions are provably learnable by algorithms that are very efficient in both time and data (Chandlee, 2014; Chandlee et al., 2015b; Jardine et al., 2014). This means that given any k-ISL function f and a sufficient finite set of points exemplifying f these algorithms invariably return a FST whose extension is f. Furthermore, the time the algorithm takes is reasonable with respect to the size of its input data. Additionally, the amount of data needed is also reasonable with respect to the size of the ISL FST corresponding to f. Here, 'reasonable' has a clear meaning from the theory of computational complexity; the aforementioned papers provide the details. These results are significant because comparable results do not exist for the subsequential class nor for the class of regular relations. While a learning algorithm exists for the subsequential class (Oncina et al., 1993) it is less efficient than the algorithms for k-ISL. And provably no algorithm exists which identifies the class of regular relations exactly from positive data.

2.5 Relevance to phonological theory

Above it was stated that the ISL functions are a restrictive class of maps—they are subclasses of both the subsequential and regular classes of maps. To better situate the results we present in the next section, we first comment here on why this is important for phonological theory and where other phonological theories lie in terms of the maps that they can describe.

Those unfamiliar with formal language theory might wonder why restrictiveness matters, but an analogy with OT will hopefully help make the relevance clear. Suppose there were three competing sets of constraints under typological investigation (call them CON_1, CON_2, and CON_3). If it became known that the typology of CON_3 was properly included in CON_2, which was in turn properly included in CON_1, this would be a fact worth knowing. It would tell us that any map describable with CON_3 is describable with CON_2 and CON_1, but not vice versa.

Likewise with these classes of maps. Every ISL function is subsequential and regular but not vice versa. This means ISL functions are restrictive in a computationally important way, and so an ISL theory of phonology draws a small circle around a large class of phonological maps. Such maps are indeed also regular (as shown by Johnson (1972) and Kaplan and Kay (1994)), but the 'ISL circle' is a tighter and therefore stronger characterization. Importantly, this is a better characterization—in computational terms—than characterizations in OT and SPE. SPE rules describe the entirety of the regular maps (Johnson, 1972; Kaplan and Kay, 1994), and OT can describe maps outside of the regular class (Frank and Satta, 1998; Riggle, 2004; Gerdemann and Hulden, 2012).

It is often pointed out that SPE rules or OT grammars more easily capture the distinction between 'natural' and 'unnatural' processes and therefore provide better typological predictions than computational restrictions. However, 'naturalness', in common practice, is implemented as a meta-theoretic restriction based on phonetic principles on possible rules or constraints, and their interactions or rankings (as in, e.g., Hayes et al., 2004). We argue that these same types of restrictions could also be used to further restrict the class of ISL functions to a class that is both computationally appealing and phonetically natural. However, since the focus of the current paper is to characterize the *computational* nature of opaque maps, we do not attempt to do so here.

To summarize, subregular function classes like ISL present a learnable, restrictive, computational theory of phonological maps. The next section explores the kinds of opaque maps ISL functions are capable of describing using Baković (2007) and Kavitskaya and Staroverov (2010) as a catalog of the known typology of opaque maps. These works in particular were chosen because they provide a more comprehensive collection of opaque phenomena that go beyond the classic examples of counter-bleeding and counter-feeding (Kiparsky, 1971, 1973). Despite the nuanced differences among the types of opaque maps we will analyze, the fact is that all have the ISL property.

3 Opaque ISL maps

In this section we demonstrate that four categories of opaque maps can be modeled with ISL functions. These cases are listed below. Three are the ones discussed by Baković (2007); the case of fed counterfeeding in Tundra Nenets comes from Kavitskaya and Staroverov (2010).

- Cross-derivational feeding in Lithuanian
- Counterbleeding in Yowlumne
- Non-gratuitous feeding in Classical Arabic
- Fed Counterfeeding in Tundra Nenets

The following three additional categories of opaque maps are detailed in Appendix II.

- Counterfeeding on Environment in Bedouin Arabic
- Counterfeeding on Focus in Bedouin Arabic
- Self-destructive feeding in Turkish

We will present each example using the analysis of the original author, though this does not amount to a claim that their analysis is correct. Our aim is simply to demonstrate that interactions that have been called opaque are ISL maps, so toward that end we maintain the original analysis. It is also important to note that while the examples are presented in a rule-based framework (as is common for opaque generalizations) and therefore include two distinct generalizations that interact opaquely, our analyses study the opaque generalization *directly* as an input-output map. That is, we did not proceed by first modeling each generalization as an ISL function and then taking the composition of the two.[6] This is important because it means our results do not depend on any particular decomposition of a given opaque generalization.

To clarify the exposition, we shall discuss the examples in order of their k-value. Recall that k is the size of the input substring that the function is mindful of at any given time. Cross-derivational feeding in Lithuanian is 2-ISL (i.e., $k = 2$), the next three cases are 3-ISL. (In Appendix II, two cases are 3-ISL and self-destructive feeding in Turkish is 5-ISL.)

All it takes to accomplish our goal is to present an ISL FST for each opaque map (i.e., an FST whose extension is equivalent to the reported generalization). However, the size of these FSTs can make it inconvenient to study them in the graphical form of Figure 3. Therefore, in this section we introduce an alternative, tabular notation for ISL FSTs, which makes them much easier to examine. In particular, we will use what we call an *abbreviated input/output table* which makes it easier to verify that a map is ISL and to determine the output string for a given input. Essentially, these tables provide all the information one needs to construct the ISL FST. We have also provided, in an online supplement, the complete FSTs for all of the example maps in §3 and Appendix II in both table and graphical format.[7]

3.1 Input/output tables

The example ISL FST in section 2 (Figure 3) has a simple alphabet of two letters (t and a). To describe the phonology of a language at the level of its phonemic and phonetic inventory, it will be necessary to have symbols for each element of these inventories. Consequently the number of states and transitions will increase dramatically, making the FST harder to read in graphical form. Generally, if there are n letters in the alphabet the number of states in a k-ISL FST is $n^k + 1$ because there are $n^k - 1$ states corresponding to the sequences up to length $k - 1$ plus the $\rtimes$ and $\ltimes$ states.

While the size of ISL FSTs may seem daunting, keep in mind that these are not the smallest FSTs with these extensions. There are efficient procedures which can make them much more compact (Mohri, 1997). The size of ISL FSTs follows from their particular structure, which is designed to make clear that the functions they compute are ISL.

What we are calling an input/output table is basically a list of the transitions of the ISL FST. A full input/output table for the ISL FST in Figure 3 which models word-final vowel deletion is shown in Table 1. The rows in an input/output table serve the same purpose as the transitions of an FST: the current output is determined by the current $(k - 1)$-suffix and the current input symbol. Recall that for the function depicted in

[6]Formally, if f and g are functions then their composition $f \circ g$ is equivalent to using the output of g as the input to f. E.g., if $g(x) = y$ and $f(y) = z$ then $f \circ g(x) = f(g(x)) = f(y) = z$.

[7]A Python script is also provided that readers can use to generate the output string for any input using the FSTs for all of the languages analyzed in §3.

Figure 3, $k = 2$, so Table 1 lists 1-suffixes. For example, Row (g) indicates that an a following a t should be output as λ. Because of the particular structure of ISL FSTs, the destination state of this transition can be inferred: it is the last $k - 1$ symbols of the 1-suffix concatenated to the new input. In this example, that would be the last $k - 1$ or 1 symbols of ta, so the FST would proceed to state a.

	1-suffix	Input	Output		1-suffix	Input	Output		1-suffix	Input	Output
a.	λ	$\rtimes$	$\rtimes$	e.	t	$\ltimes$	$\ltimes$	i.	a	t	at
b.	$\rtimes$	$\ltimes$	$\ltimes$	f.	t	t	t	j.	a	a	a
c.	$\rtimes$	t	t	g.	t	a	λ				
d.	$\rtimes$	a	λ	h.	a	$\ltimes$	$\ltimes$				

Table 1: Full input/output table for the ISL FST in Figure 3.

The process of mapping an input string to its output string can be thought of as sliding a window of length k over the input from left to right. This window is moved one input symbol at a time, not unlike a FST. This process is demonstrated in Table 2 with the derivation of /tat/→[tat]. The left portion of the window reveals the current $(k - 1)$-suffix (again in this example, it is a 1-suffix since $k = 2$). The right portion of the window includes the current input symbol. For each new input symbol, we look up its corresponding output given the $(k - 1)$-suffix in the input/output table. To clarify the roles of each symbol, the shading in the derivation matches the shading in the table. Additionally, each step is labeled with its corresponding table number and row. The first step, which corresponds to row (1a), has been omitted.

$\rtimes$	t	a	t	$\ltimes$	$\rtimes$	t	a	t	$\ltimes$	$\rtimes$	t	a	t	$\ltimes$	$\rtimes$	t	a	t	$\ltimes$
$\rtimes$	t				$\rtimes$	t	λ			$\rtimes$	t	λ	at		$\rtimes$	t	λ	at	$\ltimes$
	(1c)					(1g)						(1i)						(1e)	

Table 2: Example derivation using an input/output table (the first step is omitted)

We will make these input/output tables even more succinct by not showing those rows where the inputs and outputs are identical. An *abbreviated input/output table* is one which only shows cases where the inputs do not match the outputs. Table 3 shows an abbreviated input/output table for the ISL FST in Figure 3. In contrast to Table 1, Table 3 has only three rows. In addition to improving readability, the advantage of an abbreviated input/output table is that we can isolate the 'interesting' part of the FST, or the part that represents the structural change of the phonological map. For any input symbol and $(k-1)$-suffix not found in the abbreviated input/output table, it is understood that the output written is identical to the input (as in steps (1c) and (1e) in Table 2).

To illustrate the kind of analysis we will conduct in this paper, consider a classic case of counterfeeding opacity: vowel lowering in Danish (Hyman, 1975; Lundskaer-Nielsen and Holmes, 2011). In Danish, front vowels lower in proximity to /r/. This occurs in

	1-suffix	Input	Output
(1d)	⋊	a	λ
(1g)	t	a	λ
(1i)	a	t	at

Table 3: Abbreviated input/output table for the ISL FST in Figure 3

a chain shift, in which each vowel lowers to the vowel one 'step' in height below it. To simplify the discussion, we focus on front vowels following an /r/. A simple rule-based analysis is presented in (7) with the rules applying in the order presented.

(7) Danish lowering: ɛ → æ/ r __ ; e → ɛ / r __ ; i → e / r __

This lowering process is a 2-ISL function: whether or not an input /i/ is output faithfully depends on whether or not the preceding segment in the input is an /r/. We can describe this function exactly with the abbreviated input/output table in Table 4. An example derivation is given in Table 5.

	1-suff	Input	Output
a.	r	i	e
b.	r	e	ɛ
c.	r	ɛ	æ

Table 4: Abbreviated input/output table for Danish lowering

⋊ i r i ⋉	⋊ i r i ⋉	⋊ i r i ⋉	⋊ i r i ⋉
⋊ i	⋊ i r	⋊ i r e	⋊ i r e ⋉
		(4a)	

Table 5: Example derivation using Table 4

Table 4 succinctly represents the map which lowers front vowels after /r/. We shall use such tables throughout this paper to show that opaque interactions of ISL phonological maps are themselves ISL—that is, for these interactions, there exists some k such that the relevant changes from input to output are determined by $(k-1)$-suffixes in the input.[8] To make the tables even more compact and readable, in our analyses we will use symbols like C, N, and V as variables in the conventional manner, though again in the actual FSTs each segment would be represented separately.

[8]It follows that a map that is not ISL could not be represented with such a table.

3.2 Cross-derivational feeding in Lithuanian

We begin with an interaction between epenthesis and voicing assimilation in Lithuanian that is 2-ISL. Baković (2007) identifies this as an example of what he calls *cross-derivational feeding*, a type of opacity in which one process applies in order to avoid a derivation in which another process would create a marked structure.

In Lithuanian, [i]-epenthesis occurs between identical obstruents and voiceless obstruents assimilate to following voiced obstruents. In the following rules, let D represent a voiced obstruent and K represent any obstruent, regardless of voicing.

(8) a. Epenthesis in Lithuanian (Baković, 2007)
$\emptyset \rightarrow \text{i} / \text{K}_1\_\_\text{K}_2$, where $\text{K}_1 = \text{K}_2$
b. /at-taiki:ti/→[atitaiki:ti][9], 'to make fit well'

(9) a. Voicing assimilation in Lithuanian (Baković, 2007; Odden, 2005)
$\text{K} \rightarrow [+\text{voice}] / \_\_\ \text{D}$
b. /ap-gauti/ ↦ [abgauti], 'to deceive'

In Odden (2005)'s formulation of (8) epenthesis occurs only between homorganic obstruents, but Baković (2007) argues against this analysis because it misses the generalization that 'assimilation is bled in precisely those contexts where it would otherwise create pairs of completely adjacent consonants' (pg 235). He instead suggests the version shown in (8), though with either version of the rule the interaction of epenthesis followed by assimilation is not actually opaque, but a standard case of bleeding.

The opacity comes into play when addressing the fact that these two rules cannot account for the example in (10), under either ordering.

(10) Cross-derivational feeding in Lithuanian (Baković, 2005; Odden, 2005)
/ap-berti/ ↦ [apiberti], 'to strew all over'; *[abberti],*[abiberti]

The application of epenthesis and not assimilation to /ap-berti/ is a case of overapplication, which Baković (2007) calls cross-derivational feeding because the epenthesis appears to take place in order to *avoid a derivation* in which voicing assimilation would create the marked structure of identical obstruents. Though this is missed by the rule-based analysis, Baković (2007) shows that it can be handled rather straightforwardly in OT. As mentioned above, our analyses do not depend on the rule-based descriptions or on the ability to decompose the generalization into multiple individual ones. Therefore, while not considered opaque in the rule-based analysis, we include this case in our survey because its *extension* (which is identical in both Odden (2005)'s and Baković (2007)'s analyses) reflects the type of opacity called cross-derivational feeding.

Thus, we now show that this interaction of epenthesis and assimilation can be modeled with a single 2-ISL map. The essence of this map's computation is the following. When a voiceless obstruent appears in the input, its output is temporarily delayed until it can be determined whether or not it will be subject to voicing assimilation (9). This depends on the following segment. If the following segment is a voiced obstruent of a *different* place, the voiceless obstruent should be output as voiced, because epenthesis will not

[9]For simplicity, we focus on voicing and omit palatalization from the transcriptions and discussion.

be triggered. If the following segment is an obstruent of the *same* place, then both the voiceless obstruent and [i] are output.

This interaction is 2-ISL because a scanning window of size 2 is sufficient to distinguish the above conditions. Table 6 illustrates these various conditions. First, voiceless obstruents, which may or may not be subject to assimilation, are initially output as λ (rows 6a, 6b, 6e, and 6f). Voiced obstruents, on the other hand, are always output unchanged (rows 6c, 6d, 6g, and 6h).

	1-suff	Input	Output		1-suff	Input	Output		1-suff	Input	Output
a.	$\rtimes$	t	λ	h.	V	b	b	o.	p	t	p
b.	$\rtimes$	p	λ	i.	t	V	tV	p.	p	d	bd
c.	$\rtimes$	d	d	j.	t	$\ltimes$	t$\ltimes$	q.	t	t	ti
d.	$\rtimes$	b	b	k.	t	p	t	r.	t	d	tid
e.	V	t	λ	l.	t	b	db	s.	p	p	pi
f.	V	p	λ	m.	p	V	pV	t.	p	b	pib
g.	V	d	d	n.	p	$\ltimes$	p$\ltimes$				

Table 6: Abbreviated input/output table for Lithuanian cross-derivational feeding

When a voiceless obstruent is followed by a vowel (rows 6i and 6m), the end of the word (rows 6j and 6n), or a heterorganic voiceless obstruent (rows 6k and 6o), the voiceless obstruent is output faithfully. (In the latter case, the following heterorganic voiceless obstruent is not included in the output, as it may itself be subject to assimilation depending on what comes next, per the discussion above.) If the following symbol is a heterorganic voiced obstruent, the preceding voiceless obstruent is output as voiced (rows 6l and 6p).

The remaining cases of interest are when a voiceless obstruent precedes a homorganic obstruent. These are the cases in which epenthesis applies. When a voiceless obstruent is followed by an identical voiceless obstruent (rows 6q and 6s), the first obstruent is output followed by an [i]; thus, for example, in (6q) a /tt/ sequence is output as [ti]. In this case, the output of the second voiceless obstruent is withheld, as it itself may be followed by a trigger for voicing assimilation. When a voiceless obstruent is followed by a homorganic voiced obstruent, the voiceless obstruent is output followed by an epenthesized [i] and the voiced obstruent; thus in (6r) an input /td/ sequence is output as [tid].

This final case is exactly the case of overapplication of epenthesis due to cross-derivational feeding. The preceding tables have shown this interaction to be definable by the 1-suffix of each input; thus it is 2-ISL. Table 7 illustrates how the correct outputs for the three crucial cases—epenthesis of identical segments, voicing in heterorganic segments, and cross-derivationally fed epenthesis between homorganic segments differing in voicing—are obtained by a scanning window of size 2 implementing the input/output relations listed in the above tables.

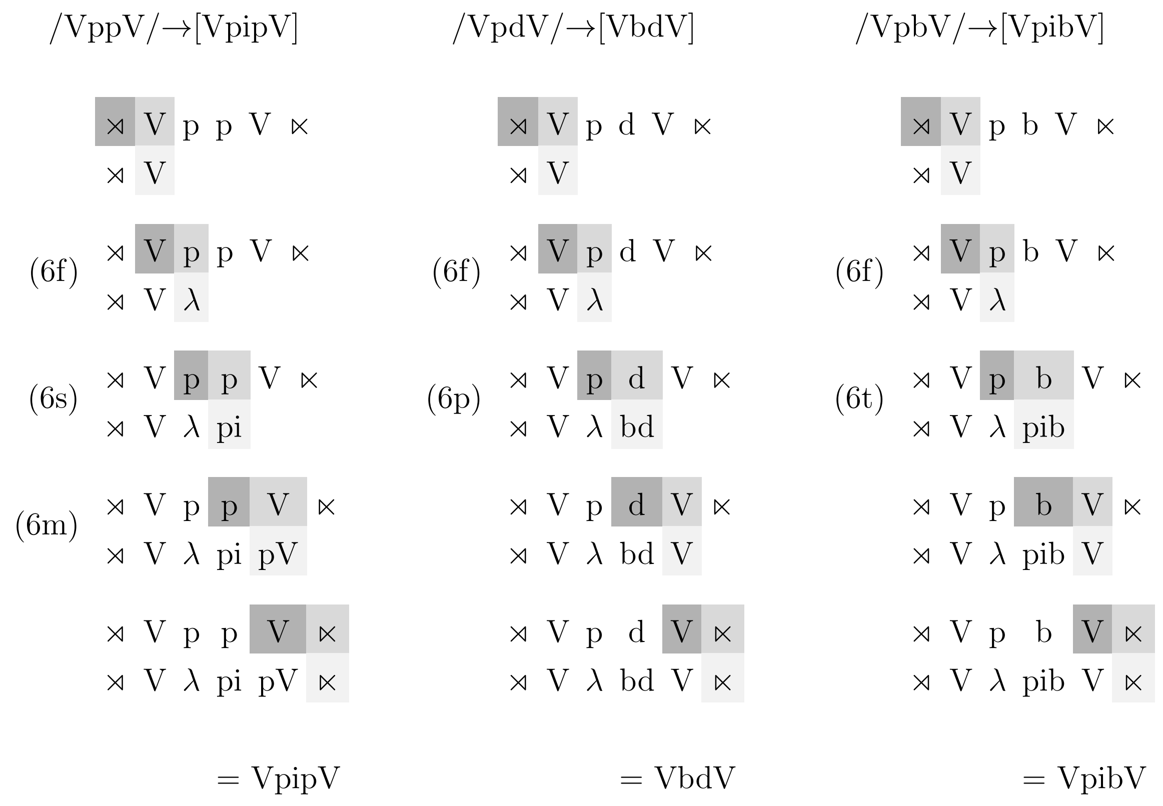


Table 7: Example derivations for cross-derivational feeding in Lithuanian

3.3 Counterbleeding in Yowlumne

We now turn to a case of **counterbleeding**, from Yowlumne[10], and show that it is 3-ISL. In Yowlumne, all long vowels become [−high] (11-a), and vowels in closed syllables shorten (11-b).

(11) a. [+long] → [−high]
b. V → [−long] / __C {C,⋉}

With long vowels in closed syllables, the application of shortening obscures the environment for lowering. The example in (12) shows apparent overapplication of (11-a), because the subsequent application of (11-b) removes the triggering long vowel.

(12) Yowlumne (McCarthy, 1999)
/mi:k-hin/ ↦ [mekhin], 'swallowed'

These processes can be viewed as a single map whose computation can be summarized as follows: when a high long vowel is read, the output is temporarily delayed until it can be determined whether 1) to both lower and shorten the vowel or 2) just lower it. Condition 1 holds if the two segments after the vowel are C⋉or CC, where C ranges over

[10]This language is referred to as Yawelmani Yokuts in McCarthy (1999), but the currently accepted name is Yowlumne (see Weigel, 2005). We thank an anonymous reviewer for pointing this out in addition to identifying an error in our original description and analysis of the shortening process.

all consonants. Condition 2 holds if the next two segments are anything except C⋉or CC.

This map is 3-ISL because these conditions can be checked by scanning a 3-segment window in the input.[11] (Note that /i:/ counts as one symbol.) The following abbreviated input/output tables, focusing on the lowering and/or shortening of an input /i:/, illustrate this. We assume the alphabet {i:, V, C}, where i: represents long high vowels and C and V are variables ranging over the set of consonants and the set of short vowels plus long non-high vowels, respectively.

First, any input /i:/ will be initially output as λ, temporarily delaying its corresponding output—whether it is eventually output as [e:] or [e] depends on whether the following input is C⋉, CC, or something else. Thus its corresponding output is λ regardless of the 2-suffix, as Table 8 illustrates with several example rows for the input /i:/.

	2-suff	Input	Output		2-suff	Input	Output		2-suff	Input	Output
a.	⋊C	i:	λ	j.	Ci:	V	e:V	s.	CV	V:	λ
b.	CC	i:	λ	k.	Vi:	V	e:V	t.	VC	V:	λ
c.	VC	i:	λ	l.	⋊i:	C	λ	u.	VC	V:	λ
d.	i:C	i:	λ	m.	Ci:	C	λ	v.	CV:	C	λ
e.	⋊V	i:	λ	n.	Vi:	C	λ	w.	VV:	C	λ
f.	CV	i:	λ	o.	i:C	V	e:CV	x.	V:C	C	VCC
g.	VV	i:	λ	p.	i:C	C	eCC	y.	V:C	⋉	VC
h.	i:V	i:	λ	q.	i:C	⋉	eC⋉				
i.	⋊i:	V	e:V	r.	CC	V:	λ				

Table 8: Abbreviated input/output table for Yowlumne counterbleeding

The change, then, hinges on inputs following an /i:/. If an /i:/ is followed by anything other than a /C/, it is not in the shortening environment. Thus, it can be output as [e:], with only lowering applying. Rows (i-k) in Table 8 illustrate: a /V/ immediately following a /i:/ will be output as [e:V]. However, if /i:/ is followed by a /C/ (rows 8l-8n), the output still cannot be determined: if it is a word-final /i:C/ sequence or if another C follows, then it should be output as [eC], with both lowering and shortening; otherwise, it should be output as [e:C], with only lowering. Thus, the output must be again delayed, and so a /C/ following /i:/ is always output as λ.

Thus an input /i:C/ sequence is output as λ, as whether or not it is realized as [e:C] or [eC] is contingent on what comes next. An input /V/ following an /i:C/ sequence is output as [e:CV] (row 8o). This output comprises the output [e:C] of the /i:C/ 2-suffix (which was previously held) concatenated to the output [V] for the new input /V/. This

[11]An anonymous reviewer added that Yowlumne also has a rounding harmony process that both counterfeeds and counterbleeds lowering. The interaction of all three of these processes is still ISL, but for $k = 6$. Under the assumption that the harmony takes place over at most two intervening consonants (e.g., uC(C)i:), a window of 6 would be required to check whether an input long high vowel is in both the contexts for rounding and shortening: uC(C)i:C#. We thank the reviewer for bringing out this interesting extension of the Yowlumne example.

can be interpreted as follows: now that the /i:C/ has been seen to precede a /V/, it is clear that the /i:/ is not in the shortening environment, and so the output reflects only lowering. In contrast, as shown in rows (8p) and (8q), an input /C/ or /⋉/ following an input /i:C/ is output as [eCV] and [eC⋉], respectively, reflecting both lowering and shortening.

Lastly, rows (8r) - (8y) demonstrate the application of shortening without lowering. The output for V: (which abbreviates all non-high long vowels) is V in the same contexts that i: is shortened (i.e., closed syllables).

Table 9 presents derivations of /Ci:CV/→[Ce:CV] and /Ci:C/→[CeC] to show how the input-output correspondences in these tables correctly capture the opaque interaction.

/Ci:CV/→[Ce:CV]						
	⋊	C	i:	C	V	⋉
	⋊	C				
(8a)	⋊	C	i:	C	V	⋉
	⋊	C	λ			
(8m)	⋊	C	i:	C	V	⋉
	⋊	C	λ	λ		
(8p)	⋊	C	i:	C	V	⋉
	⋊	C	λ	λ	e:CV	
	⋊	C	i:	C	V	⋉
	⋊	C	λ	λ	e:CV	⋉
	= Ce:CV					

/Ci:C/→[CeC]					
	⋊	C	i:	C	⋉
	⋊	C			
(8a)	⋊	C	i:	C	⋉
	⋊	C	λ		
(8m)	⋊	C	i:	C	⋉
	⋊	C	λ	λ	
(8q)	⋊	C	i:	C	⋉
	⋊	C	λ	λ	eC⋉
	= CeC				

Table 9: Example derivations for Yowlumne

These derivations highlight the fact that a scanning window of size 3 is sufficient to model the combined shortening and lowering map—the longest $(k-1)$-suffix needed to produce the correct output for any input symbol is /i:C/, which is of length 2. Thus, this opaque interaction is 3-ISL.

3.4 Non-gratuitous feeding in Classical Arabic

Another type of opacity discussed by Baković (2007) is **non-gratuitous feeding**, in which the structural description of one process is obscured by a second process that is fed by the first. For example, in Classical Arabic a vowel is epenthesized before an initial consonant cluster (rule (14-a)).[12] This then feeds glottal stop epenthesis (14-b), as initial vowels must be preceded by a glottal stop in Classical Arabic.

(13) Classical Arabic (McCarthy, 2007)
/ktub/ ↦ [ʔuktub], 'write.MascSg!'

(14) a. $\emptyset \rightarrow$ V / # __ CC
b. $\emptyset \rightarrow$ ʔ / #__V

This interaction can be considered opaque because both the triggering environment for vowel epenthesis and its output are altered by the application of glottal stop epenthesis—that is, the resulting word-initial #VCC sequence is not word-initial in the output.

The combined map is 3-ISL. This can be shown fairly easily, as the crucial input-output correspondences are defined in the same word-initial environment. Table 10 summarizes.

	2-suff	Input	Output		2-suff	Input	Output
a.	⋊	V	ʔV	c.	⋊C	V	CV
b.	⋊	C	λ	d.	⋊C	C	ʔVCC

Table 10: Abbreviated input/output table for Classical Arabic epenthesis

When the suffix is ⋊, an input /V/ is output as [ʔV]. For the same suffix ⋊, an input /C/ is instead output as λ. This is to 'hold' a word-initial /C/, as it may be the first member of a word initial CC sequence, and thus subject to vowel epenthesis. The application of vowel epenthesis is then defined for inputs when the 2-suffix is /⋊C/. If a /V/ follows (row 10c), then vowel epenthesis does not apply, and [CV] is output. If a /C/ follows (row 10d), then both vowel epenthesis and glottal epenthesis apply, and so [ʔVCC] is output. Contrasting derivations for /CVC/→[CVC] (when neither process is triggered) and /CCVC/→[ʔVCCVC] (when both processes are triggered, corresponding to (13)) are shown in Table 11.

As illustrated in Table 11, the interaction between these two processes is 3-ISL, as it can be captured with a scanning window of size 3.

3.5 Fed Counterfeeding in Tundra Nenets

Another type of opacity, described by Kavitskaya and Staroverov (2010), is called **fed counterfeeding**. An example comes from Tundra Nenets (shown in (15)), with the debuccalization and vowel deletion rules shown in (16).

(15) Tundra Nenets (Kavitskaya and Staroverov, 2010)
a. /tasʌ/ ↦ [tas], 'whole'

[12]The epenthesized V is in fact identical to the V following the consonant cluster (see McCarthy, 2007), but for notational simplicity we ignore the featural makeup of V. Ensuring that the vowels are identical would add 1 to the *k*-value for this process, in order to look ahead to see what vowel follows the consonant cluster. We also follow Baković (2007) in considering the linear version (that is, with no reference to syllable structure) of these two processes. See that work for discussion.

/CCVC/→[ʔVCCV]

(10b)

⋊	C	C	V	C	⋉
⋊	λ				

(10d)

⋊	C	C	V	C	⋉
⋊	λ	ʔVCC			

⋊	C	C	V	C	⋉
⋊	λ	ʔVCC	V		

⋊	C	C	V	C	⋉
⋊	λ	ʔVCC	V	C	

⋊	C	C	V	C	⋉
⋊	λ	ʔVCC	V	C	⋉

= ʔVCCVC

/CVC/→[CVC]

(10b)

⋊	C	V	C	⋉
⋊	λ			

(10c)

⋊	C	V	C	⋉
⋊	λ	CV		

⋊	C	V	C	⋉
⋊	λ	CV	C	

⋊	C	V	C	⋉
⋊	λ	CV	C	⋉

= CVC

Table 11: Example derivations for Classical Arabic

b. /t^{j}imjʌs/ ↦ [t^{j}imjʔ], 'it rotted'

(16) a. {t, d, s, n, ŋ} → ʔ / ___ #
b. ʌ → ∅ / ___ (ʔ) #

In a map like (15-a), we observe counterfeeding: if vowel deletion applied first then debuccalization would also apply to derive *taʔ. But (15-b), which in a rule-based analysis proceeds as /t^{j}imjʌs/ ↦ t^{j}imjʌʔ ↦ [t^{j}imjʔ], instead demonstrates feeding. Thus the same two rules can exhibit both feeding and counterfeeding, depending on the input form.

Using the alphabet {ʌ, ʔ, T, C, V} (where T = {t, d, s, n, ŋ}, C is any consonant aside from ʔ and those in T, and V is any vowel except ʌ), we can show that this combined fed counterfeeding map is 3-ISL. Since ʌ-deletion doesn't have a left context, the output of ʌ is λ no matter what 2-suffix precedes it. This is indicated in rows (a-e) of Table 12 with X standing in for any segment.[13]

The situation is similar when the input is T. As the debuccalization rule has no left context, whenever a T is input it must be held until it can be determined whether it is word-final (rows 12f-12j). Again, the output of T when the 2-suffix ends in T (row 12j) is T, representing the 'return' of the T in the suffix, which is now known to not be word-final. This is also why the output is T in row (12e): the additional input of ʌ verifies that the suffix T is not word-final and therefore should be output unchanged. Note this

[13]The output in row d is ʌ because the ʌ of the suffix was already output as λ. Since this suffix ʌ is now known to not be word-final (because another ʌ follows it), it can now be safely output.

	2-suff	Input	Output
a.	XC	ʌ	λ
b.	XV	ʌ	λ
c.	Xʔ	ʌ	λ
d.	Xʌ	ʌ	ʌ
e.	XT	ʌ	T
f.	XC	T	λ
g.	XV	T	λ
h.	Xʔ	T	λ
i.	Xʌ	T	λ
j.	XT	T	T
k.	Xʌ	ʔ	λ
l.	Xʌ	C	ʌC
m.	Xʌ	V	ʌV
n.	ʌT	⋉	ʔ⋉
o.	CT	⋉	ʔ⋉
p.	VT	⋉	ʔ⋉
q.	TT	⋉	ʔ⋉
r.	ʔT	⋉	ʔ⋉
s.	ʌʔ	⋉	ʔ⋉
t.	Cʌ	⋉	⋉
u	Vʌ	⋉	⋉
v.	Tʌ	⋉	⋉
w.	ʔʌ	⋉	⋉

Table 12: Abbreviated input/output table for Tundra Nenets fed counterfeeding

is the key to the fed counterfeeding interaction originally shown in (15-a), where an input /s/ preceding a deleted ʌ escapes debuccalization.

When an input ʔ follows a 2-suffix that ends in ʌ (row 12k), the output is also λ, because if (and only if) word-final this ʔ will also trigger deletion of ʌ. In contrast, if the input following a 2-suffix that ends in ʌ is C or V, then the output 'returns' the held ʌ, which we now know will not be deleted (rows 12l and 12m).

Since these are both word-final processes, the crucial rows of the input/output table (12n)-(12w) are those for which the input is ⋉. When ⋉ follows the 2-suffix ʌT (row 12n), the context for both deletion and debuccalization has been found, so the output is ʔ⋉. Debuccalization also applies when any of the 2-suffixes CT, VT, TT, and ʔT (rows 12o-12r) are followed by ⋉. Deletion alone applies when ⋉ follows the 2-suffix ʌʔ (row 12s). When any 2-suffix that ends in ʌ is followed by ⋉, the ʌ is deleted, as shown in rows (12t)-(12w). Table 13 illustrates this interaction with some example derivations.

3.6 Discussion

This concludes our analysis of four distinct types of opaque phonological maps, all of which have been shown to be ISL functions for some value of k. Similar analyses of the other three cases in this typology are presented in Appendix II. Altogether these analyses show that a seemingly diverse set of interactions can be given a unified computational analysis based on the common property of ISL. This does not, however, amount to a claim that opacity *itself* is ISL. Long-distance processes, which are not ISL, can also be involved in opaque interactions. Nonetheless, it is still a non-trivial result that every example in this typology of opaque maps is ISL.

While the object of each analysis presented above was the extension of the opaque interaction as a whole, each case can be (and has been) thought of as an interaction between two individual processes. This raises two important questions. First, given two ISL maps with possibly different k-values, what will the k-value of the map that describes their interaction be? The second question is closely related: is the class of ISL functions closed under composition—that is, is the composition (see Fn. 6) of any two ISL functions

	/CVTᴧ/→[CVT]		/CVCᴧT/→[CVCʔ]
	⋊ C V T ᴧ ⋉		⋊ C V C ᴧ T ⋉
	⋊ C		⋊ C
	⋊ C V T ᴧ ⋉		⋊ C V C ᴧ T ⋉
	⋊ C V		⋊ C V
(12g)	⋊ C V T ᴧ ⋉		⋊ C V C ᴧ T ⋉
	⋊ C V λ		⋊ C V C
(12e)	⋊ C V T ᴧ ⋉	(12a)	⋊ C V C ᴧ T ⋉
	⋊ C V λ T		⋊ C V C λ
(12v)	⋊ C V T ᴧ ⋉	(12i)	⋊ C V C ᴧ T ⋉
	⋊ C V λ T ⋉		⋊ C V C λ λ
	= CVT	(12n)	⋊ C V C ᴧ T ⋉
			⋊ C V C λ λ ʔ⋉
			= CVCʔ

Table 13: Example derivations for Tundra Nenets debuccalization and deletion

guaranteed to also be ISL? We address these two questions in turn.

Table 3.6 reviews the results of this section and Appendix II, presenting the k-values for each opaque map discussed in this section and the appendix along with the k-values for the individual processes of each interaction.[14] For ISL maps corresponding to single phonological rules, determining the k-value is straightforward: k is the length of the longest string in the structural description of the process. The longest k-value for one of the interactions, listed in the rightmost column, is 5, for Turkish, whereas the others are either 2 or 3. Note that there is no clear correlation between the size of the k-values for the basic processes and the k-value of the resulting interaction. For example, both epenthesis and assimilation in Lithuanian (§3.2) are 2-ISL, as is their interaction. In contrast, epenthesis in Turkish is 3-ISL and deletion is 4-ISL, whereas their interaction is 5-ISL. Thus, the composition of two ISL processes is not simply the largest of their respective k-values, nor their sum. Predicting the k-value of the composition of two ISL processes appears to be a more subtle matter of how the output of the second process depends on the output of the first.

[14]As stated in Fn. 12, the k-value for process A in Classical Arabic would increase by one to reflect the fact that the inserted vowel is actually a copy of the first vowel in the word. The interaction k-value would then likewise increase to 4.

§	**Opacity Type**	**Language**	**Process**		**Interaction**
			A	**B**	
3.2	cross-derivational feeding	Lithuanian	$k = 2$	$k = 2$	$k = 2$
3.3	counterbleeding	Yowlumne	$k = 1$	$k = 3$	$k = 3$
3.4	non-gratuitous feeding	Classical Arabic	$k = 3$	$k = 2$	$k = 3$
3.5	fed counterfeeding	Tundra Nenets	$k = 2$	$k = 3$	$k = 3$
5.1	counterfeeding on environment	Bedouin Arabic	$k = 3$	$k = 3$	$k = 3$
5.2	counterfeeding on focus	Bedouin Arabic	$k = 3$	$k = 3$	$k = 3$
5.3	self-destructive feeding	Turkish	$k = 3$	$k = 4$	$k = 5$

Table 14: The k-values for the ISL maps analyzed in §3

Nonetheless, in all of the cases analyzed here, there exists some k for which the map that combines the two ISL processes is k-ISL. This is not, however, true generally, as the ISL functions as defined in Chandlee (2014) are not closed under composition. A proof of this fact can be found in the appendix. Thus, the interaction of two ISL processes is not itself *guaranteed* to be ISL. However, subsets of the ISL class may be closed under composition. Chandlee and Lindell (to appear) study ISL maps from a logical perspective and identify a subclass which excludes the counterexample given in the appendix but includes all of the opaque interactions studied in this paper.

In sum, the preceding sections demonstrated that many cases of opacity are ISL for some k, as summarized in Table 3.6. In this subsection, we addressed two important theoretical questions about ISL and map interaction: 1) can we predict the k-value of the interaction of two ISL processes? and 2) is the class of ISL functions closed under composition? The answer to both of these questions is currently “no”, but ongoing research aims to identify the conditions under which interactions preserve the ISL property.

4 Comparison to other theories of phonology

It is natural to wonder how a theory of phonology which asserts that phonological transformations from underlying to surface forms are characterized by ISL functions compares to other theories of phonology. In this section we compare this theory to (1) a theory that asserts that phonological maps are generated by a list of rewrite-rules in the style of Chomsky and Halle (1968) and (2) classic Optimality Theory (OT) (Kager, 1999; Prince and Smolensky, 2004), which asserts that phonological maps are generated by optimizing over a set of ranked, violable constraints.

We acknowledge that theories are not monolothic entities but contain many different variants and parts. Optimality Theory is a good example. It is not possible to address every variant and set of constraints proposed in the OT framework, as well as its variants like Harmonic Grammar (HG) (Legendre et al., 1990) and Harmonic Serialism (HS)

(McCarthy, 2000; Pater, 2012). However, those theories each understand maps as optimizations of violable constraints, and there are consequences that necessarily follow from this core assumption. For this reason, we focus on the computational nature of these theories in the following discussion and put aside other aspects.

The most important aspect we leave aside here is how phonetically grounded the theories are (Hayes et al., 2004). This is not a commitment to the belief that phonetic factors play no role in phonological theory—indeed we believe the most explanatory theory will identify how computational *and* phonetic factors interact to shape phonological typology. The focus of this paper is the contribution of the *computational* factors.

4.1 Points of Comparison

We consider the following points of comparison: typological predictions, learnability, generation, and recognition. Typological predictions refer to a theory's predictions for the kinds of maps that should and should not occur. Learnability refers to the results that have been obtained regarding how the kinds of grammars the theories posit can be learned. Generation refers to whether algorithms exist which reliably compute a surface form from a given underlying form using the grammar. Recognition refers to the converse of the generation process: given a surface form and grammar, is there a procedure that can recover the possible underlying forms?

All four points of comparison are important. Typologically, it is preferable for theories to neither overgenerate nor undergenerate, though we acknowledge there is room for debate on the nature of this over/undergeneration. For instance, it can be difficult to decide between two theories that overgenerate in different ways. However, computational complexity is a clear measure for overgeneration. We thus adopt the principle that theories which overgenerate simple patterns (computationally speaking) are to be preferred over theories which overgenerate complex patterns. This gives us a strong starting point from which to further restrict the theory based on substantive constraints such as phonetic naturalness.

In addition, as human children somehow acquire their native language's phonology, a theory of phonology should provide some guarantee that the grammars it predicts to exist can be learned by provably correct learning algorithms. There are different ways to characterize what it means "to learn" (Kearns and Vazirani, 1994; Jain et al., 1999; de la Higuera, 2010) and different types of learning problems in phonology can be identified. None of the results mentioned in this paper are complete solutions to the problem of phonological acquisition. Nonetheless, it ought not be controversial that algorithms which are well-understood analytically and which are computationally efficient provide important landmarks in understanding how a child could generalize from linguistic stimuli. Indeed, theoretical learnability results have played an important role in the literature (Tesar and Smolensky 1993, 1996, 1998; see also Heinz and Riggle 2011 for a review of the relationship between learnability and phonology).

Finally, it is desirable for there to exist specific algorithms that solve the generation and recognition problems. Generation allows one to reliably compute predictions within the language. Without a generation algorithm, we cannot make reliable predictions about the output form of a given input form. And recognition is a form of parsing, allowing one to reliably interpret surface forms in terms of their potential underlying structure. While recognition may not receive as much attention as generation, recognition algorithms may play a role in some learning problems.

In the remainder of this section, we compare the ISL function theory with rule-based theories like SPE and with Optimality Theory. We first compare the theories with respect to the generation and recognition problems (§4.2). Then we consider the learnability criterion (§4.3). Finally, we address typology (§4.4). Our conclusion is that despite some known deficiencies, the ISL theory of phonology compares well and has sufficient merit to be worthy of further study.

4.2 Generation and Recognition

The fact that ISL functions can be characterized by FSTs is significant. The generation and recognition problems are solved for FSTs. Algorithms like the ones mentioned in §2.4 guarantee that the correct output is generated from a given input. Likewise, it is well known that FSTs can be inverted to do recognition (Beesley and Karttunen, 2003). In other words, if the solution to the generation problem is in terms of FSTs, then one automatically obtains a solution to the recognition problem.

Rule-based grammars also have solutions to the recognition and generation problems. This is because they can be faithfully compiled into FSTs provided it is clear whether they apply simultaneously, left-to-right, right-to-left, and/or optionally (Johnson, 1972; Kaplan and Kay, 1994). The fact that all maps for phonology can be expressed as a list of ordered SPE-style rewrite rules is strong evidence for the claim that phonological maps are regular (i.e., describable with a FST). See Karttunen (1993) and Heinz (2011a,b) for additional discussion on this point.

In Optimality Theory, there are solutions to the generation and recognitions problems under certain assumptions. Frank and Satta (1998) show that OT grammars can be faithfully compiled into FSTs provided the following three conditions hold.

(A1) GEN can be represented with a FST (and so is a regular relation).

(A2) Each constraint must also be describable with a FST.

(A3) There must be a maximum number of violations that each constraint can assign.[15]

Readers are encouraged to consult Frank and Satta (1998), Karttunen (1998), Gerdemann and van Noord (2000), Jäger (2002), Riggle (2004) and Gerdemann and Hulden (2012) for more details.

However, many theoretical phonologists do not accept assumptions A1-A3. Correspondence theory is the most widely adopted theory of GEN, but it is not known whether such a GEN defines a regular relation. Also, not all constraints widely used in the literature can be described with FSTs, such as the constraints in the ALIGN family (Eisner, 1997). Finally, while some theoretical phonologists have argued for assumption A3 (McCarthy,

[15]Jäger (2002) shows that this assumption can be relaxed provided the grammar has 'global optimality', meaning an output that is optimal for some input must be optimal for all inputs for which it is a candidate.

2003), most do not adopt it. Consequently, OT theories of phonology which fail to adopt these three assumptions lack a comprehensive solution to the generation problem.

There are some other approaches to generation and recognition in OT. For instance, Riggle's (2004) solution does not require bounding the number of violations constraints assign (A3), but it does depend on A1 and A2. His solution is guaranteed to be correct provided the map the OT grammar describes is in fact a regular relation (but not all of them are; one example is discussed below). Another solution is present in Albro's (2005) dissertation, which provides a comprehensive OT analysis of the phonology of Malagasy; it assumes GEN is represented by a mildly context-sensitive grammar.

The above discussion indicates that solutions to the generation and recognition problem under typical assumptions in OT, even if currently absent, may be forthcoming. Nonetheless, their present absence contrasts with existing algorithms that reliably solve the generation and recognition problems for SPE and the ISL function theories.

4.3 Learnability

With respect to learnability, there are algorithms which provably learn any ISL function (Chandlee et al., 2014; Jardine et al., 2014) from finitely many examples of (input, output) pairs. These algorithms are provably efficient in the size of the input to the learner. Additionally, it is the case that the amount of data required to learn the pattern exactly is polynomial in the size of the target grammar. In fact, Chandlee et al. (2014)'s results are quadratic, and Jardine et al. (2014)'s results are linear.

With respect to rule-based grammars, there are some results, but they are not as strong. In each case, the input data is assumed to be (input, output) pairs. Johnson (1984) showed how to generate all rules consistent with the input data. And Oncina et al. (1993) proved that the algorithm OSTIA (Onward Subsequential Transducer Inference Algorithm) can learn total subsequential functions in time cubic in the size of the input data. Since many segmental phonological maps appear to be subsequential (Chandlee et al., 2012; Gainor et al., 2012; Chandlee and Heinz, 2012; Heinz and Lai, 2013; Jardine, 2016), and since Kaplan and Kay (1994) show how to convert a FST into a rule, this suggests a method for learning rules. However, Gildea and Jurafsky (1996) have argued that OSTIA, while well-understood analytically, needs more phonology-specific information to be successful on corpus data.

There are numerous results for both the core OT theory and its variants. Many of these results are impressive and analytical. The first was Recursive Constraint Demotion, introduced by Tesar and Smolensky (1998), which was shown to provably learn OT grammars with a polynomial mistake bound. It was subsequently shown that established machine learning techniques could be applied to Harmonic Grammar (Pater, 2008; Bane et al., 2010; Potts et al., 2010) and that these techniques can then be applied to categorical OT (Magri, 2013). Finally, Tesar (2014) proves that, under certain conditions, an OT-based learner can learn both a grammar and underlying forms for a map, given that the map is output-driven.

However, to our knowledge there are no analytical results for optimization-based learners that can learn an entire class of grammars that includes the diversity of opaque maps that we have discussed here. In the absence of analytical results, there are interesting modeling studies which bear on this question. For example, Jarosz (2016) provides results from simulations demonstrating an algorithm that can successfully learn Harmonic Serialism grammars capable of describing some opaque maps. Also, Nazarov and Pater (2017) provide simulations for capable of learning some opaque maps represented in stratal maximum entropy grammar. While results based on specific models tested with specific input may in the future lead to more general results, they are at present less conclusive than results based on proven theorems.

4.4 Typology

Earlier we explained that many phonological maps, including opaque ones, are ISL for some k. However, as Chandlee (2014), Chandlee et al. (2015a), and Chandlee and Heinz (to appear) explain, neither progressive and regressive iterative spreading nor unbounded consonant and vowel harmony are ISL. Consequently, this theory undergenerates.

This undergeneration problem requires adjusting the theory. However, as Chandlee and Heinz (to appear) explain, the change is not expected to be significant. Chandlee et al. (2015a) show that iterative spreading can be described with a comparable class of functions called *Output Strictly Local functions*, which differ from ISL in that the output is determined based on a bounded window of the previous *output* rather than the *input*.

For long-distance phenomena, an approach is readily available based on previous work on phonotactics and the Subregular Hierarchy, a hierarchy of formal languages that are all properly contained by (i.e., less complex than) the regular languages (Rogers and Pullum, 2011; Rogers et al., 2013). Heinz (2010) argued that long-distance phonotactic constraints can be characterized by Strictly Piecewise (SP) languages. Another, related view holds that long-distance phonotactics can be characterized as Tier-based Strictly Local (TSL) languages (Heinz et al., 2011; McMullin, 2016). In the same way that the ISL functions are based on the notion of locality that defines the SL languages, there are likely to be functional counterparts to the SP and TSL classes. These classes are expected to characterize long-distance maps, though the precise nature of these functions is an area of current research. All together, this means that the current theory can be modified as follows to address the undergeneration problem: phonological grammars are not just ISL functions, but rather some combination of ISL, OSL, SP, and TSL functions. What these functions have in common is they are all *subregular*.

As for overgeneration, one of the ways in which grammars can overgenerate is to permit maps like Majority Rules (MR) vowel harmony and Sour Grapes (SG) vowel harmony (Baković, 2000; Finley, 2008; Heinz and Lai, 2013). ISL maps do not overgenerate in this way. This is because 1) every ISL function by definition is also both a regular relation and a subsequential function, and 2) it has been proven that MR maps are not regular relations and SG maps are not subsequential (Heinz and Lai, 2013).

Turning to the typological predictions of rule-based grammars, it is generally understood that ordered rules do not undergenerate. They also can describe opaque maps, despite some criticisms (Baković, 2007, 2011). With respect to overgeneration, the SG pattern is regular and therefore describable with an ordered list of rules according to Kaplan and Kay's analysis. On the other hand, rule-based grammars cannot generate non-regular maps like MR.

Turning to optimization-based theories, it is well-known that classic OT cannot generate many opaque maps (Idsardi, 1998, 2000; McCarthy, 2007; Buccola, 2013)—as discussed by Baković (2007), OT analyses of non-gratuitous feeding and cross-derivational feeding are possible and straightforward. In response, many adjustments to classic OT have been proposed, including constraint conjunction (Smolensky, 2006), sympathy theory (McCarthy, 1999), turbidity theory (Goldrick, 2000), output-to-output correspondence (Benua, 1997), stratal OT (Kiparsky (1998), Kiparsky (2000), Bermúdez-Otero (forthcoming), candidate chains (McCarthy, 2007), Harmonic Serialism (McCarthy, 2000), targeted constraints (Wilson, 2000), contrast preservation (Lubowicz, 2003), comparative markedness (McCarthy, 2003), and serial markedness reduction (Jarosz, 2014). See McCarthy (2007) for a review, meta-analysis, and more references to these proposals.

These approaches all invoke different representational schemes, constraint types, and/or architectural changes to classic OT. In comparison, *no special modifications are needed* to establish the ISL nature of the opaque maps we have studied in this paper. Furthermore, the typological and learnability ramifications of these changes to classic OT are not yet well-understood in many cases, though they are the subject of current research (e.g., Nazarov and Pater, 2017). For instance, it is not obvious whether the results of Recursive Constraint Demotion carry over from classic OT to Stratal OT.

It is also the case that classic OT overgenerates. This is not controversial. It has been shown that non-regular maps are obtainable via the interaction of very simple constraints (Frank and Satta, 1998; Riggle, 2004; Gerdemann and Hulden, 2012; Heinz and Lai, 2013). MR vowel harmony is one example (Baković, 2000; Finley, 2008). Hansson (2007) and McMullin and Hansson (2016) draw similar conclusions with respect to Agreement By Correspondence. Gerdemann and Hulden (2012)'s example is as follows:

$$\textsc{Ident},\ \textsc{Dep} \gg {}^{*}\mathrm{ab} \gg \textsc{Max}$$

Under this ranking, an input string $a^n b^m$ maps to a^n if $m < n$, but it maps to b^m if $n < m$. No FST can compute this function (i.e., it is non-regular).

Maps like MR and the one above make computations based on global access to information in the string. In this respect, optimization misses an important generalization. When computing the output of phonological maps, the *necessary information* is contained within *sub-structures of bounded size*, which, as this paper shows, are often present in the *input representation.* This is neither expected nor predicted under global optimization. On the other hand, it is one of the defining characteristics of ISL functions.

To summarize, both rule-based and optimization-based grammars overgenerate in a way that ISL maps do not. Additionally, optimization-based grammars and ISL maps undergenerate when compared to rule-based grammars, but in different ways. ISL maps cannot account for iteration and long-distance phenomena. Optimization-based grammars struggle with opaque maps as witnessed by the past two decades of research in this area. We believe the prospects for extending subregular functions to include long-distance phenomena are bright and this line of research ought to be pursued.

Figure 4 summarizes the typological results. A successful theory of phonology ought to include patterns like Bedouin Raising (BR)[16], Bedouin Opacity (BO), and Consonant

[16]Bedouin Raising here refers to just the vowel raising rule that participates in both the counterfeeding-

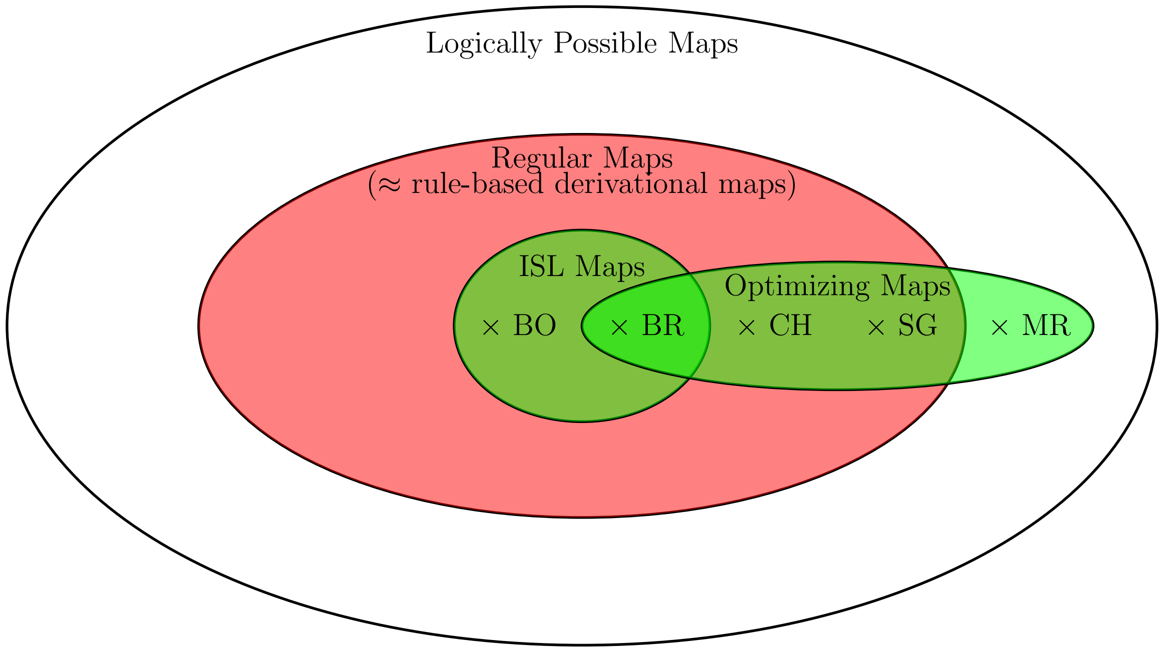


Figure 4: Alignment of typological predictions of rule-based theories, classic OT, and Subregular Theory. BO=Bedouin Opacity, BR=Bedouin Raising, CH = Consonant Harmony, SG = Sour Grapes, MR = Majority Rules.

Harmony (CH), while excluding Sour Grapes harmony (SG) and Majority Rules harmony (MR). While no such theory exists to our knowledge, we hope to have convinced readers that extending ISL maps in the ways mentioned above is a route worthy of further research.

4.5 Summary of theory comparison

A theory of phonology which characterizes UR-SR maps as ISL functions provides theoretically strong results for learnability, recognition, and generation. With respect to typology, certain types of overgeneration are avoided, such as MR and SG harmony. On the other hand, the theory cannot generate long-distance maps, though it could be modified to include them by generalizing additional subregular language classes to functions.

Rule-based theories on the other hand do not undergenerate at all. They do overgenerate SG harmony, but not MR harmony as the former but not the latter is regular. While some learnability results exist, they were not thoroughly pursued, in part due to the advent of constraint-based theories. Lastly, due to their known correspondence with FSTs, rule-based grammars can also be successfully used for both generation and recognition.

Optimization-based theories have theoretical results for generation and recognition under certain assumptions even if they are not widely adopted in practice. There are also learnability results for a variety of optimization frameworks. With respect to typology, there is both overgeneration and undergeneration. It is interesting to observe that the

on-environment (rule (18-a)) and counterfeeding-on-focus (rule (20-b)) maps in Sections 5.1 and 5.2, respectively.

signature of a successful OT analysis is when complex phenomena are understood as the interaction of simple constraints. But the overgeneration problem is precisely this: complex but strange phenomena emerging from the interaction of simple constraints. One approach is to put the burden of explanation on CON: the right set of constraints will generate the attested patterns but not ones like MR or SG. This is a reasonable approach but it misses the forest for the trees. Under this approach, the choice of CON is blunting the power of optimization to avoid generating nonregular maps. There is an important typological generalization—that phonological maps are regular—whose explanation is obtusely stated under this line of thinking.

5 Conclusion

This paper has shown that a range of phonological generalizations labeled as opaque all have the computational property of being Input Strictly Local. This means that, despite the challenges some types of opacity have presented for constraint-based theories like classic OT, in computational terms this set of opaque maps is very limited in its complexity.

The result that many opaque maps are ISL is also significant because the ISL functions as a class have several advantages for a theory of phonology: they are restrictive, they are learnable, and they have a clear cognitive interpretation. While they cannot account for long-distance phenomena, there is precedent from the study of other subregular formal languages which indicates that ISL maps can be extended in the right sorts of ways. The results reported here increase the viability that subregular properties such as input strict locality speak directly to the computational nature of phonology. As such, whether these subregular properties will ultimately supplant optimization as the core computational component of phonological grammars merits further study.

Appendix I: Abstract characterization of ISL functions

In §2.2 of the paper, we define ISL functions in terms of finite-state transducers. Here we provided an alternative, abstract characterization grounded in formal language theory. This characterization is abstract in the sense that it identifies a property of the *extension* of an ISL function, meaning the set of string pairs. This particular characterization proves useful for showing that a given function is *not* ISL. We will take advantage of this usefulness to prove that the ISL functions (as defined in Chandlee 2014) are not closed under composition.

First, some preliminary notation must be explained. The length of a string w is $|w|$. Recall that the empty string of length zero is denoted λ. If $w = uv$ let $u^{-1}w = v$ (so v is the string obtained by stripping u from the beginning of w). The set of prefixes of w, $\texttt{Pref}(w)$, is $\{p \in \Sigma^* \mid (\exists s \in \Sigma^*)[w = ps]\}$, and the set of suffixes of w, $\texttt{Suff}(w)$, is $\{s \in \Sigma^* \mid (\exists p \in \Sigma^*)[w = ps]\}$. The longest common prefix of a set of strings S, denoted $\texttt{lcp}(S)$, is $p \in \cap_{w\in S}\texttt{Pref}(w)$ such that $\forall p' \in \cap_{w\in S}\texttt{Pref}(w), |p'| < |p|$. Note for any set S the $\texttt{lcp}$ is unique and may be λ.

With the $\texttt{lcp}$, we can now define the language-theoretic concept of 'tails' of a function f as follows: $\texttt{tails}_f(x) = \{(y, v) \mid f(xy) = uv \wedge u = \texttt{lcp}(f(x\Sigma^*))\}$. Informally, the tails of a string x given a function f is the function which maps strings y to v where v is the output of $f(xy)$ after the longest common prefix of $f(x\Sigma^*)$ has been factored out.

Finally, we can define ISL functions abstractly in terms of the qualities of their tails. A function f is ISL if there is an integer k such that for all $u_1, u_2 \in \Sigma^*$, if $\texttt{Suff}^{k-1}(u_1) = \texttt{Suff}^{k-1}(u_2)$ then $\texttt{tails}_f(u_1) = \texttt{tails}_f(u_2)$. In other words, if two strings share a $(k-1)$-long suffix, they must have the same tails.

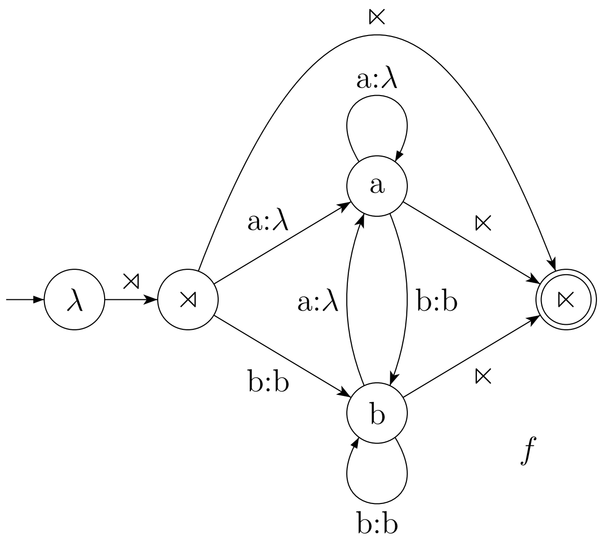
Using this characterization of ISL, we can now show that the ISL functions are not closed under composition. The counterexample in Figure 5 serves as proof of this fact.[17]

The 2-ISL function f on the left side of Figure 5 maps the inputs a^k and ba^k to λ and b, respectively, for all values of k. If these outputs are given as inputs to the 2-ISL function g on the right side of the figure, it would map them to c and b, respectively. This means $g \circ f(a^k) = c$ and $g \circ f(ba^k) = b$. Clearly, ba^k and a^k share a suffix of length $k - 1$, so if $g \circ f$ were a k-ISL function then they should also share the same tails. But they do not, since (λ, c) is in the tails of a^k and (λ, λ) is in the tails of ba^k. Thus this function is not k-ISL, and since k here is arbitrary we conclude that the function is not ISL for any value of k.

Appendix II: Additional analyses

Here we provide ISL analyses for three additional opaque generalizations, to add to the evidence presented in §3 that opaque interactions preserve the ISL property.

[17]We thank Rémi Eyraud for this example.



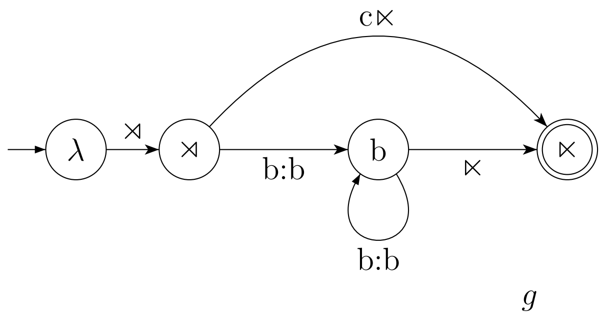


Figure 5: Counterexample to ISL closure under composition

5.1 Counterfeeding-on-environment in Bedouin Arabic

McCarthy (1999) analyzes two generalizations in Bedouin Arabic as **counterfeeding-on-environment** and **counterfeeding-on-focus**. Counterfeeding-on-environment is exemplified in Bedouin Arabic in (17). The two processes that interact are shown in (18).[18]

(17) Bedouin Arabic (McCarthy, 1999)
/badw/ $\mapsto$ [badu], 'Bedouin'

(18) a. a $\rightarrow$ i / ___ CV
b. G $\rightarrow$ V / C ___ #

This is a case of apparent underapplication: (18-a) does not raise the /a/ in [badu] because it isn't until the application of (18-b) that it falls into the raising environment.

Another way of stating this is that an /a/ is raised before an input /CV/ sequence but not before an input /CG/ sequence. Our scanning window then only needs to pick out an /aCV/ sequence to determine whether or not to raise /a/, making this map 3-ISL.

The following input/output table illustrates this. First, as shown in row (a) of Table 15, an input /a/ is output as λ, regardless of the preceding 2-suffix (X here is a variable representing any symbol). As before, this is because it is necessary to see what comes next before deciding whether the output of this /a/ should be [a] or [i]. Similarly, as in row 15b, an input /C/ following an /a/ must also be output as λ, as its output also cannot yet be determined.

The output can be determined when the 2-suffix is /aC/. When the following input

[18]In McCarthy (1999), the raising rule is actually presented as: a $\rightarrow$ i / ___ σ. We use CV for σ to simplify the exposition. Modeling the more general version of the rule in which the syllable that follows the target vowel could take more than one shape would still be an ISL map, though possibly for a higher value of k.

	2-suff	Input	Output		2-suff	Input	Output		2-suff	Input	Output
a.	XX	a	λ	e.	aC	G	aC	i.	Xa	G	aG
b.	Xa	C	λ	f.	aC	a	aC	j.	CG	C	GC
c.	aC	V	iCV	g.	XC	G	λ	k.	CG	V	GV
d.	aC	C	aCC	h.	XV	G	G	l.	CG	⋉	V

Table 15: Abbreviated input/output table for Bedouin Arabic counterfeeding-on-environment

is a /V/, completing the target sequence /aCV/ for raising, the corresponding output is [iCV], as shown in row (15c). When the input is anything else, such as a /C/ or a /G/, the /a/ of the /aC/ sequence is output without being raised. Note that for an input /a/ or /G/ only the 2-suffix itself is output; this is because the /a/ and /G/ in turn need to be held in order to see what their corresponding output is.

The other process in this interaction is simple. An input /G/ is output as λ when following a /C/ (row 15g) because again its actual output cannot yet be determined. It is output faithfully after a V, since it is clearly not in the environment for vocalization (row 15h). It is also output faithfully after /a/ (row 15i), along with /a/ itself which is now known to not be in the environment for raising.

The output for a /G/ following a C is then determined by the following input symbol. When an input /C/ follows the 2-suffix /CG/ (row 15j), the output is [GC], as the glide is not word-final and therefore should not be vocalized. Likewise for an input /V/ following /CG/ (row 15k). In contrast, the output of a /⋉/ following an input /CG/ results in the /G/ being realized as [V] (row 15l).

To sum up, this interaction of processes is a 3-ISL map; the relationship between the input and the output can be determined by looking at the 2-suffix of each input symbol. The derivations in Table 16 show how the above input/output tables obtain the correct outputs for /CaCV/ (for which raising occurs), /CVCG/ (for which vocalization occurs), and /CaCG/ (in which vocalization, but not raising, occurs—this corresponds to /badw/ 'Bedouin' in (17)) using a scanning window of size 3.

5.2 Counterfeeding-on-focus in Bedouin Arabic

Bedouin Arabic also provides an example of counterfeeding-on-focus. An [i] created by the application of the raising rule in (20-b) (repeated from (18-a)) escapes deletion by the preceding syncope rule in (20-a).

(19) Bedouin Arabic (McCarthy, 1999)
/katab/ ↦ [kitab], 'he wrote'

(20) a. $i \rightarrow \emptyset$ / __ CV
b. $a \rightarrow i$ / __ CV

This map is also 3-ISL. The crucial input sequences are /iCV/ and /aCV/, which are

	/CaCV/→[CiCV]
	⋊ C a C V ⋉ ⋊ C
(15a)	⋊ C a C V ⋉ ⋊ C λ
(15b)	⋊ C a C V ⋉ ⋊ C λ λ
(15c)	⋊ C a C V ⋉ ⋊ C λ λ iCV
	⋊ C a C V ⋉ ⋊ C λ λ iCV ⋉
	= CiCV

	/CVCG/→[CVCV]
	⋊ C V C G ⋉ C
	⋊ C V C G ⋉ ⋊ C V
	⋊ C V C G ⋉ ⋊ C V C
(15g)	⋊ C V C G ⋉ ⋊ C V C λ
(15l)	⋊ C V C G ⋉ ⋊ C V C λ V⋉
	= CVCV

	/CaCG/→[CaCV]
	⋊ C a C G ⋉ C
(15a)	⋊ C a C G ⋉ C λ
(15b)	⋊ C a C G ⋉ C λ λ
(15e)	⋊ C a C G ⋉ C λ λ aC
(15l)	⋊ C a C G ⋉ C λ λ aC V⋉
	= CaCV

Table 16: Example derivations for Bedouin raising and vocalization

output as [CV] and [iCV], respectively. The fact that ISL maps focus entirely on the input allows them to easily distinguish input /i/'s, which are subject to deletion, from the [i]'s derived by the raising rule, which survive in the output. Using the alphabet {i, a, u, C}, we represent this map in Table 17.

Rows (17a)-(17h) show the outputs for input /a/. Rows 17a-17d) show that the output of /a/ is λ when it is word initial (17a), follows an initial C or uC sequence (17b and 17c), or follows the vowel u (17d). Rows (17e)-(17h) show the non-empty outputs for input /a/. When the 2-suffix is /Ca/, corresponding to the input sequence /Caa/, the output is [a]. This is because the first /a/ can safely be output as [a], because it is not in the raising context. The second /a/, however, is now held, since its fate cannot yet be determined. When the 2-suffix is /Ci/, the output is [i] because the /i/ was being held but is now known to not be in the context for deletion (because it is followed by /a/ instead of C). The /a/ itself, however, is now held. When the 2-suffix is /aC/, the output reflects raising, because the /a/ in the 2-suffix is now known to be in the raising context (i.e., /aCa/). But again the input /a/ is held. Lastly, when the 2-suffix is /iC/, the output reflects deletion, because the /i/ in the 2-suffix is now known to be in the deletion context (i.e., /iCa/) (and again the input /a/ is held). Rows (17i)-(17p) show the outputs when the input is /i/, which shows an identical pattern to /a/.

The remaining rows show the outputs for inputs C, u, and ⋉. Recall that when the suffix is /Ca/ or /Ci/ the /a/ or /i/ is in a state of being 'held' until it can be determined whether to raise or delete, respectively. An input /C/ that follows one of these suffixes (rows 17q and 17r) must also be held, since the output cannot be decided until it is

	2-suff	Input	Output		2-suff	Input	Output		2-suff	Input	Output
a.	⋊	a	λ	k.	uC	i	λ	u.	Ca	u	au
b.	⋊C	a	λ	l.	Cu	i	λ	v.	Ci	u	iu
c.	uC	a	λ	m.	Ca	i	a	w.	aC	u	iCu
d.	Cu	a	λ	n.	Ci	i	i	x.	iC	u	Cu
e.	Ca	a	a	o.	aC	i	iC	y.	Xa	⋉	a⋉
f.	Ci	a	i	p.	iC	i	C	z.	Xi	⋉	i⋉
g.	aC	a	iC	q.	Ca	C	λ	aa.	aC	⋉	aC⋉
h.	iC	a	C	r.	Ci	C	λ	ab.	iC	⋉	iC⋉
i.	⋊	i	λ	s.	aC	C	aCC				
j.	⋊C	i	λ	t.	iC	C	iCC				

Table 17: Abbreviated input/output table for Bedouin Arabic counterfeeding-on-focus

determined whether the next symbol is a vowel. If a /C/ input instead follows either of the 2-suffixes /aC/ or /iC/ (rows 17s and 17t), what was being held is returned unchanged, along with the input /C/ itself. When an input /u/ follows /Ca/ or /Ci/ (rows 17u and 17v), those vowels are returned unchanged (as we now know neither is in its respective context for being changed), followed by [u] itself. An /u/ following /aC/ or /iC/ (rows 17w and 17x), however, completes the environment for raising or deletion, and so the outputs reflect those changes: [iCu] and [Cu], respectively.

As for ⋉ inputs, when ⋉ is read at a point at which some input is being 'held', the held string is the output of ⋉. One way to think of this is that the substring that comprises the rule's structural description has been partially found. If the string ends before it is found in its entirety (i.e., in this example if the string ends in one of /a/, /i/, /aC/, or /iC/), then the output of ⋉ 'returns' the held portion. In rows (y) and (z) of Table 17, Xa and Xi abbreviate all 2-suffixes in which any symbol is followed by /a/ and /i/, respectively (e.g., Xa = {Ca, ua, aa, ia}).

Table 18 provides example derivations for Bedouin raising and deletion.

5.3 Self-destructive feeding in Turkish

Finally, we look at an example from Turkish (Sprouse, 1997) of **self-destructive feeding**, in which one process feeds another, and the environment of the first is partially destroyed by the application of the second. This results in apparent overapplication of the first process. In Turkish, epenthesis (22-a) occurs between two consonants at the end of the word. Additionally, an intervocalic /k/ deletes before a morpheme boundary (22-b).

(21) Turkish (Sprouse, 1997)
/bebek+n/ $\mapsto$ [bebein], 'your baby'

(22) a. $\emptyset \rightarrow$ i / C__C#

	/CaCu/→[CiCu]		/CiCu/→[CCu]		/CaCi/→[CiCi]
	⋊ C a C u ⋉ ⋊ C		⋊ C i C u ⋉ ⋊ C		⋊ C a C i ⋉ ⋊ C
(17b)	⋊ C a C u ⋉ ⋊ C λ	(17j)	⋊ C i C u ⋉ ⋊ C λ	(17b)	⋊ C a C i ⋉ ⋊ C λ
(17q)	⋊ C a C u ⋉ ⋊ C λ λ	(17r)	⋊ C i C u ⋉ ⋊ C λ λ	(17q)	⋊ C a C i ⋉ ⋊ C λ λ
(17w)	⋊ C a C u ⋉ ⋊ C λ λ iCu	(17x)	⋊ C i C u ⋉ ⋊ C λ λ Cu	(17o)	⋊ C a C i ⋉ ⋊ C λ λ iC
	⋊ C a C u ⋉ ⋊ C λ λ iCu ⋉		⋊ C i C u ⋉ ⋊ C λ λ Cu ⋉	(17z)	⋊ C a C i ⋉ ⋊ C λ λ iC i⋉
	= CiCu		= CCu		= CiCi

Table 18: Example derivations for Bedouin Arabic raising and deletion

b. $k \rightarrow \emptyset$ / V__+V

As shown in (21), epenthesis feeds deletion, but the input /k/ that triggers the epenthesis is itself deleted, obscuring the reason epenthesis occurred.

This map is 5-ISL. In the following, K represents dorsal consonants, C represents all other consonants, and V represents vowels. The alphabet also includes + for morpheme boundaries, which are a necessary part of the deletion environment. If needed, morpheme boundaries can be removed from the output in a context-free manner by outputting all input +'s as λ no matter what the 4-suffix, as indicated in row (a) of Table 19.

An input K following a vowel is output as λ, as it may potentially be in the deletion environment (row 19b). If anything besides /+/ follows a K, then the K can be output, as it is not in the deletion environment (which recall explicitly includes the morpheme boundary). This is illustrated in rows (19c)-(19f).

Any C or K following another C or K is output as λ, as shown in rows (19g)-(19n). (Note that this occurs regardless of whether or not a morpheme boundary intervenes.) If the following input symbol is /⋉/, then we have the environment for epenthesis, and [iC] or [iK] is output accordingly, as shown in rows (19o)-(19t). Note in particular rows (19q) and (19t), in which the 4-suffix includes a post-vocalic K that precedes a morpheme boundary followed by a consonant. The output for ⋉ in these rows reflects both epenthesis and deletion (recall that the postvocalic K is being held, so its absence from the output of ⋉ achieves the deletion).

The derivation of /CVK+C/→[CViC] (corresponding to example (21)) is given in

	4-suff	Input	Output		4-suff	Input	Output
a.	XXXX	+	λ	k.	XXXC	K	λ
b.	XXXV	K	λ	l.	XXXK	K	λ
c.	XXVK	V	KV	m.	XXC+	K	λ
d.	XXVK	C	KC	n.	XXK+	K	λ
e.	XXVK	K	K	o.	XXCC	⋉	iC⋉
f.	XXVK	⋉	K⋉	p.	XC+C	⋉	iC⋉
g.	XXXC	C	λ	q.	VK+C	⋉	iC⋉
h.	XXXK	C	λ	r.	XXCK	⋉	iK⋉
i.	XXC+	C	λ	s.	XC+K	⋉	iK⋉
j.	XXK+	C	λ	t.	VK+K	⋉	iK⋉

Table 19: Abbreviated input/output table for Turkish self-destructive feeding

Table 20. Note that the scanning window is of size 5, which is necessary to identify the situation in which deletion and epenthesis interact, namely a /VK+C⋉/ sequence. Thus, the Turkish self-destructive feeding map is 5-ISL.

/CVK+C/→[CViC]

	⋊	C	V	K	+	C	⋉
	⋊	C					
	⋊	C	V	K	+	C	⋉
	⋊	C	V				
(19b)	⋊	C	V	K	+	C	⋉
	⋊	C	V	λ			
(19a)	⋊	C	V	K	+	C	⋉
	⋊	C	V	λ	λ		
(19j)	⋊	C	V	K	+	C	⋉
	⋊	C	V	λ	λ	λ	
(19q)	⋊	C	V	K	+	C	⋉
	⋊	C	V	λ	λ	λ	iC⋉

=CViC

Table 20: Derivation for Turkish self-destructive feeding

References

Albro, D. (2005). *A Large-Scale, LPM-OT Analysis of Malagasy*. PhD thesis, University of California, Los Angeles.

Baković, E. (2000). *Harmony, Dominance and Control*. PhD thesis, Rutgers University.

Baković, E. (2005). Antigemination, assimilation and the determination of identity. *Phonology*, 22:279–315.

Baković, E. (2007). A revised typology of opaque generalisations. *Phonology*, 24(2):217–259.

Baković, E. (2011). Opacity and ordering. In Goldsmith, J., Riggle, J., and Yu, A. C., editors, *The Handbook of Phonological Theory*, pages 40–67. London: Wiley-Blackwell, 2 edition.

Baković, E. (2013). *Blocking and Complementarity in Phonological Theory*. Equinox, Bristol, CT.

Bane, M., Riggle, J., and Sonderegger, M. (2010). The VC dimension of constraint-based grammars. *Lingua*, 120:1194–1208.

Beesley, K. R. and Karttunen, L. (2003). *Finite State Morphology.* CSLI Publications.

Benua, L. (1997). *Transderivational Identity: Phonological Relations between Words.* PhD thesis, University of Massachusetts, Amherst.

Bermúdez-Otero, R. (in preparation). *Stratal Optimality Theory.* Oxford: Oxford University Press.

Buccola, B. (2013). On the expressivity of Optimality Theory versus ordered rewrite rules. In *Proceedings of Formal Grammar 2012 and 2013*, volume 8036 of *Lecture Notes in Computer Science*, pages 142–158. Berlin: Springer-Verlag.

Chandlee, J. (2014). *Strictly Local Phonological Processes.* PhD thesis, U. of Delaware.

Chandlee, J., Athanasopoulou, A., and Heinz, J. (2012). Evidence for classifying metathesis patterns as subsequential. In *The Proceedings of the 29th West Coast Conference on Formal Linguistics*, pages 303–309. Cascillida Press.

Chandlee, J., Eyraud, R., and Heinz, J. (2014). Learning strictly local subsequential functions. *Transactions of the Association for Computational Linguistics*, 2:491–503.

Chandlee, J., Eyraud, R., and Heinz, J. (2015a). Output strictly local functions. In *Proceedings of the 14th Meeting on the Mathematics of Language (MoL 2015)*, pages 112–125, Chicago, USA.

Chandlee, J. and Heinz, J. (2012). Bounded copying is subsequential: Implications for metathesis and reduplication. In *Proceedings of the 12th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 42–51, Montreal, Canada. Association for Computational Linguistics.

Chandlee, J. and Heinz, J. (2016). Computational phonology. In Aronoff, M., editor, *Oxford Research Encylcopedia of Linguistics.* Oxford University Press.

Chandlee, J. and Heinz, J. (to appear). Strict locality and phonological maps. *Linguistic Inquiry.*

Chandlee, J., Jardine, A., and Heinz, J. (2015b). Learning repairs for marked structures. In *Proceedings of the 2014 Meeting on Phonology, MIT*. Linguistics Association of America.

Chandlee, J. and Lindell, S. (to appear). A logical characterization of strictly local functions. In Heinz, J., editor, *Doing Computational Phonology.* Oxford.

Chomsky, N. and Halle, M. (1968). *The Sound Pattern of English.* Harper & Row.

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

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

Eisner, J. (1997). Efficient generation in primitive Optimality Theory. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 313–320, Madrid.

Finley, S. (2008). *The formal and cognitive restrictions on vowel harmony*. PhD thesis, Johns Hopkins University.

Frank, R. and Satta, G. (1998). Optimality theory and the generative complexity of constraint violability. *Computational Linguistics*, 24(2):307–315.

Gainor, B., Lai, R., and Heinz, J. (2012). Computational characterizations of vowel harmony patterns and pathologies. In *The Proceedings of the 29th West Coast Conference on Formal Linguistics*, pages 63–71.

Gerdemann, D. and Hulden, M. (2012). Practical finite state Optimality Theory. In *Proceedings of the 10th International Workshop on FSMNLP*, pages 10–19. ACL.

Gerdemann, D. and van Noord, G. (2000). Approximation and exactness in finite state optimality theory. In *Proceedings of the Fifth Meeting of the ACL Special Interest Group in Computational Phonology*, pages 34–45.

Gildea, D. and Jurafsky, D. (1996). Learning bias and phonological-rule induction. *Computational Linguistics*, 24(4).

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

Goldrick, M. (2000). Turbid output representations and the unity of opacity. In *Proceedings of NELS 2000*.

Hansson, G. O. (2007). Blocking effects in Agreement by Correspondence. *Linguistic Inquiry*, 38(2):373–372.

Hayes, B., Kirchner, R., and Steriade, D. (2004). *Phonetically Based Phonology*. Cambridge: Cambridge University Press.

Heinz, J. (2010). Learning long-distance phonotactics. *LI*, 41:623–661.

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

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

Heinz, J. and Lai, R. (2013). Vowel harmony and subsequentiality. In Kornai, A. and Kuhlmann, M., editors, *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pages 52–63, Sofia, Bulgaria.

Heinz, J., Rawal, C., and Tanner, H. G. (2011). Tier-based strictly local constraints for phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 58–64, Portland, Oregon, USA. Association for Computational Linguistics.

Heinz, J. and Riggle, J. (2011). Learnability. In van Oostendorp, M., Ewen, C., Hume, B., and Rice, K., editors, *Blackwell Companion to Phonology*. Wiley-Blackwell.

Hyman, L. (1975). *Phonology: Theory and Analysis.* New York: Holt, Rinehart and Winston.

Idsardi, W. (1998). Tiberian Hebrew spirantization and phonological derivations. *Linguistic Inquiry*, 29:37–73.

Idsardi, W. (2000). Clarifying opacity. *The Linguistic Review*, 17:199–218.

Jäger, G. (2002). Gradient constraints in finite state OT: The unidirectional and the bidirectional case. In Kaufmann, I. and B.Stiebels, editors, *More than Words: A Festschrift for Dieter Wunderlich*, pages 299–325. Berlin: Akademie Verlag.

Jain, S., Osherson, D., Royer, J. S., and Sharma, A. (1999). *Systems That Learn: An Introduction to Learning Theory (Learning, Development and Conceptual Change).* The MIT Press, 2nd edition.

Jardine, A. (2016). Computationally, tone is different. *Phonology.* To appear.

Jardine, A., Chandlee, J., Eyraud, R., and Heinz, J. (2014). Very efficient learning of structured classes of subsequential functions from positive data. In Clark, A., Kanazawa, M., and Yoshinaka, R., editors, *JMLR: Workshop and Conference Proceedings*, volume 34, pages 94–108.

Jarosz, G. (2014). Serial markedness reduction. In *Proceedings of 2013 Annual Meetings on Phonology, Amherst, MA*, volume 1(1). Linguistic Society of America.

Jarosz, G. (2016). Learning opaque and transparent interactions in harmonic serialism. In *Proceedings of the 2015 Annual Meetings on Phonology, Vancouver, BC.* Linguistic Society of America.

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

Johnson, M. (1984). A discovery procedure for certain phonological rules. In *Proceedings of the 10th International Conference on Computational Linguistics and the 22nd Annual Meeting of the ACL*, pages 344–347.

Kager, R. (1999). *Optimality Theory.* Cambridge University Press.

Kaplan, R. and Kay, M. (1994). Regular models of phonological rule systems. *Computational Linguistics*, 20:331–78.

Karttunen, L. (1993). Finite-state constraints. In Goldsmith, J., editor, *The Last Phonological Rule*, pages 173–194. Chicago: The University of Chicago Press.

Karttunen, L. (1998). The proper treatment of optimality in computational phonology. In *FSMNLP'98*, pages 1–12. International Workshop on Finite-State Methods in Natural Language Processing, Bilkent University, Ankara, Turkey.

Kavitskaya, D. and Staroverov, P. (2010). When an interaction is both opaque and transparent: the paradox of fed counterfeeding. *Phonology*, 27:255–288.

Kawahara, S. (2015). A catalogue of phonological opacity in Japanese. *Reports of the Keio Institute of Cultural and Linguistic Studies*, 46:145–174.

Kearns, M. and Vazirani, U. (1994). *An Introduction to Computational Learning Theory*. MIT Press.

Kiparsky, P. (1971). Historical linguistics. In Dingwall, W. O., editor, *A survey of linguistic science*, pages 576–642. College Park: University of Maryland Linguistics Program.

Kiparsky, P. (1973). Abstractness, opacity, and global rules. In Fujimura, O., editor, *Three dimensions in linguistic theory*, pages 57–86. Tokyo: TEC.

Kiparsky, P. (1998). Paradigm effects and opacity. Stanford University.

Kiparsky, P. (2000). Opacity and cyclicity. In Ritter, N. A., editor, *A review of Optimality Theory. Special issue of The Linguistic Review*, volume 17, pages 351–65.

Legendre, G., Miyata, Y., and Smolensky, P. (1990). Harmonic grammar: A formal multilevel connectionist theory of linguistic well formedness: Theoretical foundations. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 388–395, Cambridge, MA.

Lubowicz, A. (2003). *Contrast preservation in phonological mappings*. PhD thesis, University of Massachusetts, Amherst.

Lundskaer-Nielsen, T. and Holmes, P. (2011). *Danish: An Essential Grammar*. Routledge Essential Grammars. Routledge.

Magri, G. (2013). HG has no computational advantages over OT: towards a new toolkit for computational OT. *Linguistic Inquiry*, 44.4:569609.

McCarthy, J. and Prince, A. (1996). Faithfulness and identity in prosodic morphology. In Kager, R., van der Hulst, H., and Zonneveld, W., editors, *The prosody-morphology interface*. Cambridge: Cambridge University Press.

McCarthy, J. J. (1999). Sympathy and phonological opacity. *Phonology*, 16:331–399.

McCarthy, J. J. (2000). Harmonic serialism and parallelism. In Hirotani, M., Coetzee, A., Hall, N., and Kim, J., editors, *NELS 30: Proceedings of the 30th Annual Meeting of the North East Linguistic Society*, pages 501–524. GLSA, University of Massachusetts Amherst.

McCarthy, J. J. (2003). Comparative markedness. *Theoretical Linguistics*, 29.

McCarthy, J. J. (2007). *Hidden Generalizations: Phonological Opacity in Optimality Theory*. London: Equinox.

McMullin, K. (2016). *Tier-Based Locality in Long-Distance Phonotactics: Learnability and Typology*. PhD thesis, University of British Columbia.

McMullin, K. and Hansson, G. O. (2016). Long-distance phonotactics as Tier-Based Strictly 2-Local languages. In Albright, A. and Fullwood, M. A., editors, *Proceedings of the Annual Meeting on Phonology 2015*.

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

Nazarov, A. and Pater, J. (2017). Learning opacity in stratal maximum entropy grammar. *Phonology*, 34(2):299324.

Odden, D. (2005). *Introducing phonology*. Cambridge: Cambridge University Press.

Oncina, J., García, P., and Vidal, E. (1993). Learning subsequential transducers for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:448–458.

Pater, J. (2008). Gradual learning and convergence. *Linguistic Inquiry*, 39:334–345.

Pater, J. (2012). Serial Harmonic Grammar and Berber syllabification. In Borowsky, T., Kawahara, S., Shinya, T., and Sugahara, M., editors, *Prosody Matters: Essays in Honor of Elisabeth O. Selkirk*, pages 43–72. London, Equinox Press.

Potts, C., Pater, J., Jesney, K., Bhatt, R., and Becker, M. (2010). Harmonic Grammar with linear programming: From linear systems to linguistic typology. *Phonology*, 27:77–117.

Prince, A. and Smolensky, P. (1993). Optimality Theory: Constraint interaction in generative grammar. Technical Report 2, Rutgers University Center for Cognitive Science.

Prince, A. and Smolensky, P. (2004). *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell.

Riggle, J. (2004). *Generation, Recognition, and Learning in Finite State Optimality Theory*. PhD thesis, UCLA.

Rogers, J., Heinz, J., Fero, M., Hurst, J., Lambert, D., and Wibel, S. (2013). Cognitive and sub-regular complexity. In *Formal Grammar*, volume 8036 of *Lecture Notes in Computer Science*, pages 90–108. Springer.

Rogers, J. and Pullum, G. (2011). Aural pattern recognition experiments and the sub-regular hierarchy. *Journal of Logic, Language and Information*, 20:329–342.

Sanders, N. (2003). *Opacity and Sound Change in the Polish Lexicon.* PhD thesis, University of California Santa Cruz.

Smolensky, P. (2006). Optimality in phonology II: Harmonic completeness, local constraint conjunction, and feature domain markedness. In *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar*, volume II: Linguistic and Philosophical Implications, chapter 14. MIT Press.

Sprouse, R. (1997). A case for enriched inputs. Available as ROA-193 from the Rutgers Optimality Archive.

Tesar, B. (2014). *Output-Driven Phonology: Theory and Learning.* Cambridge Studies in Linguistics.

Tesar, B. and Smolensky, P. (1993). The learnability of Optimality Theory: an algorithm and some basic complexity results. Unpublished ms., University of Colorado, Boulder.

Tesar, B. and Smolensky, P. (1996). Learnability in Optimality Theory (long version). Technical Report 96:3, Department of Cognitive Science, Johns Hopkins University.

Tesar, B. and Smolensky, P. (1998). Learnability in Optimality Theory. *Linguistic Inquiry*, 29:229–268.

Weigel, W. F. (2005). *Yowlumne in the Twentieth Century.* PhD thesis, University of California, Berkeley.

Wilson, C. (2000). *Targeted constraints: An approach to contextual neutralization in Optimality Theory.* PhD thesis, Johns Hopkins University.