STRICTLY LOCAL PHONOLOGICAL PROCESSES

by

Jane Chandlee

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Linguistics

Spring 2014

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

STRICTLY LOCAL PHONOLOGICAL PROCESSES

by

Jane Chandlee

Approved: _____
Benjamin Bruening, Ph.D.
Chair of the Department of Linguistics and Cognitive Science

Approved: _____
George H. Watson, Ph.D.
Dean of the College of Arts and Science

Approved: _____
James G. Richards, Ph.D.
Vice Provost for Graduate and Professional Education

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Jeffrey Heinz, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Irene Vogel, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Benjamin Bruening, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Herbert Tanner, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

James Rogers, Ph.D.
Member of dissertation committee

ACKNOWLEDGEMENTS

This is actually the hardest section to write, because there simply are no words to convey my gratitude to the people who have seen me through to this point. I am humbled by the extent of their generosity, dedication, and caring. I'll do what I can, but in truth I'll be thanking them all for years to come.

First, to Jeff Heinz, whose enthusiasm, kindness, and tireless support never cease to amaze me: thank you. From our very first meeting I have never doubted that I found my mentor.

Thanks also to my committee members. To Irene Vogel and Ben Bruening, thank you for setting high standards and inspiring me to meet them by being the kind of scholars I want to be. To Jim Rogers, thanks for your valuable insights and contributions and for helping me to start taking myself seriously as a scholar. And to Bert Tanner, thank you for your support of my work and for opening my eyes to the vast potential of interdisciplinary research.

I would also like to thank the professors I worked with throughout my Master's program at Boston University, for humoring a fiction writer who wanted to learn about linguistics and for giving me the knowledge and confidence to go on to a PhD program. Thanks in particular to Cathy O'Connor, Jonathan Barnes, Paul Hagstrom, and especially Nanette Veilleux of Simmons College.

And last but far from least, thank you to my parents, Eileen and Larry, for taking me seriously when I changed my career plans, for not taking me seriously when I said I couldn't do this, and for never once suggesting that I try to get my job at the bank back.

And now, onward.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This dissertation identifies a strong computational property of phonological and morphological processes with local triggers. It is shown that the input-output mapping that underlies these processes can be modeled with Strictly Local (SL) functions, a previously undefined class of subregular relations. The SL functions, which are divided into two proper subclasses of subsequential functions (the Input SL functions and Output SL functions) are characterized in automata-theoretic terms by combining the properties of subsequential transduction (Mohri, 1997) and the Strictly Local formal languages (McNaughton and Papert, 1971; Rogers and Pullum, 2011; Rogers et al., 2013). Importantly, the property of strict locality is independent of and compatible with both rule- and constraint-based grammatical formalisms, since it holds of the input-output mappings that both formalisms describe. The range of processes that are shown to be Strictly Local includes substitution, deletion, insertion, synchronic metathesis, local partial reduplication, and general affixation. This computational property aids in identifying the set of 'phonologically possible' processes within the larger set of logically possible processes. In addition, a learning algorithm that provably learns the class of ISL functions by using strict locality as an inductive principle is also presented. These combined contributions to typology and learning demonstrate how computational analysis can enhance our understanding of the nature of locality in phonological processes.

Chapter 1

INTRODUCTION

1.1 Restricting Phonology

The starting point of this thesis is a deceptively simple question: what is a possible phonological process? Even acknowledging the wide range of variation we find in the phonologies of the world's languages, the range of *logically possible* variation is much wider. Phonologists have clear intuitions about what is and is not a plausible phonological pattern, yet the precise metric by which plausibility is assessed remains an open question. What exactly distinguishes the subset of attested and expected phenomena - in other words, what is *phonologically possible* - from the larger set of what is logically possible?

This thesis addresses this question from the computational perspective and proposes that phonological processes with local triggers all have a defining property known as *strict locality*. This property is revealed through an analysis of these processes as infinite mappings (i.e., functions) from input to output forms, mappings that can be computed by only keeping track of substrings of bounded length. In this way strict locality is shown to hold over the computation of the mapping, as opposed to the grammatical formalism (e.g., rules or constraints) employed to describe the process in question. This result has two main contributions. First, in describing the input-output mapping directly, the analysis is equally compatible with and independent of the primary theories of generative phonology (i.e., Sound Pattern of English (SPE) (Chomsky and Halle, 1968) and Optimality Theory (OT) (Prince and Smolensky, 1991, 1993, 2004)), both of which describe the same mapping. The result thus establishes a clear boundary for the range of local processes that these theories should and should not

predict. Second, the property of strict locality leads directly to an approach to how these processes can be learned.

Thus far phonological possibility has most often been expressed as phonological naturalness, with 'natural' being variably defined in terms of either universal principles or phonetic motivation. In the latter case, patterns are deemed natural because they facilitate (or 'optimize') articulation or perception (see, for example Hume and Johnson (2001) and Hayes et al. (2004)) or are otherwise amenable to a phonetic explanation (Blevins, 2004). Explanations in the former category point to the widespread occurrence of certain patterns in typologically-diverse languages, with the assumption that their prevalence is due to a bias for UG-supported or unmarked language structures. For example, word-final consonant deletion before consonant-initial words is considered natural (Schane et al., 1974) because it creates the unmarked CV syllables that are permitted in all languages. Naturalness accounts in both camps are supported by experimental studies that suggest that natural generalizations are easier to learn than unnatural ones (Schane et al. (1974); Healy and Levitt (1980); Wilson (2003, 2006); Peperkamp et al. (2006); Seidl and Buckley (2005); Carpenter (2010); Hayes and White (2013), among others). However, other work has shown that generalizations that do not meet such criteria for naturalness not only exist (Bach and Harms, 1972; Hellberg, 1978; Anderson, 1981; Buckley, 2000; Odden, 2007; Blevins, 2008) but can have psychological reality (Pycha et al., 2003; Hayes et al., 2009). Indeed, Buckley (2002, 2003) argues that the apparent ease with which children acquire 'natural' processes is due to their regularity (i.e., 'accessibility'), not their inherent naturalness.

Early on, the concept of naturalness was conflated with formal complexity, as measured by the number of features specified in the rule for a process (Chomsky and Halle, 1968; Anderson, 1974). For example, given a language that includes {b, d, g} in its sound inventory, the observation that the underlying forms /dɑb/ and /dɑd/ are pronounced as [dɑp] and [dɑt], respectively, is consistent with both of the following rules.

(1) a. [-son, +ant] → [-voice] / __#

b. [-son] → [-voice] / __#

In this case, despite the lack of evidence for the surface form of a word ending in /g/, rule (1-b) might be considered more natural because it is formally simpler than rule (1-a) in that it includes fewer features (see Halle (1959, 1961, 1962, 1964)). And indeed it represents a more significant linguistic generalization, in the sense of Chomsky and Halle (1968). Since this measure of formal complexity is inversely related to the number of segments affected by the rule (i.e., a smaller number of features identifies a larger set of segments), its success as a metric of naturalness relies on the psychological reality of feature-based representations (not to mention the correct feature system). In other words, if speakers actually represent processes in terms of segments, then rule (1-a) should be preferred under the assumption that a process that targets fewer segments is simpler. This is why Chomsky (1965) emphasizes that 'it is necessary to devise notations and to restrict the form of rules in such a way that significant considerations of complexity and generality are converted into considerations of length, so that real generalizations shorten the grammar and spurious ones do not' (42). The larger claim is that the feature representation of rules corresponds to the language learner's own evaluation measure, allowing it to select the correct (= formally simpler) generalizations and disregard the 'spurious' ones. It is less clear how to measure the formal complexity of a constraint-based grammar, though measures in terms of restrictiveness have been proposed (Anttila, 1997; Prince and Tesar, 1999; Riggle, 2010).

Note that neither approach to defining phonological plausibility (naturalness and formal complexity) necessarily predicts the non-existence of patterns. Indeed, non-existence cannot be proven. This is why the question of plausibility is often tied to the question of *learnability*. The phonological grammar consists (in part) of those generalizations that the learner has induced from the observed data. Thus the data might reflect both natural and unnatural generalizations, or both formally simpler and formally more complex generalizations, but the biases of the learner will determine

the content of the grammar. The experimental studies cited above provide conflicting evidence for whether a naturalness bias is at work in phonological learning. Fewer studies have tested for a 'simplicity' bias, but the subjects in the study of Pycha et al. (2003) did learn a formally simpler generalization better than a more complex one.[1]

This is where the computational approach can make a real contribution: it identifies precise computational properties that hold of the observed patterns but not of the unattested ones and therefore establishes clear predictions for what patterns will and will not be found cross-linguistically. This idea is not novel. As early as Chomsky (1959) it was observed that 'such purely mathematical investigation may provide deeper insight into the formal properties that distinguish natural languages among all sets of finite strings in a finite alphabet' (139). The approach is to recognize the set of patterns we are interested in characterizing as a proper subset of what is logically possible. (This differs from the natural/unnatural distinction, since those sets are necessarily disjoint.) Using computational analysis, we can narrow in on this subset by establishing the restrictions that make it a proper subset. In this way the result that local processes are in fact Strictly Local extends previous research by Johnson (1972), Koskenniemi (1983), and Kaplan and Kay (1994) that showed that phonological mappings are regular relations. Since the Strictly Local functions will be shown to be a proper subset of regular relations, the central contribution is a better (i.e., more restrictive) characterization of local processes.

Throughout the thesis the term 'process' will be used to refer to differences between an underlying form or UR (i.e., underlying representation, an item from the mental lexicon) and its surface form or SR (i.e., surface representation, the pronounced variant of the lexical item). It is important to keep in mind, however, that the analyses do not assume a rule-based grammatical formalism. It is true that in constraint-based formalisms such as OT and its variants like Harmonic Grammar (HG) (Legendre et al., 1990), and Harmonic Serialism (HS) (McCarthy, 2000a; Pater, 2012), the notion of a

[1] The results of Hayes et al. (2009) also suggest an influence of a simplicity bias, thought the authors note that their study was not designed to test that hypothesis specifically.

'process' does not exist, because differences between the UR and SR are not encoded as 'changes' but rather stem from the interaction of the constraint ranking and the candidate set. Yet, the advent of these theories has not led phonologists to abandon the analysis and discussion of individual patterns, and in fact much of the progress that has been made in OT research has come from showing how it accounts for them. Though it is less clear how to isolate the individual generalizations of an OT grammar, since the UR is matched to the SR in one fell swoop as opposed to a step-by-step derivation, Baković (2013) discusses how a rewrite rule of the form in (2-a) corresponds to the fragment of an OT grammar in (2-b). The context-sensitive change from A to B expressed by the rule corresponds to a preference for whatever faithfulness violations are incurred by that change over the allowance of the marked sequence CAD.

(2) a. $A \rightarrow B \ / \ C \_\_ D$

b. $*CAD \gg FAITH(A \rightarrow B)$

The equivalence schematized in (2) is not meant to downplay the theoretical differences between SPE and OT, differences that have clear and interesting implications for our understanding of mental representations in phonology (Kager, 1999; McCarthy, 2008). Rather, it is meant to justify the methodology of this thesis, which is to abstract away from the *procedure* of the UR $\mapsto$ SR mapping and directly analyze the nature of the mapping itself. This approach, which offers insights into the nature of phonological processes that are not conditioned by the choice of rules versus constraints, is not without precedent. Recently Tesar (2008, 2014) has argued that the majority of phonological input-output mappings are *output-driven*, and this property offers significant benefits to a learner that assumes its target grammar is restricted in this way. Though Tesar's actual learning framework is grounded in OT, the property of being output-driven is uncontroversially not bound to any particular generative theory of phonology. Thus, this thesis will present analyses of single differences between a UR and SR, differences which under a rule-based formalism are described with a single

rule, with the understanding that the overall results are equally compatible with a constraint-based grammatical formalism.

These analyses will support the central claim that a large number and range of phonological and morphological processes have the computational property of strict locality, which will be defined in Chapter 3. This property will be shown to distinguish a process like (3), which has no phonetic motivation but is attested (in Kashaya (Oswalt, 1961, 1976; Buckley, 2000)), from the one in (4), which is not only unattested but would be considered implausible by most if not all phonologists.

(3) Kashaya

a. /i/ → [u] / d __

b. /cad+i/ ↦ [cadú] 'Look!'

(4) /i/ → [u] / x __(x is a string with an even number of obstruents)

Despite the strong intuitions of phonologists regarding the implausibility of a process like (4), it is unclear what actually rules it out. Nothing inherent to either SPE or OT prevents the construction of rules or constraints, respectively, that can describe such a pattern. This is not an oversight, but rather a decision to devote attention to what are viewed as more relevant distinctions, such as the natural/unnatural distinction discussed above. In other words, (4) hasn't been ruled out because it so *clearly* falls outside of the range of what is phonologically possible. Yet that certainty is precisely the point. The property that distinguishes (4) from (3) - strict locality - will be argued to be a *defining* property of local phonological generalizations. This means the impossibility of (4) will be attributed not to a property of rules or constraints, but to a property of the phonological mappings themselves.

This claim of theory-independence is not meant to imply that the theory is somehow extraneous. Indeed, as will be discussed in Chapter 7, the complete picture of what constitutes a possible phonological pattern will surely incorporate several explanatory factors, including computational properties, phonetic naturalness, and other

insights into the mental representations of language that phonological theory can provide. But the proposed property of strict locality will play a key role by delimiting the typological range a phonological theory should predict.

1.2 Locality in Phonology

Strict locality, as a computational property, will be defined in formal terms that draw from previous work in formal language theory and computer science (McNaughton and Papert, 1971; Rogers and Pullum, 2011; Rogers et al., 2013). But locality in general has long been recognized as significant to phonology, in the sense that phonological changes tend to involve adjacent segments. Kenstowicz (1994), for example, remarks on 'the well-established generalization that linguistic rules do not count beyond two' (597). Yet the role of locality in phonology cannot be directly conceived of as adjacency, given the prominence of processes like vowel harmony that involve non-adjacent segments.[2] It is telling, however, that such non-local phenomena have not been used as evidence against the overall local nature of phonology. Rather, the tendency has been to recast non-local patterns in such a way that they become local.

Consider a non-local process in which the target and trigger are separated by some number of intervening segments. A variety of proposals have been put forth for how to 'ignore' this intervening material so that the target and trigger can be treated as adjacent. Most notably, Autosegmental Phonology (Goldsmith, 1976, 1979; Clements, 1976) proposed the separation of certain classes of segments onto their own tiers. For example, given a consonant tier and a vowel tier, a vowel harmony process can be recast as local since the harmonizing vowels are adjacent on their own tier. Other approaches along these lines include the use of metrical structure (Liberman, 1975; Liberman and Prince, 1977) or planes (McCarthy, 1979; Archangeli, 1985; Odden, 1994). Alternatively, intervening segments can be treated as 'transparent' due to underspecification

[2] Though this thesis will only investigate its role in phonology, locality also comes into play in syntax, in which certain constraints (e.g., subjacency) enforce dependencies between the closest possible elements.

of the feature that identifies the target (Steriade, 1987; Paradis and Prunet, 1989, 1991; Shaw, 1991), or through some other means (Jensen, 1974; Halle and Vergnaud, 1978; Kiparsky, 1981; Archangeli and Pulleyblank, 1987, 1994; Anderson and Ewen, 1987; Gafos, 1996; Gafos and Lombardi, 1999; Baković and Wilson, 2000; Heinz, 2010). Lastly, some authors have instead argued that intervening segments do not need to be ignored at all, since they in fact participate in the process (Flemming, 1995; Ní Chiosáin and Padgett, 1997; Bessell, 1998; Walker, 1998; Ní Chiosáin and Padgett, 2001). For example, if consonants also take on the relevant feature in a vowel harmony process, then the harmonizing is in fact local.

The conception of phonological locality proposed in this thesis will be computational in nature, following the work of Heinz (2007, 2009), and it will be defined in terms of *contiguity* rather than strict adjacency. It is not a coincidence that both the rules of SPE and the constraints of OT, HG, HS that are employed to describe local phenomena refer to contiguous substrings of a word. This reliance on contiguous substrings is a reflection of the significance of locality to phonological generalizations. Accordingly, this thesis will propose locality as a defining property of local phonological processes by modeling these processes with a previously undefined class of subregular relations called the Strictly Local functions. Furthermore, it will be shown that the structure of the class of SL functions enables a learning algorithm to learn these phonological processes.

1.3 Outline

Chapter 2 has four main objectives. It first introduces both the empirical and computational perspectives that serve as the backdrop for the analyses presented in the thesis. This includes the conception of phonological processes as functions and the formalism of finite state automata. Second, it presents the formal languages known as Strictly Local languages, which form the basis for the property of strict locality that will be shown to hold of local phonological processes. Third, it briefly reviews the previous work on learning in phonology, to anticipate the demonstration in Chapter 6 of how

strict locality can be used as an inductive principle by a phonological learner. Lastly, it establishes the mathematical notations that will be used throughout the thesis.

Chapter 3 defines the class of Strictly Local functions, which in fact consists of two intersecting but non-equivalent classes called the Input Strictly Local functions and the Output Strictly Local functions. These functions will be established as a proper subset of the subsequential functions, which in turn are a proper subset of the regular relations. The restriction that identifies the SL functions as a proper subset is based on the SL formal languages. In addition, it is shown how a phonological rule of the form $x_i \rightarrow y_i \ / \ U \ \_\_ V$ can be compiled into a finite state machine that describes a Strictly Local function. In this rule, U and V refer to the sets of left and right contexts, respectively, for a phonological change, x_i is a member of the set of targets of the change, and y_i is its respective structural change. Once this characterization of Strictly Local functions is established, it will be shown to model the common phonological processes of local substitution, deletion, and insertion.

Chapter 4 extends the Strictly Local analysis to metathesis processes. It first motivates the copy-delete analysis of metathesis (Blevins and Garrett, 1998, 2004; Heinz, 2005a; Chandlee and Heinz, 2012), under which metathesis is actually the result of a copying process followed by deletion of the original. It then divides metathesis patterns into three categories - local metathesis, long-distance metathesis, and displacement - and identifies the conditions under which each is Strictly Local. In sum, the claim is that all synchronic metathesis is Strictly Local, while certain diachronic displacement cases are not.

Chapter 5 moves beyond phonology proper to discuss a number of word-formation processes that are also Strictly Local. In particular, general affixation is shown to be Strictly Local, as well as local partial reduplication (i.e., partial reduplication in which the reduplicant is affixed adjacent to the base material it was copied from). Non-local partial reduplication and full reduplication, which are not Strictly Local, are also discussed.

Chapter 6 presents a learning algorithm for the class of Input Strictly Local

functions, called the ISLFLA. It first discusses the algorithm known as OSTIA (Oncina et al., 1993), the intuitions of which serve as the foundation for the ISLFLA, as well as previous work on using OSTIA to learn phonological rules (Gildea and Jurafsky, 1995, 1996). It then describes in detail how the ISLFLA uses the property of strict locality to generalize an Input Strictly Local function from a finite set of positive data and proves that it indeed learns *any* function in this class in less time and with less data than OSTIA. Lastly, it provides four demonstrations of the algorithm based on attested phonological processes and discusses the insights provided by the learner in the larger context of phonological acquisition.

Chapter 7 discusses the contributions of the thesis and the key areas for future work. Specifically, it quantifies the empirical coverage of SL functions using the P-Base database of phonological patterns (Mielke, 2004) and then addresses the potential to expand this coverage in a variety of directions, including optional rules and free variation, processes that apply across word boundaries, non-local processes such as long-distance harmony and dissimilation, and suprasegmental phenomena like tonal patterns and stress assignment. Lastly, the current results will be situated into the bigger picture, both in terms of the relation between individual processes and the total grammar and the relation between computational properties and other explanatory factors like phonetic naturalness.

Chapter 8 concludes, followed by an appendix that provides a formal construction for Strictly Local finite state transducers that describe phonological rules as well as theorems for the properties of Strictly Local functions that have been proven thus far.

Chapter 2

BACKGROUND

This chapter first argues for the role of phonological processes in the phonological grammar. It then introduces the conception of a phonological process as a mapping or function as well as the utility of identifying formal properties of mappings and grammars. Since the analyses used throughout the thesis will rely on finite state automata, the conventions of this formalism will also be presented. Also discussed are the Strictly Local languages, a class of formal languages that provide the foundation for the strict locality property that will be shown to hold of phonological mappings. Lastly, a brief review of the previous work on learning in phonology is provided, followed by the mathematical preliminaries that will be used throughout the rest of the work.

2.1 Empirical Perspective

Despite the differences between rule- and constraint-based grammatical formalisms, both assume that the grammar maps an underlying form to a surface form. Consider, for example, the observable fact that the English plural morpheme has three distinct pronunciations, as shown in (1).

(1) a. bats, [bæt-s]
b. balls, [bɑl-z]
c. bases, [beɪs-ɨz]

Closer inspection of a larger set of examples would reveal that the occurrence of each pronunciation is not random, but is conditioned by the last segment of the word the suffix attaches to. Specifically, the [-s] variant follows voiceless segments, [-z] follows

voiced segments, and [-ɨz] follows sibilants (voiced and voiceless). The question is how to account for these variable yet systematic pronunciations within the grammar. One option is to posit three different plural morphemes, one for each pronunciation, the choice of which is conditioned by the preceding segment. But this analysis doesn't explain why the pattern looks like (1) as opposed to (2).

(2) a. bats, [bæt-z]
b. balls, [bɑl-ɨz]
c. bases, [beɪs-s]

In other words, it fails to capture the generalization that the morpheme variant is not just conditioned by the preceding segment, but either matches it in voicing ([-s] and [-z]) or avoids a sequence of two sibilants ([-ɨz]). It also fails to capture the generalization that a parallel set of facts can be observed with the past tense morpheme, as shown in (3).

(3) a. pitched, [pɪtʃ-t]
b. played, [pleɪ-d]
c. bunted, [bʌnt-ɨd]

Again we observe three distinct pronunciations, [-t], [-d], and [-ɨd], that either match the voicing of the preceding segment ([-t] and [-d]) or avoid a sequence of two alveolar stops ([-ɨd]). Furthermore, looking beyond English we find many languages that likewise favor or enforce word-final obstruent clusters with the same voicing feature. A possible analysis that reflects these larger generalizations is that these morphemes each have a single underlying form - /-z/ for the plural and /-d/ for the past tense - which is mapped to the correct surface form by the phonological grammar. Under a rule-based grammatical formalism, this mapping would involve the processes of epenthesis and assimilation in (4). Sample derivations are shown in Figure 2.1.

(4) a. Sibilant cluster epenthesis: $\emptyset \rightarrow$ ɨ / [+sibilant] __[+sibilant]

b. Alveolar stop cluster epenthesis:

$\emptyset \rightarrow$ ɨ / [-son,-cont,+ant,+cor] __[-son,-cont,+ant,+cor]

c. Voice assimilation: [+voice] $\rightarrow$ [-voice] / [-voice] __#

	/bæt+z/	/bɑl+z/	/beɪs+z/
Sibilant cluster epen.	—	—	[beɪsɨz]
Voice assimilation	[bæts]	[bɑlz]	[beɪsɨz]

	/pɪtʃ+d/	/pleɪ+d/	/bʌnt+d/
Alveolar stop cluster epen.	—	—	[bʌntɨd]
Voice assimilation	[pɪtʃt]	[pleɪd]	[bʌntɨd]

Figure 2.1: Rule-based derivations of plural and past tense allomorphy

Under a constraint-based grammatical formalism, the mapping would involve the constraint rankings in (5), as demonstrated in the tableaux in Figure 2.2. In (5) *SS and *DD are markedness constraints that are violated by sequences of two sibilants and two alveolar stops, respectively, and *TD# is a markedness constraint violated by a word-final sequence of a voiceless and voiced obstruent. DEP is a faithfulness constraint violated by a segment in the surface form without a correspondent in the underlying form, and IDENT(voice) is a faithfulness constraint violated by a segment whose voicing feature differs between the underlying and surface form.[1]

(5) {*SS, *DD} $\gg$ {*TD#, DEP} $\gg$ IDENT(voice)

Thus, the assumption that a phonological grammar maps an underlying form to a surface form says nothing about *how* the grammar matches a particular UR to its SR. Both rule- and constraint-based theories describe the same mapping. The next section

[1] This analysis omits the constraints that ensure the epenthetic vowel is [ɨ].

/bæt+z/	*SS	*DD	*TD#	DEP	IDENT(voice)
[bætɨz]				!*	
☞[bæts]					*
[bætz]			!*		

/bʌnt+d/	*SS	*DD	*TD#	DEP	IDENT(voice)
☞[bʌntɨd]				*	
[bʌntt]		!*			*
[bʌntd]		!*	*		

Figure 2.2: Fragment of a constraint-based grammar

will discuss how this commonality of the UR ↦ SR mapping serves as the foundation for the computational perspective taken in this thesis.

2.2 Computational Perspective

In the current work, a process such as voicing assimilation is taken to be an infinite function that maps any UR that contains a TD# sequence to a corresponding SR with a TT# sequence, as in (6).

(6) a. f(bæt+z) = bæts
b. f(bɑl+z) = bɑlz
c. f(pɪtʃ+d) = pɪtʃt
d. ...

In this way, the analyses that follow abstract away from the mechanism by which this mapping is represented in the grammar. The insights gained by analyzing the nature of these mappings will be shown to include 1) a more precise characterization of what constitutes a possible phonological process, and 2) an approach to how these mappings can be learned. Furthermore, these findings will be independent of and compatible with both rule- and constraint-based theories of phonological grammar. The next two

sections will discuss these contributions in turn.

2.2.1 Expressive power of grammars

A lot of what we know about the computational properties of language patterns comes from work identifying the expressive power of the grammars that generate them. One means of measuring this expressive power is to classify the pattern on the Chomsky hierarchy of formal languages[2] (Chomsky, 1956), shown in (7).

(7) finite $\subset$ regular $\subset$ context-free $\subset$ context-sensitive $\subset$ recursively enumerable

Grammars are classified on this hierarchy based on the languages they generate, meaning regular grammars generate regular languages, context-free grammars generate context -free languages, etc. Previous work in the computational analysis of natural language patterns revealed a difference between the expressive power needed to describe syntactic versus phonological patterns (see Heinz and Idsardi (2011)). For example, syntactic phenomena can be context-free (Chomsky, 1956, 1959) or context-sensitive (Schieber, 1985; Kobele, 2006). Interestingly, though SPE-style phonological analyses use context-sensitive rewrite rules of the form in (2-a), Johnson (1972) showed that the CAD $\mapsto$ CBD mapping is actually a regular relation (i.e., it generates a regular language) provided the rule does not apply to its own output within the same cycle. In other words, after one rule application derives CBD from CAD, the same rule cannot apply again to the B created by the first application. This restriction echoes the strict cyclicity condition that was proposed by Chomsky (1973) for syntax and adapted to phonology by Mascaró (1976).

The need for the condition on the rule's application is demonstrated with the example rule in (8).

---

[2] Though related, *mappings* and *languages* are two different entities. Languages are simply sets of strings (e.g., $\mathcal{L} = \{w_1, w_2, w_3, \ldots\}$), while mappings are sets of string pairs (e.g., $\mathcal{R} = \{(w_1, w_2), (w_3, w_4), \ldots\}$ where $\mathcal{R}(w_1) = w_2, \mathcal{R}(w_3) = w_4$,...).

(8) $\emptyset \rightarrow$ ab / __b

A single application of this rule to the input *ab* derives *aabb*. This output has two possible locations for an additional insertion: either *aa*__*bb* or *aab*__*b*. The first one is ruled out by the condition, since it falls within the output of the first application of the rule. Were that option allowed, the rule would in fact generate the context-free language $a^n b^n$. With the restriction on the rule's application, however, it instead generates the regular language $a(ab)^n b$. Johnson's result thus draws a distinction between what a grammar is capable of generating and what it actually needs to generate.

Kaplan and Kay (1994) expanded on this finding using the fact that the regular relations are closed under composition. If two regular relations f and g are ordered such that the output of f is used as the input of g, a single relation, written $g \circ f$, can be generated via functional composition. This composed relation, which is equivalent to $g(f(x))$, maps the input of f directly to the output of g.[3] The closure property means that if both f and g are regular, then $g \circ f$ will be regular as well. Thus a grammar of ordered rewrite rules, each of which (as Johnson showed) describes a regular relation, can be composed into a single relation that is also regular. This single relation will map the UR directly to the SR without proceeding through intermediate representations. The significance of this result is that the interactions among the individual generalizations, or rules, in the total grammar does not increase the grammar's power beyond regular. An alternative means of combining rules into the grammar is proposed by Koskenniemi (1983), whose rule system proceeds in parallel fashion rather than sequentially. In this *two-level* system, the input-output mapping of the entire phonology has to fall within the intersection of all the individual rule mappings in order to be valid. This approach has theoretical consequences as, like OT, it does away with rule ordering and intermediate representations, but it still does not increase the formal power of the system (Karttunen, 1993).

[3] Assuming g is defined for the image of that input under f.

The work of Johnson (1972), Kaplan and Kay (1994), and Koskenniemi (1983) narrowed the computational complexity bound of phonological processes to regular, but an even tighter computational bound is desirable for at least two reasons. First, the class of regular relations is too large, in the sense that many regular relations exist that are not likely to be attested in any natural language phonology. In other words, being regular is a necessary but not a sufficient condition for being a phonological mapping. Recall the example of an implausible phonological rule from the previous chapter, repeated below in (9-b).

(9) a. /i/ → [u] / d __

b. /i/ → [u] / x __(x is a string with an even number of obstruents)

Both of the rules in (9) describe regular relations, which means the computational property of being regular cannot tell us why (9-a) is plausible while (9-b) is not.

Second, the class of regular relations is not believed to be identifiable in the limit in the sense of Gold (1967)[4], so the property of regularity cannot be used to learn phonological mappings from positive data. These two concerns are in fact related; part of the reason why a learner cannot learn any regular relation is because the hypothesis space of regular relations is too large and unstructured for the learner to make the right distinctions among the data. A natural next step then is to narrow the bound on the hypothesis space. That is precisely what this thesis sets out to do, in proposing a *subregular* class of relations to model phonological processes. But first, since these relations will be characterized with finite state automata, the next section will introduce this formalism.

2.2.2 Finite state descriptions of language patterns

The regular languages mentioned above can be described with several converging characterizations. These include regular expressions, monadic second order logical

[4] See §2.4 for a definition of this learning framework.

formulae, and finite state automata, which will be introduced in this section. Only the basics of automata theory will be presented here; for more on automata and their linguistic applications, see Roche and Schabes (1997), Hopcroft et al. (2001), Beesley and Karttunen (2003), and Roark and Sproat (2007).

Consider the language $\mathcal{L}_{aa}$ that consists of strings of a's and b's that contain an even number of a's. This language contains an infinite number of strings, which makes it more convenient to represent it with the finite state acceptor (FSA) in Figure 2.3. This FSA can be thought of as a grammar for $\mathcal{L}_{aa}$, since for any given string of a's and b's the FSA can be used to determine whether or not that string is in $\mathcal{L}_{aa}$.

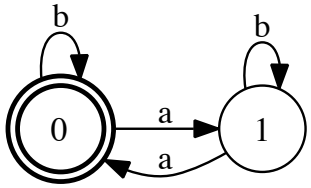


Figure 2.3: FSA for the language $\mathcal{L}_{aa}$

FSAs consist of a set of *states*, which are represented in Figure 2.3 with circles, and *transitions*, which are represented with labeled arrows that connect the states. Transitions that start and end in the same state are called *loops*. The state marked in bold is the *initial state*. Beginning in this state, the FSA reads a given input string one symbol at a time and takes the transition that matches the current input symbol. States marked with double circles are *final* or *accepting* states. If the last symbol of the input leads to a final state, the string is *accepted*, meaning it is in the language the FSA describes. If the last symbol leads to a *non-final* state, the string is not in the language.

As an example, consider two strings *abaaa* and *baaa*. We know the former is in $\mathcal{L}_{aa}$ but the latter is not. The series of transitions the FSA will take (called a *path*) when parsing *abaaa* is shown in (10).

(10)

	a		b		a		a		a	
0	→	1	→	1	→	0	→	1	→	0

When the FSA reaches the end of the string, it is in state 0. Since this is an accepting state, the string is correctly determined to be in the language. In contrast, the path for the input *baaa* is shown in (11).

(11)

	b		a		a		a	
0	→	0	→	1	→	0	→	1

This time the FSA ends in state 1, which is not an accepting state, so this string is not in the language. More generally, the FSA will only be in the accepting state 0 if it has seen an even number of a's, and it will only be in the non-accepting state 1 if it has seen an odd number of a's. Thus for *any* input, it will correctly determine whether or not it is in the language $\mathcal{L}_{aa}$.

This example clearly demonstrates that the states of a FSA represent the crucial pieces of information that are needed to distinguish those strings that are and are not in the language. For $\mathcal{L}_{aa}$, that crucial information is simply the parity of the number of a's the string contains. Thus two states suffice: one for even and one for odd. Languages will differ in what information the FSA needs to keep track of, but it is the fact that *some* FSA exists to describe a language that makes it a regular language.

Switching now from formal languages to relations, there is a corresponding finite state description of regular relations as well: finite state *transducers* (FSTs). These represent relations by again reading an input string but also producing a corresponding *output* string. FSTs can also be thought of as *acceptors* of *pairs* of strings. In other words, the pair (w_1, w_2) is accepted by a FST if the FST produces the output w_2 for the input w_1. The difference between FSAs and FSTs is reflected in the diagrams by labeling the transitions with pairs of symbols instead of single symbols, as in Figure 2.4. The symbol to the left of the colon corresponds to the input and the symbol to the right corresponds to the output.

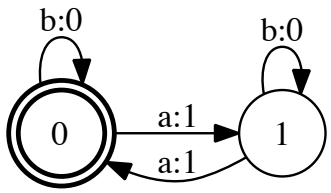


Figure 2.4: A finite state transducer

The FST in Figure 2.4 maps strings of a's and b's to strings of 1's and 0's, such that all a's are rewritten as 1's and all b's are rewritten as 0's. This mapping then includes $(aaaa, 1111)$, $(bbb, 000)$, $(aba, 101)$, $(bbaa, 0011)$, etc., and a sample path for input *aba* is shown in (12). The top line is the input, the bottom line is the output, and the middle line is the series of states, as before.

(12)

	a		b		a	
0	→	1	→	1	→	0
	1		0		1	

The relation represent by Figure 2.4 is only defined for input strings with an even number of a's (i.e., its domain is $\mathcal{L}_{aa}$). This means a string like *abaa* does not have a corresponding output, since the end of this input will be reached in the non-final state 1, just as in the FSA example above. Also note that the FST in Figure 2.4 is *deterministic*, which means that each state only has one outgoing transition for each possible input symbol. The consequence of determinism is that any given input string will only be mapped to a single output string. Thus the relation represented by Figure 2.4 is more accurately called a *function*. The alternative, non-determinism, would result in the FST mapping each input to a *set* of outputs.

The class of regular relations are those mappings that can be represented with a FST (either deterministic or non-deterministic). As discussed above in §2.2.1, the regular relations are sufficient for modeling phonological rules of the form A → B / C

__D (where A, B, C, and D are regular languages), but there is motivation for a restriction to a subclass of these relations. One viable subclass is the class of subsequential functions, which are describable with deterministic FSTs in which all states are final and each state has a *final output string* that is appended to the current output if the input string concludes in that state. These additional properties of subsequential FSTs (hereafter SFSTs, see Definition 9 in §2.4.2) restrict the kinds of mappings that they can describe, so that the subsequential functions are a proper subset of the regular relations (Mohri, 1997), as indicated in Figure 2.5.

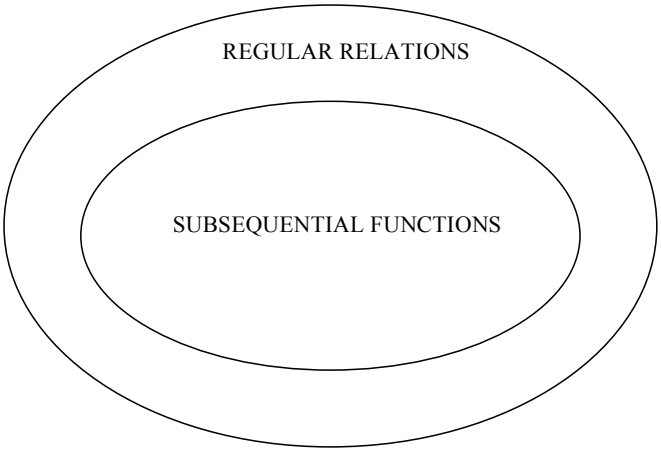


Figure 2.5: Relationship between regular relations and subsequential functions (to be revised)

As an example, consider the SFST in Figure 2.6. This automaton is similar to the one in Figure 2.4 above, except both states are final and the state labels include the final output string of the state (to the right of the comma). Note: λ represents the empty string. This SFST describes a function that maps input strings with an even number of a's to themselves, while input strings with an odd number of a's are mapped to themselves plus one additional 'a' appended to the end. To clarify how this works, the paths for inputs *abaa* and *baa* are shown in (13) and (14), respectively.

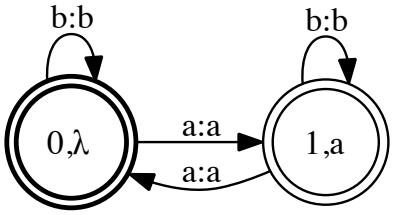


Figure 2.6: Subsequential finite state transducer

$$
\begin{array}{lccccccccccc}
 & & \text{a} & & \text{b} & & \text{a} & & \text{a} & & & \\
(13) & 0 & \rightarrow & 1 & \rightarrow & 1 & \rightarrow & 0 & \rightarrow & 1 & & \\
 & & \text{a} & & \text{b} & & \text{a} & & \text{a} & & \text{a} &
\end{array}
$$

$$
\begin{array}{lccccccccc}
 & & \text{b} & & \text{a} & & \text{a} & & \\
(14) & 0 & \rightarrow & 0 & \rightarrow & 1 & \rightarrow & 0 & \\
 & & \text{b} & & \text{a} & & \text{a} & & \lambda
\end{array}
$$

The mapping represented by the SFST in Figure 2.6 is more specifically a *left subsequential* mapping, because it can be described with a SFST that reads the input from left-to-right. A corresponding class of *right subsequential* functions exists which read the input from right-to-left. This means the class of subsequential functions is in fact two intersecting classes, as shown in Figure 2.7. As will be shown with phonological examples in the next chapter, certain mappings are only subsequential if the SFST can read the input from right-to-left.

Previous work has shown that many phonological processes can be modeled with (either left or right) subsequential functions, including vowel harmony (Gainor et al., 2012; Heinz and Lai, 2013), metathesis and partial reduplication (Chandlee et al., 2012; Chandlee and Heinz, 2012), consonant harmony (Luo, 2013) and dissimilation (Payne, 2013), and some (but not all) tonal patterns (Jardine, 2013).[5] Though these findings

[5] Based on this thesis we can add substitution, deletion, and insertion to this list. See the next chapter for details.

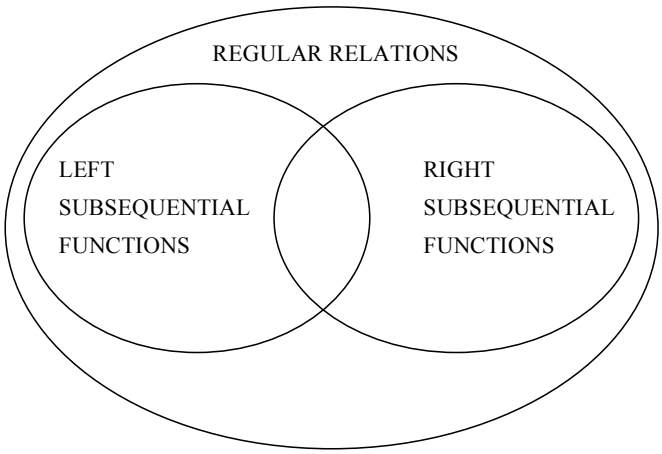


Figure 2.7: Left and Right subsequential functions

are significant in that they provide a tighter characterization of phonological mappings than that proposed by Johnson (1972), Koskenniemi (1983), and Kaplan and Kay (1994), this thesis will propose an even *tighter* characterization, which is motivated by the fact that not all subsequential functions are feasible phonological mappings.

The desired restriction will be achieved by combining the properties of subsequential transduction with the properties of a certain class of formal languages, called the Strictly Local languages, to define a class of functions called the Strictly Local functions. This means all Strictly Local functions are also subsequential functions, though the Strictly Local class will cross-cut the classes of left and right subsequential. Figure 2.8 depicts the relationship among the classes of regular relations, left and right subsequential functions, and the (to be defined) Strictly Local functions. This diagram will be revised in the next chapter once the SL functions are discussed in more detail. Recasting this hierarchy in terms of finite state automata, we have seen that the regular relations are describable with FSTs and the subsequential functions are describable with SFSTs. The SL functions will also be characterized in terms of automata, meaning a certain type of SFST will be defined that describes only Strictly Local functions. Since the properties that make these SFSTs Strictly Local are drawn from the formal

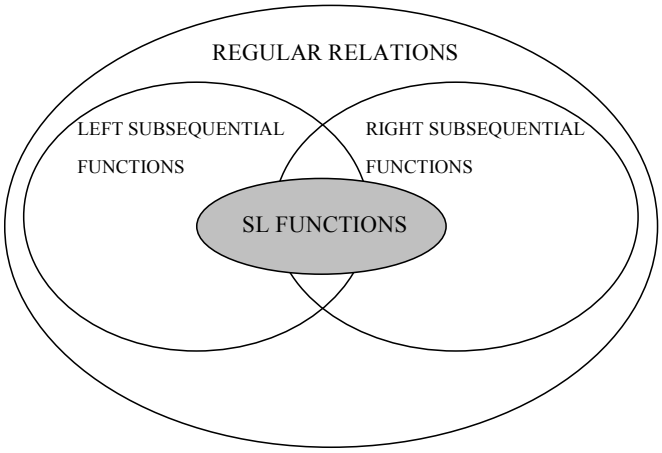


Figure 2.8: Relationship between subsequential and SL functions (to be revised)

Strictly Local languages, the next section will discuss those languages in detail.

2.2.3 Strictly Local languages and the Subregular Hierarchy

The regular languages were defined above as those languages that can be described with regular expressions, monadic second order logical formulae, or finite state acceptors. But there are also several well-studied classes of *subregular* languages that are described with more restrictive grammars. Collectively, these languages form a Subregular Hierarchy (McNaughton and Papert, 1971; Rogers and Pullum, 2011; Rogers et al., 2013), depicted in Figure 2.9. Like the Chomsky Hierarchy (in (7) above), the regions of the Subregular Hierarchy are distinguished by the formal complexity of the grammars that can describe them. Here only the most restricted class, the Strictly Local (hereafter SL) languages will be discussed, though a couple of other regions will be commented on in Chapter 7. For more details on the entire hierarchy, see McNaughton and Papert (1971), Rogers and Pullum (2011), and Rogers et al. (2013).

The SL languages are those that can be described with grammars that consist of a set of substrings of length k (called k-factors). A given string is a member of the

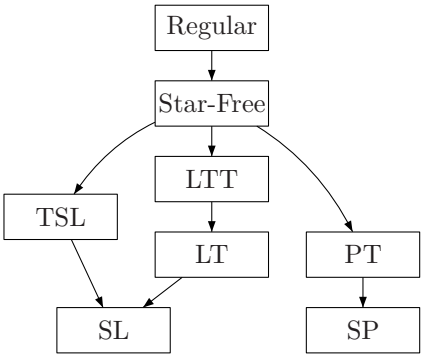


Figure 2.9: Subregular Hierarchy of formal languages (LTT = Locally Threshold Testable, TSL = Tier-based Strictly Local, LT = Locally Testable, PT = Piecewise Testable, SL = Strictly Local, SP = Strictly Piecewise)

language if and only if its own k-factors are a subset of the grammar.[6] For example, consider a language that includes all strings of a's and b's that do not end in 'a'. Using # to represent a word boundary, the restriction on the strings of this language can be expressed as follows: strings cannot contain the 2-factor $a\#$. The grammar for this language is then all possible 2-factors *except* $a\#$, or G = $\{\#a, \#b, aa, ab, ba, bb, b\#\}$. Since factors of length 2 are sufficient to describe this language, it is called a Strictly 2-Local, or SL-2, language. The example in (15) shows how this grammar G recognizes *aaab* as a string of this language, while *aaaa* is ruled out.

(15) a. 2-factors of *aaab*: $\{\#a, aa, ab, b\#\} \subseteq$ G ✓

b. 2-factors of *aaaa*: $\{\#a, aa, a\#\} \Rightarrow a\# \notin$ G ×

The 2-factors of *aaab* are all in G, so this string is in the language. But *aaaa* contains the 'illegal' 2-factor $a\#$, so it is not in the language. The Strictly Local designation refers to the fact that a string's well-formedness can be determined solely by examining these contiguous substrings of bounded length. This is sometimes explained as a scanner

[6] Alternatively, the grammar could be the set of k-factors that are *not* permitted, in which case a string is a member of the language if and only if its own k-factors are *not* in the grammar.

reading the string from left-to-right with a window that only reveals k symbols at a time. As soon as an illegal sequence of k symbols is found, the string is automatically ruled out, regardless of what came before or what may come after. Likewise, if the end of the string is reached without detecting any illegal sequences, the string is well-formed, regardless of which k-factors from the grammar it contains or the order in which they appear.

The SL languages also have an automata-theoretic characterization, meaning they are the languages that can be recognized by a certain type of FSA. A FSA for the example SL-2 language defined above (strings of a's and b's that do not end in 'a') is shown in Figure 2.10.

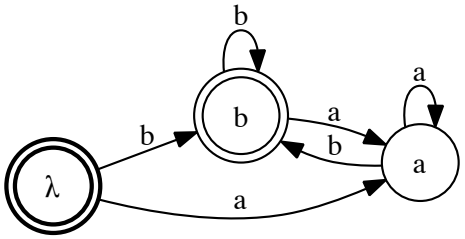


Figure 2.10: Canonical SL-2 finite state acceptor

This FSA is the *canonical* FSA for this SL-2 language, because the states correspond to factors up to length $k-1$ and the transitions are defined such that being in a state means those are the most recent $k-1$ symbols of the input. Thus, in Figure 2.10 the states are labeled with all possible substrings from the alphabet {a, b} up to length $k-1=2-1=1$. The FSA will only be in state a if it just read an 'a', and it will only be in state b if it just read a 'b'. Given such a structure, a SL FSA encodes the k-factors that are and are not permitted in the strings of the language it represents via the transitions that are and are not defined. For example, the fact that there is a transition from state b on the input 'a' in Figure 2.10 tells us that the k-factor ba is allowed in this language. And the restriction against strings ending in 'a' is achieved

by marking state a as a non-final state. Any input string that ends in this state will be (correctly) rejected from the language.

In addition to the k-factor grammar and automata-theoretic characterizations, the SL languages are also defined as those languages that have the property expressed in the following theorem.

Theorem 1 (Suffix Substitution Closure (Rogers and Pullum, 2011)). *$\mathcal{L}$ is Strictly Local if $\exists k \in \mathbb{N}$ such that given strings u_1, v_1, u_2, v_2, and x where the length of x is $k-1$, if u_1xv_1, u_2xv_2 are in $\mathcal{L}$, then it must be the case that u_1xv_2 is in $\mathcal{L}$.*

This property - called the Suffix Substitution Closure or the SSC - is just another way of encoding strict locality as the defining property of a SL language. It essentially says that if two strings that are known to be in the language overlap by a sequence of $k-1$ symbols, then the string created by swapping the portions that follow those $k-1$ symbols is also guaranteed to be in the language. This guarantee stems from the fact that if the substrings of these two strings (e.g., u_1, u_2, v_1, v_2, x) are already known to be acceptable, then nothing about the recombination of these substrings can disrupt well-formedness. In other words, there can be no 'long-distance' dependency that makes the combination of u_1 with v_2 ill-formed.

Thinking about the SSC in terms of the FSA description, consider two strings with different paths that at one point reach the same state. So the strings *bab* and *abb* both at one or more points take the FSA in Figure 2.10 to state b. After that point, whatever path the FSA takes for the rest of one string is also available when it is parsing the other string. Nothing about the path that led to the state in the first place determines where it can go next. Only the state itself determines what can come next.

These three characterizations - the k-factor grammars, the SL FSAs, and the SSC - all converge to define the exact same class of languages. Thus a language is SL if there is *some* k for which the language can be described in any of these three ways. In contrast, to prove a language is *not* SL is to prove there is *no* such k. The SSC is

the clearest way to make such an argument. Consider, for example, the language $\mathcal{L}_{aa}$ discussed above, in which all strings contain an even number of a's and any number of b's. Using the SSC, it can be shown that this language is not SL for any k. If it was SL for some k, then the SSC should hold. Consider some k that is an arbitrary even number. Then we know the strings $aa^{k-1}b$ and $ba^{k-1}a$ are both in $\mathcal{L}_{aa}$. Following the SSC, we can parse these strings into u_1, v_1, u_2, v_2, and x as in (16).

(16)

u_1	x	v_1	u_2	x	v_2
a	a^{k-1}	b	b	a^{k-1}	a

By the SSC, the string u_1xv_2 must also be in the language. But this string is $aa^{k-1}a$ or a^{k+1}, which is not in $\mathcal{L}_{aa}$ because it contains an odd number of a's (recall k is even). A parallel argument can be made assuming k is odd. Since in both cases k is arbitrary, we can conclude that there is no value of k for which the SSC will hold of this language. Therefore by Theorem 1, it is not a SL language. Other examples of non-SL languages include 1) strings of a's and b's that both start and end with 'b' (e.g., ✓*baab*, **aaab*), 2) strings of any number of a's and exactly three b's (e.g., ✓*aababb*, **aaabb*), and 3) strings of a's, b's, and c's such that a 'b' cannot follow an 'a' anywhere in the string (e.g., ✓*baacacc*, **baacacb*). In contrast, other examples of SL languages include 1) strings that do not begin with 'b' (SL-2), 2) strings in which a 'b' does not immediately precede an 'a' (SL-2), and 3) strings that do not have sequences of three b's in a row (SL-3).

Once we start thinking about alphabets of phonemes instead of abstract symbols like 'a' and 'b', these restrictions that characterize SL languages begin to resemble phonotactic restrictions. And indeed, Heinz (2007, 2009, 2010) has discussed how local phonotactic constraints can be modeled with SL languages. Consider again the example of English word-final obstruent clusters. The constraint on valid surface forms can be modeled with a SL-3 grammar that excludes the 3-factors for word-final obstruent clusters that disagree in voicing (e.g., *gs#*, *tz#*, *bs#*, etc.). Such a grammar will only generate and recognize the set of well-formed surface forms according to the phonotactic

restriction. But, as mentioned above, a phonological mapping cannot be modeled with a set of strings, and therefore the SL languages are not sufficient for modeling the process by which an underlying voiced obstruent surfaces as its voiceless counterpart. What is needed is a functional counterpart to these languages that will encode this notion of strict locality in a mapping. Defining this functional counterpart - the SL functions - is precisely the central contribution of this thesis.

This chapter thus far has presented the two key components of the definition of SL functions that will be proposed: 1) subsequential transduction and 2) SL languages. How these components combine to define SL functions is the topic of the next chapter. But first, since one of the motivations for modeling phonological processes with SL functions is that it leads to an approach to the learning problem, the next section will review some of the previous literature on learning in phonology.

2.3 Learning Phonology

A native speaker's complete knowledge of the sound patterns in his or her language includes at least three key areas: knowledge of contrast, knowledge of phonotactics, and knowledge of alternations or processes. The role of each of these areas in the acquisition of the others is a subject of ongoing research, and so attention is often focused on one area at a time. This thesis proposes an approach to the learning of phonological processes, and so this section will briefly review the literature on this area of learning in the two main frameworks of generative phonology, SPE and OT. For a selection of the research on phonotactic learning, see Hayes and Wilson (2008), Heinz (2007, 2009), and Heinz and Koirala (2010). And for more detailed discussion of the research that is outlined below, see Tesar (2003) and Heinz and Riggle (2011).

2.3.1 Learning rules

Comparatively little is known about the learning of phonological generalizations expressed as rules. Johnson (1984) presents a procedure for learning an ordered set of phonological rules of the form a $\rightarrow$ b / C, where C is a feature matrix of which 'a' is

a subset. Given a paradigm of underlying and surface forms parsed into component morphemes, this learner works backward by unapplying the (hypothesized) last rule to the surface form and positing the intermediate forms and rules that will lead to the underlying form. Unordered rules thus pose a problem for this procedure. Furthermore, it relies on an enumerative strategy for determining the last rule in the series, which offers little insight into the inductive principles at work in phonological learning.

Albright and Hayes (1999, 2002) propose an analogical learning model that, given inflectional data pairs (e.g., present $\mapsto$ past), identifies the structural changes and contexts of the alternations. Initially this procedure creates separate rules for each observed data point, but the learner gradually generalizes over the set of rules it has collected so far to create rules that account for as many forms as possible. Touretzky et al. (1990) propose a similar method in which the learner is given underlying-surface pairs. Lastly, Gildea and Jurafsky (1995, 1996) take an approach very similar to the one that will be presented in this thesis and demonstrate how phonological rules can be learned when encoded as subsequential finite state transducers. Much more will be said about their methodology and findings in Chapter 6, but their work highlights the utility of *insightful* learning biases in phonological learning.

2.3.2 Learning in OT

The shift in generative phonology toward constraint-based grammars initiated a large and ongoing body of work on how such grammars can be learned. In part, the greater attention that has been paid to learning within OT is due to the fact that the structure of OT grammars readily lends itself to learning algorithms that can identify the constraint ranking that is consistent with a given set of positive data. The inherent logic of an OT grammar is that a winning candidate (which can be identified as the correct surface form) must be preferred by at least one constraint that dominates all of the constraints that prefer any other candidate. This fact can be exploited by a learning algorithm that is equipped with the (by assumption) innate constraint set. The first algorithm to do this, called Recursive Constraint Demotion (Tesar and

Smolensky, 1993; Tesar, 1995), gradually identifies the constraint ranking by placing at the top those constraints that do not prefer any losers to winners. It recursively applies this logic, each time disregarding any winner-loser pairs that have already been accounted for as well as any constraints that have already been ranked, until it has either identified the ranking that is consistent with all of the data or has detected an inconsistency in the data. A later version of this algorithm, called Error-Driven Constraint Demotion (Tesar and Smolensky, 1996, 1998, 2000), constructs its own set of winner-loser pairs. Again taking the observed surface form to be the winner, the learner uses the given underlying form to generate a set of losing candidates that will provide the most information regarding the correct constraint ranking. In this sense the structure and logic of OT enables the learner to generate its own negative data from the observed (positive) data, which provides crucial information for inducing the grammar.

This central approach to learning within OT has since been adapted to and explored in the context of a variety of sub-problems, including learning the hidden structure of overt forms (Tesar, 1998a,b; Tesar and Smolensky, 1998; Jarosz, 2013), learning the underlying forms (Tesar et al., 2003; Alderete et al., 2005; Merchant and Tesar, 2008; Jarosz, 2006b), learning from surface forms alone (Riggle, 2004, 2006b), dealing with structural ambiguity (Tesar, 2002, 2004; Jarosz, 2009a), learning restrictive grammars (Jarosz, 2006a, 2009b), handling optionality, gradience, and noisy data (Boersma, 1997, 1998, 2000; Boersma and Hayes, 2001), handling exceptions (Pater, 2004), predicting stages of acquisition (Jarosz, 2007), the role of frequency in acquisition (Jarosz, 2010, 2011), learning phonotactics (Prince and Tesar, 1999, 2004; Hayes, 2004; Pater, 2004; Magri, 2010, 2011, 2012), and learning alternations from phonotactics (Tesar and Prince, 2003).[7]

The extensive progress that has been made in OT models of learning indeed adds to the appeal of OT as a theory of phonological grammar. And these learning

[7] See also Pulleyblank and Turkel (1995, 1996, 1998, 2000) for a genetic algorithm approach to learning in OT.

mechanisms are in fact compatible with *any* phenomenon (linguistic or otherwise) that can be described in OT. Tesar and Smolensky (1998), for example, state that their learning results 'are nevertheless independent of the details of any substantive analysis of particular phenomena. The results apply equally to phonology, syntax, and any other domain admitting an Optimality Theoretic analysis' (230). Such versatility of the learning mechanism discredits assumptions about language-particular cognitive mechanisms, and offers no explanation for the computational (Heinz and Idsardi, 2011) and experimental (Lai, 2012) evidence for different learning mechanisms for different linguistic domains (e.g., phonology versus syntax). Thus, following the arguments of Heinz (2007), the approach to learning advocated in this thesis will make use of a property that holds of the UR $\mapsto$ SR mapping itself, not of the particular grammatical formalism used to describe it. The idea is that the defining property of the mappings and the inductive principle of the learner are one and the same, because the structure of the mappings is also the structure of the learner's hypothesis space (see also Dresher (1999), who makes a similar argument in favor of the cue-based learning model of Dresher and Kaye (1990)). This connection explains *why* the phonological patterns are restricted in the way that they are, since the learner can only learn the patterns that have this property. Patterns without the property that happen to be consistent with the data will not find their way into the grammar.

The next and final section of this chapter will present the mathematical notations that will be used throughout the rest of the thesis.

2.4 Mathematical Preliminaries

2.4.1 Strings, languages, relations, and functions

An *alphabet*, Σ, is a finite set of symbols. A *string* or *word*, w, is a finite sequence of symbols from Σ. The set of all possible strings using symbols in Σ is represented as Σ^*, and this set includes the unique string of zero symbols, called the empty string and represented with λ. The length of a string is the number of symbols it contains

and is designated $|w|$ (so $|\lambda| = 0$). The set of all possible strings of length n and of length up to and including n is Σ^n and $\Sigma^{\leq n}$, respectively.

If $w = ps$ for some $p, s \in \Sigma^*$, then p is a *prefix* of w and s is a *suffix* of w. The set of prefixes of w, $\texttt{Pref}(w) = \{p \in \Sigma^* \mid \exists s \in \Sigma^* \; [w = ps]\}$, and the set of suffixes of $w, \texttt{Suff}(w) = \{s \in \Sigma^* \mid \exists p \in \Sigma^* \; [w = ps]\}$. The unique suffix of w of length n is $\texttt{Suff}^n(w)$. If $|w| < n, \texttt{Suff}^n(w) = w$. If $w = ps$, then $ws^{-1} = p$ and $p^{-1}w = s$. The longest string in a set S, $\texttt{max}(S)$, is $s \in S$ such that $\forall s' \in S, |s'| < |s|$. The longest common prefix of a set of strings S, $\texttt{lcp}(S)$, is then $\texttt{max}(\cap_{w \in S}\texttt{Pref}(w))$.

A *language* $\mathcal{L}$ is a subset of Σ^* ($\mathcal{L} \subseteq \Sigma^*$). A relation $\mathcal{R}$ is a subset of the cross-product of the two languages: $\mathcal{R} \subseteq \mathcal{L}_1 \times \mathcal{L}_2 = \{(w_1, w_2) \mid w_1 \in \mathcal{L}_1 \text{ and } w_2 \in \mathcal{L}_2\}$. The *left projection* of a relation $\mathcal{R}$ is $\psi_1 = \{w_1 \mid \exists w_2 \; [(w_1, w_2) \in \mathcal{R}]\}$, and the *right projection* of $\mathcal{R}$ is $\psi_2 = \{w_2 \mid \exists w_1 \; [(w_1, w_2) \in \mathcal{R}]\}$. A relation f is a *function* if $\forall w_1 \in \psi_1$ there is exactly one $w_2 \in \psi_2$. For a function f and a string $x \in \Sigma^*$, the set of *tails* of x with respect to f is $\texttt{tails}_f(x) = \{(y, v) \mid f(xy) = uv \wedge u = \texttt{lcp}(f(x\Sigma^*))\}$. This set is empty if $x \notin \cap_{w \in \psi_1}\texttt{Pref}(w)$.

2.4.2 Finite state automata

Definition 1 (DFSA). *A* deterministic finite state acceptor (DFSA) *is a five-tuple* $\tau = \{Q, q_0, F, \Sigma, \delta\}$, *where* Q *is a finite set of states,* $q_0 \in Q$ *is the initial state,* $F \subseteq Q$ *is the set of final states,* Σ *is the alphabet, and* $\delta : (Q \times \Sigma) \mapsto Q$ *is the transition function (i.e.,* $\delta(q, a) = q'$ *if there is a transition from* q *to* q' *on input* a*).*

The transition function of a DFSA can be extended to strings as follows (adapted from Hopcroft et al. (2001)). $\forall q \in Q$,

$$\hat{\delta}(q, w) = \begin{cases} q & if\ w = \lambda \\ \delta(\hat{\delta}(q, w'), a) & if\ w = w'a\ for\ some\ a \in \Sigma \end{cases}$$

The language recognized by τ is $\mathcal{L}(\tau) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$.

Definition 2 (FST). *(Oncina et al., 1993) A* finite state transducer (FST) *is a six-tuple* $\tau = \{Q, \Sigma, \Gamma, q_0, F, E\}$, *where* Q *is a set of states,* Σ *and* Γ *are the input and*

output alphabets, respectively, q_0 *is the initial state,* $F \subseteq Q$ *are final states, and* $E \subseteq (Q \times \Sigma \times \Gamma^* \times Q)$ *is a set of* edges.

The set of edges E of a FST is associated with a transition function δ: $\forall(q, a, o, q') \in E$, $\delta(q, a) = (o, q')$. Two auxiliary functions that specify only the output string and destination state of a given transition will also be useful. They are defined as follows: $\forall(q, a, o, q') \in E$, $\delta_1(q, a) = o$, and $\delta_2(q, a) = q'$.

Definition 3 (SFST)**.** *A* subsequential finite state transducer (SFST) *is a six-tuple* $\tau = \{Q, \Sigma, \Gamma, q_0, E, \sigma\}$*, where* Q *is again a finite set of states, all of which are final, and* q_0, Σ, Γ*, and* E *are defined as for FSTs. The* final output function $\sigma : Q \mapsto \Gamma^*$ *maps each state to a (possibly empty) string that is appended to the output of any input string that ends in that state.*

SFSTs are deterministic on the input, meaning $(q, a, u, r), (q, a, v, s) \in E \Rightarrow (u = v \wedge r = s)$. A *path* in a SFST is a sequence of edges in E, $\pi = (q_0, a_1, o_1, q_1)...$ (q_{n-1}, a_n, o_n, q_n), $q_i \in Q, a_i \in \Sigma, o_i \in \Gamma^*, 1 \leq i \leq n$, which can also be condensed to $(q_0, a_1...a_n, o_1...o_n, q_n)$. Let Π_τ be the set of all possible paths in τ. The function f realized by τ is then $f(w_1) = w_2\sigma(q)$ such that $(q_0, w_1, w_2, q) \in \Pi_\tau$ (Oncina et al., 1993).

2.4.3 Learning framework

The learning results presented in Chapter 6 follow the Gold paradigm of identification in the limit (Gold, 1967). This section presents the crucial definitions for this framework.

A *text* t is an infinite sequence (or set) of items from the target language or mapping. The *nth* element of this sequence is denoted by $t(n)$, and by $t[n]$ is meant the finite portion of the sequence up to the *nth* item, or $t(0), t(1), ..., t(n)$. A *learner* ϕ: $t[n] \mapsto G$ maps finite sequences of texts to grammars. These grammars can be thought of as hypotheses for the grammar that generates the target language or mapping. If the target is a language, these hypothesis grammars might, for example, be in the form

of finite state acceptors. If the target is a mapping, they might be in the form of a finite state transducer.

There are two criteria for a learner to identify a language or mapping in the limit. The first is *convergence*. A learner converges on a text t iff there exists $n \in \mathbb{N}$ such that for all $m > n$, $\phi(t[m]) = \phi(t[n])$. In other words, there is a point at which, no matter what additional data the learner observes, its hypothesis grammar will never change. The second criterion is *correctness*, which is simply that this grammar the learner has converged to generates exactly the target language or mapping. And by extension, a learner identifies a *class* of languages or mappings in the limit if for *any* language or mapping in that class, the learner identifies it in the limit.

While some dispute the relevance of the Gold framework to natural language learning (Clark and Lappin, 2012), others defend it Heinz (to appear). The Gold framework is a relatively difficult paradigm in which to prove learnability, on account of the requirement that the learner identify *exactly* the correct target and that it do so using only positive examples. In other paradigms, the requirement of exact identification is relaxed (see for example, Valiant (1984) and Kearns and Vazirani (1994)) or the use of negative data is allowed (i.e., the text includes examples that the learner is explicitly told are *not* in the language). Some classes of languages that are not Gold-learnable are learnable in a different paradigm (e.g., the regular languages, see de la Higuera (2010)). But the Gold paradigm is appropriate in the context of natural language learning, since child language learners neither use nor consistently receive examples of negative data.

Chapter 3

MODELING PHONOLOGICAL PROCESSES WITH STRICTLY LOCAL FUNCTIONS

This chapter defines Strictly Local functions and shows how they can model local phonological processes such as substitution, deletion, and insertion. Strictly Local functions are the functional counterpart of the SL languages discussed in the previous chapter. And like those languages, the definition of SL functions will involve k-factors, or substrings up to a certain length k. Recall that SL languages can be defined with a grammar that consists of the k-factors that are allowed in strings of the language. They can thus model the set of valid surface forms in a natural language (i.e., phonotactics). To model phonological processes, the SL functions will map (or 'rewrite') illegal k-factors in the input to the corresponding 'repaired' string as dictated by the process (i.e., a string with segments substituted for others, or deleted, or inserted, etc.).

Defining SL functions in terms of a grammar, however, is not as straightforward as in the case of SL languages. Consider the simple rule in (1), which will be used throughout this chapter for demonstration purposes.

(1) a → b / a __a

This process substitutes a 'b' for an 'a' that appears between two other a's. It follows that the language that includes this process disallows the sequence $*aaa$ in surface strings, a phonotactic restriction that could be modeled with a SL-3 grammar that omits this 3-factor. But what grammar models the process that repairs these sequences? One might consider a grammar of tuples in which the first member is a k-factor and the second is the string that k-factor is mapped to (e.g., $\{(aaa, aba), (baa, baa), (bbb, bbb), ...\}$). But the k-factors of a word overlap, such that the word itself is not

simply the concatenation of its k-factors. For example, the word $w = abcda$ contains the 2-factors $\#a, ab, bc, cd, da$, and $a\#$, but the concatenation of these 2-factors ($aabbccddaa$) does not equal w. This overlapping nature of k-factors raises the question of how to correctly piece together the output string for a given input using a tuple-grammar like the one above.

Therefore, this chapter will instead present an automata-theoretic characterization of SL functions, which handles this 'overlapping problem' and produces the correct output string for a given input. The SL functions will in fact be defined with two distinct subclasses: the Input Strictly Local (ISL) functions and the Output Strictly Local (OSL) functions. These two classes are formally distinct, but there is also a linguistic motivation for separating them. Consider again the example rule in (1). Table 3.1 shows that this rule in fact corresponds to two distinct mappings depending on whether it is applied to the input $aaaaa$ in a simultaneous manner or left-to-right (Kaplan and Kay, 1994; Hulden, 2009).

Table 3.1: Mapping by Mode of Application

Simultaneous	Left-to-Right
$aaaaa \mapsto abbba$	$aaaaa \mapsto ababa$

There is another logically possible mode of application: right-to-left. But a corresponding third definition for the function that can model this mapping is not necessary, for reasons that will be discussed below. The difference between simultaneous and left-to-right application lies in whether the left context of the rule is verified using the most recent *input* (= simultaneous) or the most recent *output* (= left-to-right). This becomes intuitive when one considers the difference in the two application modes. For simultaneous application all possible targets of the rule are located before any change has been made (i.e., only the input string matters), while in left-to-right application

one change can feed or bleed a following application (i.e., the output determines what happens next).

This difference carries over to the SL function definitions: simultaneous application is modeled with *Input* Strictly Local functions and left-to-right application is modeled with *Output* Strictly Local functions. The definitions of these two classes of functions, as well as a procedure for compiling phonological rules into their respective transducers, will be presented in §§3.1 and 3.2. But first, it is worth reiterating that although the example processes used throughout this and the next two chapters will be first expressed with rules, the analysis is always meant to show that it is the *mapping* that is Strictly Local, not the rule itself. Furthermore, though the FST constructions will be applied to rules, the rules are just an intermediate step in the procedure of modeling a phonological mapping with a SL transducer. Developing a SL transducer construction from an OT grammar is beyond the scope of this thesis, but the wealth of previous work on FST representations of OT (see Frank and Satta (1998), Karttunen (1998), Eisner (2000), Gerdemann and van Noord (2000), and Riggle (2004), among others) suggests that such a construction will be a feasible area of future research.

3.1 Input Strictly Local Functions

This section will define Input Strictly Local functions, which will be shown to model the simultaneous application of a phonological rule. Before presenting the definition itself, it is worth elaborating on the definition of *tails* from the previous chapter, as this concept forms a crucial part of the definition. Formally, the tails of a string x in the prefixes of the domain of a function is the set $\texttt{tails}_f(x) = \{(y, v) \mid f(xy) = uv \wedge u = \texttt{lcp}(f(x\Sigma^*))\}$. Informally, the tails are the contributions to the output string of all possible extensions of x.

As an example, consider a function for post-nasal obstruent voicing, a process (attested in Coatlán Mixe (Wichmann, 1995)) that can be represented with the rule in (2).

(2) [-son] → [+voice] / [+nasal] __

For demonstration purposes, assume an alphabet of Σ = {D, T, V, N}, where D is a voiced obstruent, T is its voiceless counterpart, V is a vowel, and N is a nasal. Now consider the example string VN. The function would map this string to the output VN. In fact, *any* input string that begins with VN will be mapped to an output string that begins with VN. This means that every string in the infinite set of strings from Σ^* that have VN as a prefix[1] will be mapped to an output string with VN as a prefix; thus VN is called the *longest common prefix* of all of these output strings.

The remainder of the output string will differ depending on the next input symbol. Extending the input with T, V, N, or # will extend the output by T, V, N, and λ, respectively. But extending it with D will extend the output by T. These pairings of 'input extension' and 'output extension' are members of the set of tails (with respect to f) for VN, as shown in (3).

(3) $\texttt{tails}_f$(VN) = {(T, T), (V, V), (N, N), (D, T), (#, λ)...}

As an infinite set, $\texttt{tails}_f$(VN) includes all possible extensions of VN (of any length). Only the shortest of these (the single symbol ones) are represented in (3), but that is sufficient to observe that the longest common prefix of the collective output extensions is λ. In this way, the set of tails isolate the contributions to the output of each possible input extension. Note that these output contributions do not have to be unique - both T and D extend the output of VN by T. What is important is that there is no prefix common to the *entire* set of outputs.

The other crucial aspect of the definition of ISL functions is that the tails of VN are also the tails of DVN, TVN, NN, and, in fact, any string that ends with N. What all such strings have in common is a suffix of length $k - 1$ (since in this example k =

[1] Here the term *prefix* has no morphological import, but is simply some portion of a string starting from the beginning. See §2.4 for a formal definition.

2).[2] The fact that strings with the same $k-1$ length suffix have the same set of tails is the defining property of an ISL function (i.e., they are the functions for which this property is true).

Definition 4 (Input Strictly Local Function)**.** *A function f is Input Strictly Local iff $\exists k \in \mathbb{N}$ such that for all $u_1, u_2 \in \Sigma^*$, it is the case that if $\texttt{Suff}^{k-1}(u_1) = \texttt{Suff}^{k-1}(u_2)$ then $\texttt{tails}_f(u_1) = \texttt{tails}_f(u_2)$.*

As discussed in the previous chapter, an automata-theoretic characterization of these functions will be employed in the analyses that follow. The following theorem defines the FSTs that describe ISL functions - a proof that these FSTs correspond exactly to the class of ISL functions is in the appendix.

Theorem 2. *A function f is Input Strictly Local iff $\exists k \in \mathbb{N}$ such that f can be described with a SFST[3] $\mathcal{T}_{ISL}(f)$ for which*

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

The first part of the theorem, which echoes the definition of the FSAs that accept SL languages, says that the states of the SFSTs that describe Input SL functions correspond to the set of possible $k-1$-factors. The second part defines the transitions between states in such a way that it is only possible to be in a state if those $k-1$ symbols correspond to the most recent *input*.

To prove that a given phonological process is SL, it is sufficient to construct a SL FST that describes it. Before the construction procedure is presented, a quick note on FST alphabets. The definition of SFSTs in §2.4 of Chapter 2 includes distinct input and output alphabets, Σ and Γ, respectively. Since both ISL and OSL FSTs are by definition SFSTs, this distinction will be maintained. And in fact, from a phonological standpoint it makes sense to have two alphabets, under the assumption that URs

[2] Again, here the term *suffix* has no morphological meaning, but simply refers to a portion of the string starting from the end. See §2.4 for a formal definition.

[3] See Definition 3.

contain only phonemes and SRs contain only allophones. Also by assumption, the input alphabet is a subset of the output alphabet.

Given a rule of the form in (4) that applies simultaneously, where x, y, u, and v are strings, $\mathcal{T}_{ISL}$ can be constructed as follows.[4]

(4) $x \rightarrow y \ / \ u \_\_ v$ (simultaneous)

For $k = |uxv|$, the state set $\mathrm{Q} = \Sigma^{\leq k-1}$ and the initial state $q_0 = \lambda$. Transitions are defined for each state and input symbol. By Theorem 2, the destination state of these transitions is always the state that corresponds to the suffix of length $k-1$ of the sequence q plus a. But some care must be taken to determine the outputs of these transitions.

Consider some string $s \in \Sigma^*$, which is of length $n = |s|$. This string can also be written as $s_1 s_2 ... s_n$, where $s_i \in \Sigma$ for $1 \leq i \leq n$. Likewise, the strings u, x, v from (4) can be written $u_1 ... u_m$, $x_1 ... x_l$, and $v_1 ... v_r$, where $m = |u|$, $l = |x|$, and $r = |v|$. Now consider the string of input symbols in (5).

(5)

s_1	...	s_n	u_1	...	u_m	x_1	...	x_l	v_1	...	v_{r-1}	v_r

Since $k = m + l + r$, we know there is a state in Q that corresponds to $u_1 ... v_{r-1}$. In some sense this is the state the FST is 'looking for', since once it is in this state it has seen the entire context for the process and must enact the change of $x \rightarrow y$. Before it reaches that state, however, it must proceed through a series of states that each contain some portion of uxv. This is because Q contains states for all possible sequences up to length $k-1$, which means there are states for $s_n ... v_{r-2}$, $s_{n-1} ... v_{r-3}$, $s_{n-2} ... v_{r-4}$, and so on. These states all have some portion of uxv as a suffix.

Once the FST is in a state that has all of u as a suffix, as long as it continues to read in symbols from xv it must withhold the output (i.e., it must output λ). This is because x may or may not have to be output as y - that decision cannot be made

[4] See the appendix for a formal construction.

until all of xv has been read in or a symbol not in xv is read, in which case the FST 'resets' and begins looking for u again.

Therefore, when determining the output of a transition, there are four conditions to consider:

1. q has u plus some proper prefix of xv as a suffix, and a extends further into xv:

	q							a				
s_1	...	s_n	u_1	...	u_m	x_1	...	x_l	v_1	...	v_{r-1}	v_r

   The output of this transition is λ, and the destination state is called a 'holding' state (because the output is being withheld).

2. q has u plus some proper prefix of xv as a suffix, but a moves outside of xv. Since q is a holding state, the output must include whatever was being held (i.e., whatever portion of xv was already seen). However, if the destination state is also a holding state and the string being held in that state overlaps the output to some degree, this 'overlap' must not be included in the output.[5] The overlapping portion can be determined as the longest prefix of the string being held in the destination state that is also a suffix of the output:

					overlap			
output string:	x_1	x_2	...	v_1	v_2	a		
string held in destination state:					x_1	x_2	x_3	x_4

3. q is all but the last symbol of uxv, and a is that last symbol.

			q									a
s_1	...	s_n	u_1	...	u_m	x_1	...	x_l	v_1	...	v_{r-1}	v_r	

   The output is yv, minus whatever portion (if any) of it is being held in the destination state.

4. If none of the other conditions apply, the output is the same as the input.

Lastly, the final output function σ simply maps each state to the (possibly empty) string it is holding. The above procedure produces the $\mathcal{T}_{ISL}$ in Figure 3.1 for the rule in (1), which describes the correct mapping for simultaneous application.

[5] This situation only arises when u, x and/or v are not distinct, as in (1).

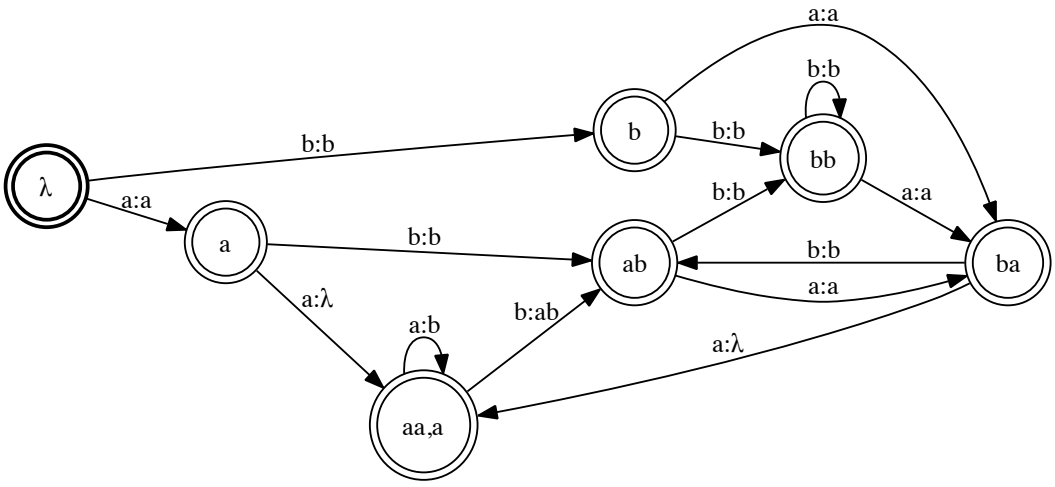


Figure 3.1: $\mathcal{T}_{ISL}$ for the simultaneous application of (1), $k = 3$, $\Sigma = \{\text{a, b}\}$

Transitions $\delta(a, a)$ and $\delta(ba, a)$ fall under condition 1 above, transition $\delta(aa, b)$ falls under condition 2, transition $\delta(aa, a)$ falls under condition 3, and all other transitions fall under condition 4. Note that σ has assigned the string a to state aa. This is the a that will become a b if another a is read in. If the string instead ends in state aa, the σ function 'returns' that a unaltered by appending it to the current output.

3.2 Output Strictly Local Functions

As stated above, the difference between ISL and OSL is that in the latter the transduction pays attention to the most recent output. This leads to the following definition for Output Strictly Local functions, which can model the left-to-right application of a rule.

Definition 5 (Output Strictly Local Functions)**.** *A function f is Output Strictly Local iff $\exists k \in \mathbb{N}$ such that for all $u_1, u_2 \in \Sigma^*$, it is the case that if $\texttt{Suff}^{k-1}(f(u_1)) = \texttt{Suff}^{k-1}(f(u_2))$ then $\texttt{tails}_f(u_1) = \texttt{tails}_f(u_2)$.*

As with ISL functions, a corresponding FST characterization of OSL functions will be used in the analyses that follow. This FST is defined in the following theorem (see the appendix for a proof that these FSTs describe OSL functions).

Theorem 3. *A function f is Output Strictly Local iff $\exists k \in \mathbb{N}$ such that f can be described with a SFST $\mathcal{T}_{OSL}(f)$ for which*

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

The procedure for compiling a rule of the form in (6) into an OSL transducer is as follows.

(6) $x \rightarrow y \ / \ u \_\_ v$ (left-to-right)

For $k = |uxv|$, the state set $\mathrm{Q} = \Gamma^{\leq k-1}$ and the initial state $q_0 = \lambda$. For the transitions from each state q on each input symbol, the destination state always corresponds to the suffix of length $k-1$ of q plus the output of the transition. The outputs of the transitions and the final output funciton can be determined using the same procedure as for $\mathcal{T}_{ISL}$ above. Figure 3.2 shows $\mathcal{T}_{OSL}$ for the rule in (1) and describes the correct mapping for left-to-right application.

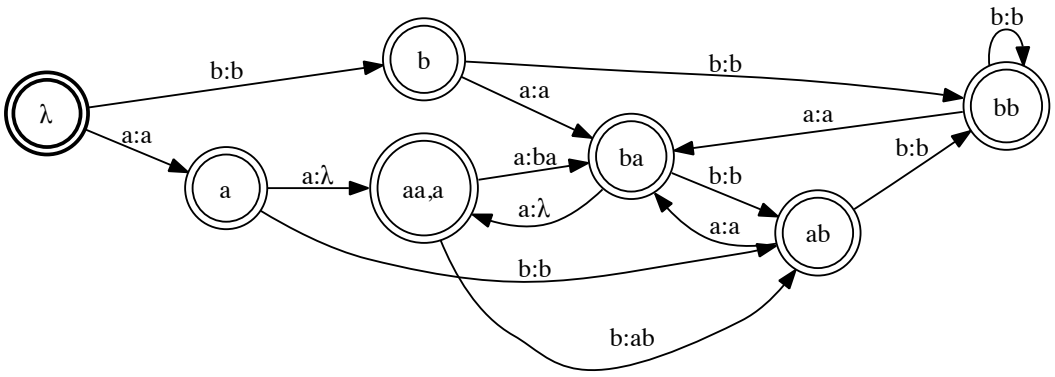


Figure 3.2: $\mathcal{T}_{OSL}$ for the left-to-right application of (1), $k = 3$, $\Sigma = \{\mathrm{a}, \mathrm{b}\}$

What about right-to-left application? Indeed we want to model this mode of application as well, to accommodate cases of regressive harmony like the example from Kolokuma Ijo shown in (7).

(7) Kolokuma Ijo (Williamson, 1965)

a. /wai͡/ ↦ [w̃ai͡] 'breath'

b. /sanlo/ ↦ [sãnlo] 'gills'

c. [+son] → [+nasal] / __[+nasal]

Recall from the previous chapter that the subsequential functions are divided into two subclasses, left and right subsequential, that differ in terms of the direction in which the input string is read (left-to-right or right-to-left, respectively). Since the SL functions are a proper subset of subsequential, it follows that these two options for reading the input string should be available for SL functions as well. The example in (7-a) shows us that the rule is applied iteratively, which means an ISL function will not be sufficient to model this mapping. Thus, what we need is an OSL function that reads the input string from right-to-left.

We can simulate this right-to-left reading of the input by first reversing the input and then reversing the output that the OSL function maps this reversed input to. This procedure allows us to essentially treat regressive rules as progressive (i.e., left-to-right). Consider the progressive nasal harmony rule in (8), which is the mirror version of the rule in (7-c).

(8) [+son] → [+nasal] / [+nasal] __

Assuming an alphabet of $\Sigma = \{N, D, V\}$, where N is a nasalized segment, D is an obstruent, and V is a sonorant, $\mathcal{T}_{OSL}$ for this process is shown in Figure 3.3.

Now consider the input VVND. If we reverse this string, we get DNVV, which is mapped to DNNN by the FST in Figure 3.3. If we then reverse this output, we get NNND. Thus we have modeled the mapping of VVND ↦ NNND, which corresponds

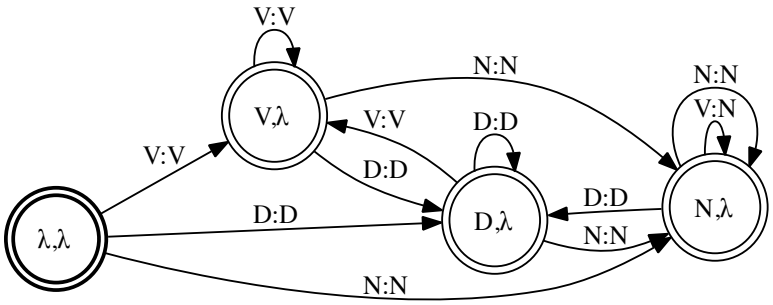


Figure 3.3: $\mathcal{T}_{OSL}$ for (8), $k = 2$, $\Sigma = \{\text{N, D, V}\}$

to regressive nasal harmony.[6]

So far we have seen how to model the simultaneous, left-to-right, and right-to-left application of a rule of the form in (4), repeated in (9),

(9) $x \rightarrow y \ / \ u\_\_v$

where x, y, u, v are strings. Many phonological processes can be described with rules of this form, but the empirical coverage is limited in that the rule format assumes a single target, x, and a single triggering context of $u\_\_v$. In practice, however, most processes involve multiple targets and contexts, because they apply to natural classes rather than single strings. In addition, it is unclear how the schema in (9) will accommodate rules triggered by a word boundary. If the word boundary # is simply included in Σ, the formalism over-generates in that # can appear anywhere in the strings x, y, u, and v. For example, it would allow for rules such as in (10).

(10) a. $\#\rightarrow a \ / \ a \ \_\_a$

b. $a \rightarrow \# \ / \ ab \ \_\_a$

c. $a \rightarrow b \ / \ a \ \_\_\#\#\#$

[6] See Heinz and Lai (2013) for more details on using reversed strings to achieve right-to-left mappings.

Rules such as these are not plausible from a phonological standpoint.[7] The rule schema therefore needs to be modified in such a way that it can handle word-boundary processes without allowing for rules like those in (10). The next two sections will adapt this rule schema to accommodate word boundary processes and multiple targets and triggers, respectively.

3.3 Word Boundary Processes

To disallow rules of the sort in (10), the word boundary symbol # must be restricted to being either the first symbol of u or the last symbol of v. So given (4) we specify that $x,\ y \in \Sigma^*$, $u \in \{\#, \lambda\} \cdot \Sigma^*$ and $v \in \Sigma^* \cdot \{\#, \lambda\}$. This means # is not included in the alphabet Σ. It does, however, contribute to the value of k, since it is part of the structural environment uxv.

Word-final processes can now be handled with the final output function. So far that function has served to return any held material in the event a transduction concludes in a holding state. Now its use will be extended to perform the rewriting of the k-factor in the event the rule is triggered by the word-final boundary. In that case (and only in that case), instead of rewriting uxv on a transition, as before, the final output function assigns yv to the state that corresponds to all but the last symbol of uxv (the last symbol being #). As a result, if the string ends, the trigger of the process has been found, and the change yv is appended to the current output.

Modifying the example from above, consider the rule in (11), which includes the word-final boundary in the triggering context.

(11) $a \rightarrow b \ / \ a \_\#$

The transducer for this rule is shown in Figure 3.4. Note that in this case both the ISL and the OSL FST construction produce the same FST. This is because the simultaneous and left-to-right application of word-boundary processes do not differ (i.e., there can

[7] Though see Chapter 7 for discussion of processes that take place across word boundaries.

only be a single application of the rule).[8] More generally, we only observe different mappings for the different modes of application when there is some overlap between the target of the rule and the triggering context. Thus for this and all future examples, $\mathcal{T}_{SL}$ will be used if $\mathcal{T}_{ISL} = \mathcal{T}_{OSL}$.

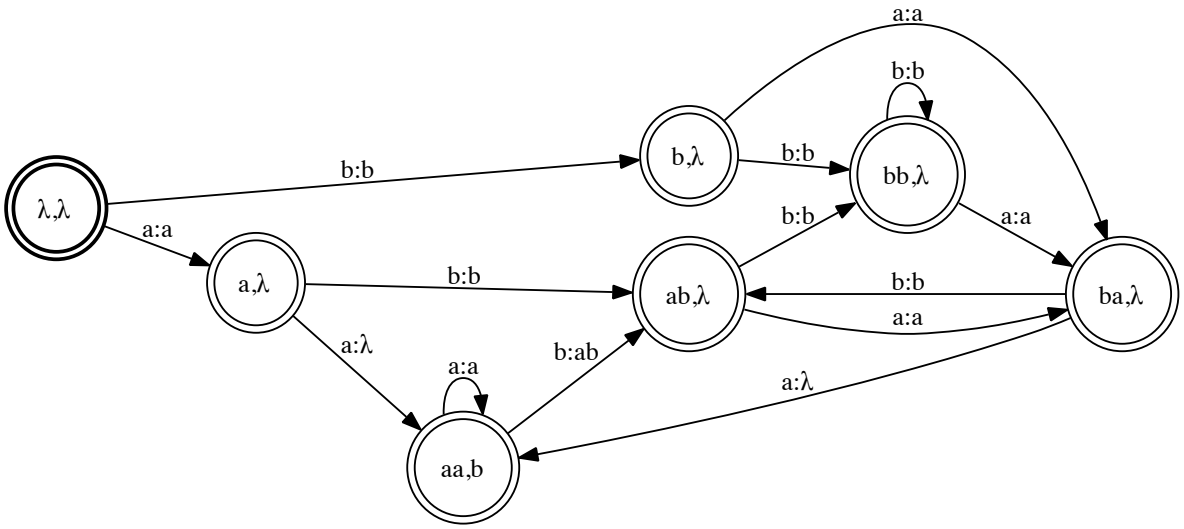


Figure 3.4: $\mathcal{T}_{SL}$ for (11), $k = 3$, $\Sigma = \{\text{a, b}\}$

Processes triggered by the word-*initial* boundary require a change to the state set and the initial state. Consider the example rule in (12), which includes the word-initial boundary in its triggering context:

(12) $a \rightarrow b \;/\; \# \_\_ a$

In order to perform the correct transduction, the FST needs to distinguish between a word-initial *aa* and a non-initial *aa*. This distinction is accomplished with a separate state for #*a* and *aa*. Thus the state set needs to include states for # followed by all

[8] An exception is a deletion rule like a $\rightarrow \lambda$ / # __a. Applied simultaneously this rule maps *aaa* to *aa*. Applied left-to-right it maps *aaa* to λ.

possible $k-2$-factors. The change to the initial state (in just these cases) is $q_0 = \#$. $\mathcal{T}_{SL}$ for the rule in (12) is shown in Figure 3.5.

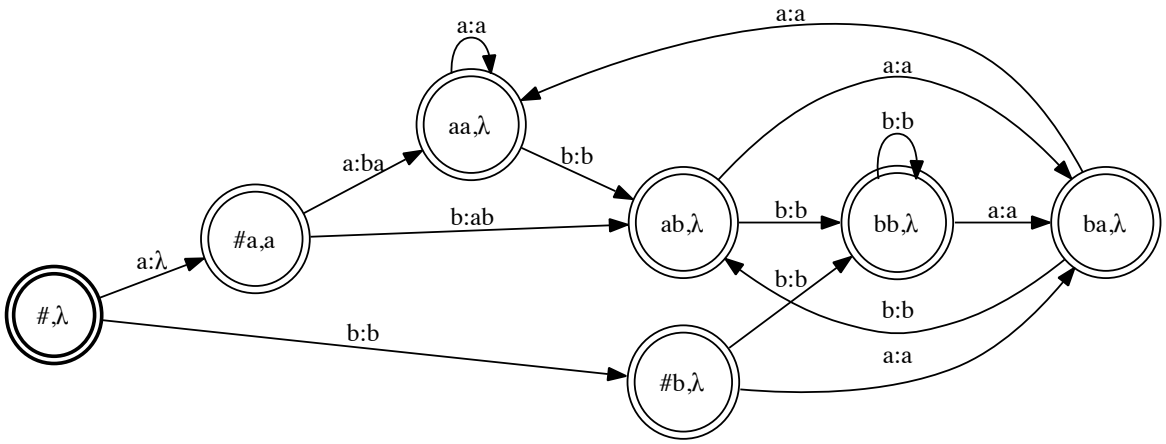


Figure 3.5: $\mathcal{T}_{SL}$ for (12), $k = 3$, $\Sigma = \{\text{a, b}\}$

3.4 Multiple Targets and Contexts

Phonologists frequently generalize phonological processes as applying to multiple targets in multiple contexts. These sets of targets and contexts are typically not arbitrary, but correspond to natural classes. Consider, for example, the nasal assimilation process found in English as well as many other languages. This process can be expressed with a rule like (13), by which a velar nasal is substituted for an alveolar one just in case the following sound is a velar stop, like /k/.[9]

(13) $\text{n} \rightarrow \text{ŋ} \; / \; \_\text{k}$

[9] This process can also be represented with an underspecified nasal, N, that takes on the place feature of a following obstruent. The distinction between these two representations of the rule are not relevant to the present discussion.

This rule clearly falls within the scope of the above constructions, with x = n, y = ŋ, $u = \lambda$, and v = k. However, the process of nasal assimilation is not fully captured by the rule in (13). For one thing, the change also takes place with v = g. Secondly, a clearly related change takes place before bilabial stops. So in fact, nasal assimilation consists of the four rules in (14).[10]

(14) a. n → ŋ / __k

b. n → ŋ / __g

c. n → m / __p

d. n → m / __b

To capture the fact that this set of rules actually describes a single process, phonologists prefer to use natural classes in their rule formalisms. The set of rules in (14) (as well as the fact that /n/ surfaces as [n] before alveolars) can instead be represented with the single rule in (15).

(15) [+nasal] → [αplace] / __[αplace, −son]

Since the FST constructions presented above assume an alphabet of segments, we must find another way to capture the fact that the rules in (14) describe a single process.[11] One possibility is to create a separate FST for each rule by the above procedure and then compose the four FSTs into a single FST by transducer composition. This method would enforce a strict ordering of the rules in (14), since composition uses the output of one function as the input to the next. In this particular example, ordering the rules would not present a problem, since the triggering contexts are distinct. For a given /n/, only one of the possible values of v (i.e., k, g, p, b) can follow it, so exactly one of

[10] Again, under the analysis with the underlying unspecified N, two additional rules are needed to change N to [n] before the alveolar stops /t/ and /d/.

[11] Using an alphabet of features is a natural solution, but one that falls outside the scope of this thesis.

the rules will apply. Another example, however, shows that the ordering enforced by composition will not always be so benign.

The (now extinct) Native American language Tonkawa has a vocalization process by which a glide becomes its vowel counterpart before a consonant or word-finally (Hoijer, 1933). The feature-based rule for this process is shown in (16), and an example is shown in (17).

(16) [-syllabic,-cons] → [+syllabic] / __{[+cons], #}

(17) Tonkawa

a.	ʔeʔejawoʔc	'I work'
b.	ʔeʔejau doːxoʔc	'I finish working'

Expanding this rule to its segmental representation gives the pair of rules in (18),

(18) a. j → i / __{C, #}

b. w → u / __{C, #}

where C ranges over the consonant inventory of Tonkawa, C = {p, t, k, k^w, ʔ, ts, s, x, x^w, h, m, n, l, j, w}. Thus the full expansion of (16) contains 32 rules, one for each of the rules in (18) for each of the 16 possible right contexts.

As members of C, the glides /j/ and /w/ should trigger vocalization of a preceding glide, meaning the input *jjj* should be mapped to *iii*.[12] Likewise, the mappings *www* ↦ *uuu*, *jwj* ↦ *iui*, and *wjw* ↦ *uiu* are expected. Note that the improbability of these input forms is not relevant. The FST that models a particular process should produce the correct output for *any* possible input. The actual subset of possible inputs that the language will make use of is determined by other factors. In SPE, for instance, morpheme structure constraints dictate the subset of Σ^* that a rule will actually see as input. But these constraints are independent of the rule itself. And in OT, the

[12] This is true for either mode of application; when a rule only has a right context, simultaneous and left-to-right application yield the same mapping. This point will be discussed further below.

conception of a process as a total function is actually built into the theory through the assumption of Richness of the Base (Prince and Smolensky, 1991, 1993).

To return to the present case, combining the two rules in (18) via composition will not yield the correct mappings for *jwj* or *wjw* regardless of how the rules are ordered, because one rule will bleed the other. This is demonstrated with the sample derivations in (19).

(19)

/wjw/		/jwj/	
/j/-vocalization	[wiw]	/w/-vocalization	[juj]
/w/-vocalization	*[wiu]	/j/-vocalization	*[jui]

Either ordering fails on at least one input, which suggests that composition is not the correct method by which a set of rules defined over strings should be combined to capture the single process normally defined with features. Instead, a single FST should be constructed for the entire set. This requires modifying the rule schema to (20),

(20) $x_i \rightarrow y_i \ / \ U \ \_\_ V$

where $U \subseteq \{\#, \lambda\} \cdot \Sigma^* \cdot$ is a set of left contexts, $V \subseteq \Sigma^* \cdot \{\#, \lambda\}$ is a set of right contexts, and $(x_i, y_i) \in X \times Y$ where $X, Y \subseteq \Sigma^*$ are sets of targets and structural changes, respectively. Crucially, the sets U, X, and V must be *finite* subsets in order for the rule to be described with a SL function. This condition will be discussed further toward the end of the chapter. The cross-product of X and Y is necessary, because it is not the case that any string from X can be rewritten as any string from Y (which would be implied by simply specifying $X \rightarrow Y$). For example, in the case of Tonkawa, it is not sufficient to define X as the set of glides and Y as the set of high vowels. The glides /j/ and /w/ must be paired with [i] and [u], respectively.

Whereas before the FST only had to look for a single uxv string, now it must keep track of a set of them. That set includes all possible uxv combinations, with $u \in U$, $x \in X$, and $v \in V$. The value of k is now the length of the longest such

uxv string. To see how this works, first consider an example process with two targets changing in the same context, as in (21). $\mathcal{T}_{ISL}$ for this process is shown in Figure 3.7 (next page).

(21) a. a → b / a __a

b. b → c / a __a

Figure 3.7 maps *aaaba* to *abaca*. Note the *b:a* transition from state *aa* to state *ab*. In the previous example (Figure 3.1), this transition was *b:ab*. Since in that example *aab* is not a rewritable *k*-factor, the *a* being held in state *aa* is returned when *b* is read in. In this example, however, *aba is* a rewritable *k*-factor, and so state *ab* is a holding state. Hence the 'b'-transition out of *aa* withholds *b*, the portion of the output being held in the destination state *ab*.

Now consider again the Tonkawa example, which has multiple targets *and* multiple contexts. For ease of reading, let C again refer to any consonant. So the set of right contexts is V = {#, C}. $\mathcal{T}_{ISL}$ for this process is shown in 3.6.

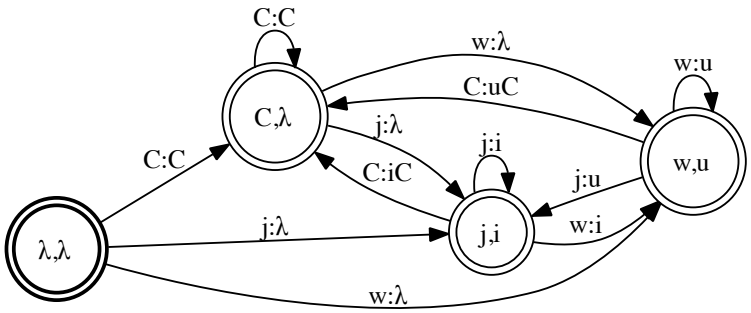


Figure 3.6: $\mathcal{T}_{ISL}$ for Tonkawa glide vocalization, $k = 2$, Σ = {C, j, w}

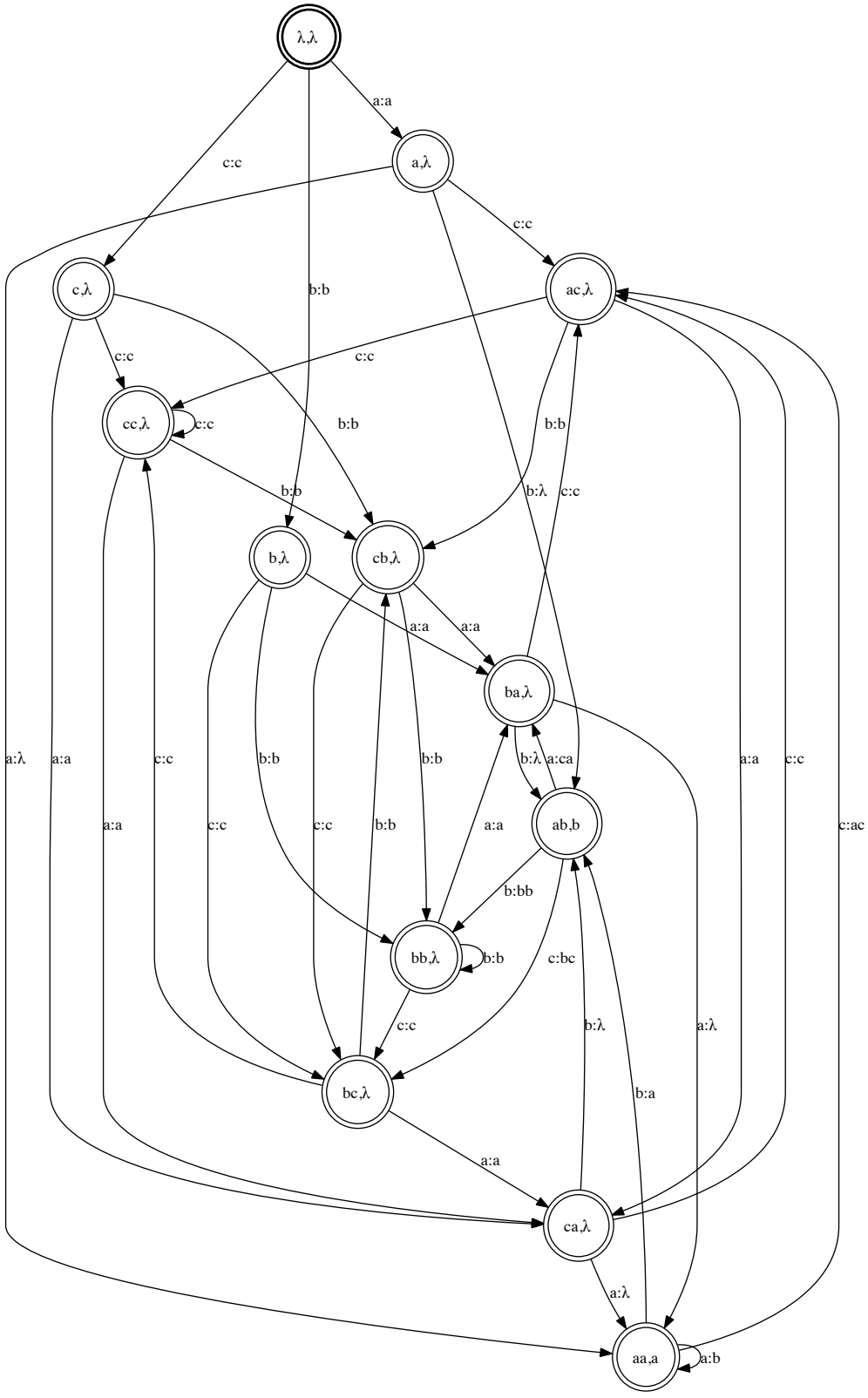


Figure 3.7: $\mathcal{T}_{ISL}$ for (21), $k = 3$, $\Sigma = \{\text{a, b, c}\}$

Note that the example in Figure 3.6 also demonstrates the special case when U = $\{\lambda\}$, since the process is only triggered by a following context. This is also a case in which left-to-right application yields the same mapping as simultaneous. To see why, consider the left-to-right application of the rule in (22) to the string *aaaa*.

(22) a → b / __a

In this rule the same string, *a*, is the target and the triggering context, a circumstance that can (as we've seen) lead to a different mapping depending on the mode of application. However, since the rule only has a following context and we are moving from the left to the right, each *a* will not become the target of the rule until *after* it has served as a context. So the rule will apply to the first *a* (triggered by the second), then the second (triggered by the third), and then the third (triggered by the fourth), to give the output of *bbba*. This is the same result as applying the rule simultaneously.

By this reasoning, in the opposite case of a rule with only a preceding context, as in (23), the two modes of application should produce different mappings.

(23) a → b / a __

And indeed this is the case.[13] $\mathcal{T}_{ISL}$ for (23) (which maps *aaaa* to *abbb*) is shown in Figure 3.8. $\mathcal{T}_{OSL}$ (which maps *aaaa* to *abab*) is shown in Figure 3.9.

Notice these FSTs contain no λ-output transitions, which means there are no holding states (and therefore the final output function assigns λ to all states). This is because 1) with no following context, once the target is seen the change can be made immediately, and 2) $|x| = 1$. If $|x| > 1$, then λ transitions are needed until the FST verifies that all of x has been seen. It turns out that the presence or absence of λ-output transitions has an effect on the learning of the process the FST describes. In particular, it affects the amount of data the learner needs to see to correctly converge on the FST. More will be said about this in Chapter 6.

[13] Though this time right-to-left application is the same as simultaneous.

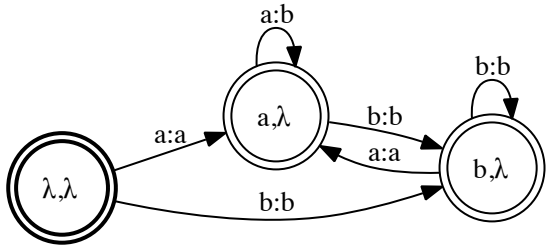


Figure 3.8: $\mathcal{T}_{ISL}$ for (23), $k = 2$, $\Sigma = \{a, b\}$

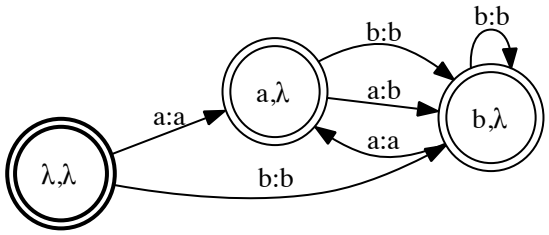


Figure 3.9: $\mathcal{T}_{OSL}$ for (23), $k = 2$, $\Sigma = \{a, b\}$

Now that it has been shown that SL functions can collectively model three different modes of rule application - simultaneous, left-to-right, and right-to-left, the diagram in Figure 2.8 can be revised to better represent the relationship between subsequential and SL functions. Some mappings (e.g., progressive harmony) can only be described with OSL functions that read the input from left-to-right, and likewise some mappings (e.g., regressive harmony) can only be described with OSL functions that read the input from right-to-left. This means the OSL functions, like the subsequential functions, are two intersecting classes, as shown in Figure 3.10.

ISL functions, on the other hand, do not describe different mappings depending on the direction in which the input is read. As a result, they are located in the diagram in the intersection of left and right subsequential. The diagram distinguishes

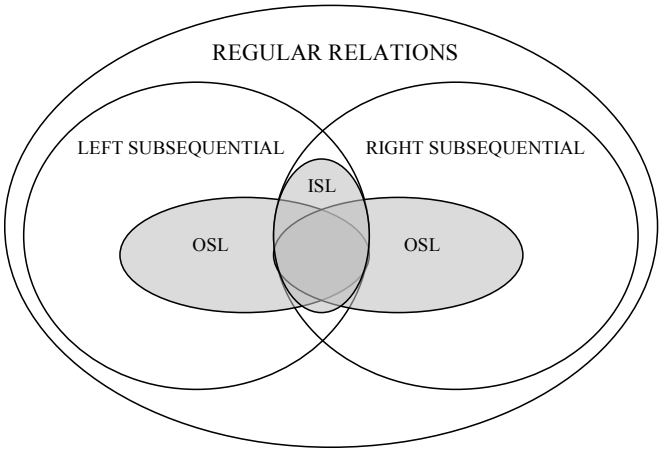


Figure 3.10: Relationship among left subsequential, right subsequential, OSL, and ISL functions

four regions of ISL functions based on whether the mapping corresponds to left-to-right OSL (written $\overrightarrow{OSL}$), right-to-left OSL (written $\overleftarrow{OSL}$), both, or neither. These four possibilities are listed in Table 3.2 below, along with examples for each.

Table 3.2: Subregions of the Class of ISL Functions

ISL $= \overrightarrow{OSL}$	Rule (18)
ISL $= \overleftarrow{OSL}$	Rule (23)
ISL $= \overrightarrow{OSL} = \overleftarrow{OSL}$	Rule (11)
ISL $\neq$ OSL	See Figure A.4

This chapter has so far demonstrated that SL functions can model the mappings for rules that fit any of the following schemas:

(24) a. $x_i \rightarrow y_i \ / \ U \ \_\_ V$

b. $x_i \rightarrow y_i \ / \ \_\_ V$

c. $x_i \rightarrow y_i \ / \ U \ \_\_$

Collectively, these three schemas cover all local substitution processes, which are among the most common phonological processes observed in the world's languages. Also among the most common processes are deletion and insertion, which can also be modeled with SL functions.

In what follows, the term deletion is being used as a blanket term for processes like syncope, apocope, apheresis, etc. - any process that can be represented with a rule like in (25), where all targets x are paired with $\emptyset$.[14]

(25) $x_i \rightarrow \emptyset \ / \ U \_\_ V$

Consider the example of a deletion process in Coeur d'Alene in which /n/ is deleted before a voiceless alveolar fricative (Johnson, 1975):

(26) Coeur d'Alene

a. /hənsti:məlgʷis/ ↦ [hɪstɛ:milgʷɛs] 'my relative'

b. /snt∫'ɛmqəns/ ↦ [sɪnt∫'amqəs] 'its tip'

c. /t∫ɛgʷiχəns/ ↦ [t∫əgʷt∫əgʷaχəs] 'his wings'

(27) /n/ → ∅ / __{s, ɬ}

$\mathcal{T}_{SL}$ for this rule is shown in Figure 3.11. To improve the readability of the FST, following Beesley and Karttunen (2003) the symbol ? is used as an abbreviation for any segment in the inventory that isn't /s/, /n/, or /ɬ/. This FST differs minimally from the ones used above for substitution processes. In substitution processes, a target string x_i is held until the requisite context for the change is verified. If it is verified, then it is replaced with y_i; otherwise x_i is returned unchanged. For Coeur d'Alene, the result of assigning y to $\emptyset$ is that, if the context of the deletion is found, the held target is simply not returned.

[14] The use of $\emptyset$ follows the notation from SPE (Chomsky and Halle, 1968), though this symbol typically designates the empty *set* rather than the empty *string*.

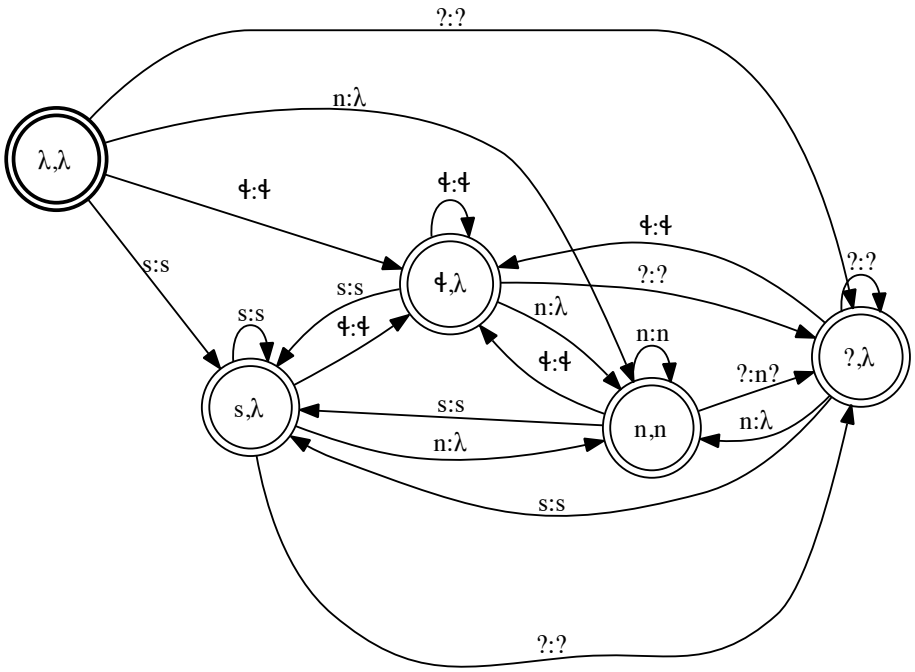


Figure 3.11: $\mathcal{T}_{SL}$ for Coeur d'Alene n-deletion, $k = 2$, $\Sigma = \{\text{s, n, ?, ɬ}\}$

The third and final type of process that will be discussed in this chapter is insertion, which again is being used as a blanket term for epenthesis, prothesis, anaptyxis - any process that can be represented with a rule like in (28).

(28) $\emptyset \rightarrow y \ / \ U \ \_\_V$

An example of insertion is a process in Czech by which the glide [j] is epenthesized between /i/ and any other vowel (Harkins, 1953). Note: here V = vowel.

(29) $\emptyset \rightarrow$ j / i __V

(30) Czech

a. biografu ↦ bijografu 'movie theatre'

b. pionirski ↦ pijonirski 'pioneer'

Figure 3.12 presents $\mathcal{T}_{SL}$ for this process. Again in the interest of readability, a reduced alphabet of $\Sigma = \{\text{i, j, ?, V}\}$ is assumed, where ? is any segment in the inventory aside from those already listed.

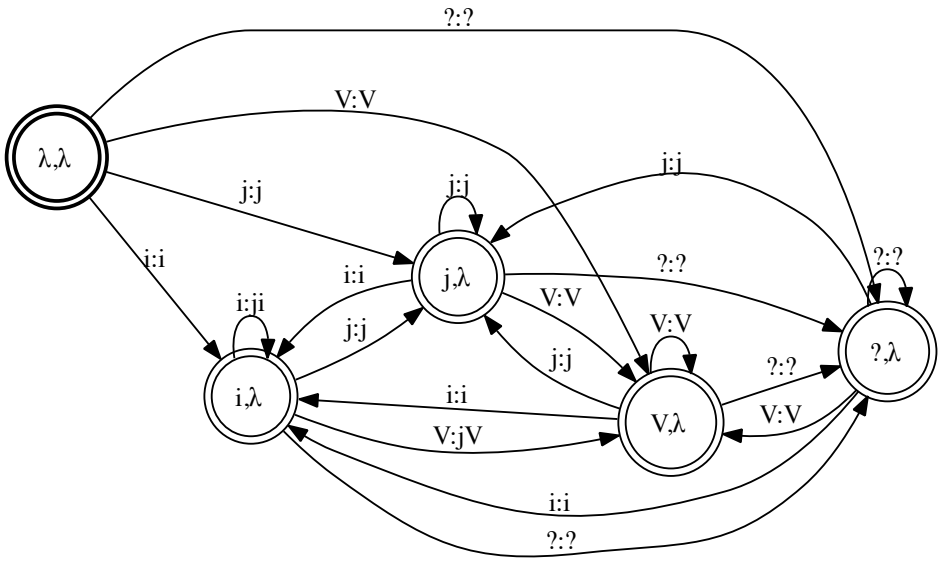


Figure 3.12: $\mathcal{T}_{SL}$ for Czech *j*-epenthesis, $k = 2$, $\Sigma = \{\text{?, V, j, i}\}$

Thus Strictly Local functions can model the local phonological processes of substitution, deletion, and insertion. More generally, by conjecture, any such process for which the target and the triggering context form a contiguous substring bounded by some length k can be modeled with a SL function. This observation is formalized in the following proposition, the proof of which is being left for future work.

Proposition 1. *The mapping of a rule $A \rightarrow B$ / C __D is SL iff CAD is a finite language.*

The requirement that CAD be finite is what rules out a process like in (4), repeated below in (31).

(31) /i/ $\rightarrow$ [u] / x __(x is a string with an even number of obstruents)

The set of strings that satisfy the condition on x (i.e., C) is infinite, so by extension CAD is infinite. To see why it then follows that the process is not SL, recall the defining property of SL functions (expressed in Definitions 4 and 5) that the same suffix of length $k-1$ implies the same tails. Consider two input strings w, w' from an alphabet $\Sigma = \{\text{d, i}\}$. Let $w = (id)^k$ and $w' = d(id)^k$, for some arbitrary $k \in \mathbb{N}$. If k is even, then the tails of w include (λ, λ), (d, d), and (i, u), while the tails of w' include (λ, λ), (d, d), and (i, i). If k is odd, then the tails of w include (λ, λ), (d, d), and (i, i) while the tails of w' include (λ, λ), (d, d), and (i, u). Thus no matter what value we choose for k, w and w' have the same suffix of length $k-1$ but they have different sets of tails. This means the process is not SL for any k.

In this way, the SL classification delimits the range of processes that are phonologically possible and accounts for why a process like (31) is unattested and phonologically implausible. Of course, there are other kinds of non-SL processes that *are* attested, namely long-distance phenomena like vowel harmony with transparent vowels (Nevins, 2010; Gainor et al., 2012; Heinz and Lai, 2013) and long-distance consonant harmony (Hansson, 2001; Rose and Walker, 2004; Luo, 2013) and dissimilation (Suzuki, 1998; Bennett, 2013; Payne, 2013). More will be said about these types of non-SL processes in Chapter 7. The next chapter turns to the discussion of metathesis processes and establishes the conditions under which they too can be categorized as SL.

Chapter 4

METATHESIS

The previous chapter demonstrated that substitution, deletion, and insertion processes can be modeled with SL functions provided the target and triggering context form a contiguous substring bounded by some length k. A natural question is how many of these processes actually meet this condition. There are certainly examples of substitution that do not meet it, such as long-distance consonant harmony (Hansson, 2001; Rose and Walker, 2004) and dissimilation (Suzuki, 1998; Bennett, 2013) (these exceptions will be discussed further in Chapter 7). It is less clear whether there exist similar long-distance cases of deletion and insertion. A review of cross-linguistic typological surveys of these processes could therefore provide more evidence for the empirical coverage of SL functions.

In addition to identifying the kinds of processes that are SL, it is useful to examine SL and non-SL processes of a similar type side-by-side. The goal is to better convey what makes a process SL by seeing how it differs from a non-SL version of the same type. To that end, this chapter turns to metathesis processes, by which the segments in a word are reordered. Metathesis, or more generally, phonological movement, serves as a useful case study for closely examining the SL/non-SL distinction, in part because it has inspired many cross-linguistic surveys that provide ample evidence for its typological range (see Grammont, 1906; Ultan, 1978; Blevins and Garrett, 1998, 2004; Hume, 2000; Buckley, 2011, among others). Interestingly, the investigation presented here will reveal that the SL/non-SL distinction corresponds to the synchronic/diachronic

distinction. Specifically, it will show that all synchronic metathesis is SL, while certain diachronic displacement cases are not.[1]

As a secondary goal, this chapter will also provide computational evidence that metathesis differs minimally from the more 'common' processes of substitution, deletion, and insertion, thereby supporting the research that has aimed to rescue metathesis from its original reputation as a marginal or exceptional process. Prior to Grammont (1906) and Ultan (1978), metathesis was largely considered to be non-regular or sporadic, attributed to speech errors or diachronic sound change or otherwise excluded from the synchronic phonological grammar. This tendency to explain metathesis with grammar-external factors reflected the view that it is somehow exceptional. In some ways this exceptionality is echoed in the grammatical treatments of metathesis: Hume (2001) notes that metathesis rules differ from those for substitution, deletion, and insertion because they target more than one segment (e.g., $sk \rightarrow ks$). Likewise, Chomsky and Halle (1968)'s treatment of the Kasem metathesis process exemplified in (1) also involved a rule (shown in (2)) that differs from those more commonly proposed for phonological changes.

(1) Kasem

/pia+i/ $\mapsto$ [paii] 'yams'

(2) SD: [+voc, -cons]$_1$, [-cons]$_2$, [+voc, -cons]$_3$

SC: 1 2 3 $\rightarrow$ 2 1 3 except when 2 = 3 = [a]

[1] A note on terminology. A range of phenomena have been discussed under the banner of 'metathesis' that vary in terms of the number of segments that are relocated (one or two) and whether the movement is local or long-distance. Authors have likewise used a variety of terms to distinguish these patterns: reciprocal (two segments) versus simple (one segment) metathesis, interversion (contiguous segments) versus metathesis (non-contiguous segments), transposition (local) versus displacement (long-distance), etc. In this chapter the following three categories will be distinguished: 1) local metathesis, in which two adjacent segments switch positions, 2) long-distance metathesis, in which two segments switch positions around overt intervening material, and 3) displacement, in which a single segment or feature is relocated in the string.

Chomsky and Halle note that this analysis of Kasem metathesis uses 'powerful transformational machinery of the sort that is used in the syntax. This increase in the power of the formal devices of phonology did not seem fully justified since it was made only to handle a marginal type of phenomenon' (427). Though their proposed rule does not actually have the level of power needed for syntax (it is regular, and in fact subsequential), their remark again reflects the view that metathesis is somehow exceptional.[2]

OT analyses of metathesis, on the other hand, rescued it from the fringes, since under the assumptions of Correspondence Theory (McCarthy and Prince, 1995, 1999) metathesis is neither exceptional nor unexpected. However, the actual treatment of metathesis in a constraint-based grammar can still render it exceptional (albeit indirectly) when the constraints that account for it increase the formal complexity of the grammar. OT analyses of metathesis introduced the family of LINEARITY constraints (McCarthy and Prince, 1999; McCarthy, 2000b; Hume, 1995, 1998, 2001); if a certain segment is disfavored in a certain context by a markedness constraint, then segment reordering will be preferred in a language that ranks faithfulness to linear order below that markedness constraint. Violations of LINEARITY are based on the number of precedence relations in the input string, which means the number of possible violations increases quadratically with the length of the string (Heinz, 2005b). Such constraints increase the grammar's formal complexity (Frank and Satta, 1998), and this added power does not even guarantee the correct predictions. Using a counterexample from Kwara'ae, Heinz (2005b) shows that the gradient nature of LINEARITY wrongly predicts that the most local reordering will be universally preferred.

This chapter will instead provide computational evidence for the unexceptionality of metathesis by categorizing it as SL along with local substitution, deletion, and insertion. In doing so it will assume the alternative view of metathesis as not resulting from transposition but rather from a copying process followed by deletion of the original segment the copy was made from (Blevins and Garrett, 1998, 2004; Heinz, 2005a;

[2] Though in a different chapter they (somewhat contradictorily) refer to metathesis as 'a perfectly common phonological process' (361).

Chandlee and Heinz, 2012). Since copying is a particular type of insertion, metathesis patterns can be decomposed into the more 'elementary' operations of insertion and deletion. Such a view lessens the unwarranted exceptionality of metathesis. In addition, the decomposition analysis allows for a unified treatment of local metathesis and long-distance displacement, for which a transposition analysis is less intuitive. The next section will detail and motivate this alternative view of metathesis as the result of copying and deletion. §§4.2 and 4.3 will then show that local and long-distance metathesis, respectively, are SL processes. §4.4 will discuss examples of both SL and (diachronic) non-SL displacement, and then §4.5 will summarize the findings presented in this chapter.

4.1 Decomposing metathesis

Consider the classic case of CV-metathesis in the Rotuman phase alternation. All content words in Rotuman have a 'complete' form and an 'incomplete' form; the distinction more or less corresponds to definite versus indefinite. In many cases, the incomplete form is derived from the complete form via word-final consonant-vowel metathesis (Churchward, 1940), as exemplified in (3):[3]

(3) Rotuman

a. hosa ↦ hoas 'flower'

b. hula ↦ hual 'moon'

c. tiko ↦ tiok 'flesh'

A single, transposition rule for this process is shown in (4), but the same surface form is derived with the two strictly ordered rules in (5).

(4) $C_1V_2 \rightarrow V_2C_1 \ / \ V_1\_\#$

(5) a. Copy: $\emptyset \rightarrow V_2 \ / \ V_1\_CV_2\#$

[3] The incomplete phase can also be derived via deletion, umlaut, or diphthongization. See McCarthy (2000b) and references therein for accounts of what conditions the choice of process.

b. Delete: $V_2 \rightarrow \emptyset$ / V_1V_2C__#

Rule (5-a) copies the vowel that 'moves' and rule (5-b) deletes the original vowel the copy was made from. In a similar proposal, Blevins and Garrett (1998) analyze the Rotuman pattern as anticipatory co-articulation of V_2, which perseverates over the consonant to compensate for its reduction when V_1 is stressed. The phonetic motivation for their account supports the assumption in (5) that it is the vowel that is copied, though the same surface form could be derived by copying the consonant (e.g., $V_1C_1V_2 \mapsto V_1C_1V_2C_1 \mapsto V_1V_2C_1$).

Another example that involves vowel copying is found in Kwara'ae, which has a very similar metathesis pattern as Rotuman, as shown in (6-a) and (6-b) below. As discussed by Heinz (2005a), the non-metathesized form is used as a citation form (for careful speech), while the metathesized form is the normal form used in typical discourse. In addition, Kwara'ae speakers optionally exhibit the form in (6-c) in certain focused positions.

(6) Kwara'ae

a. Citation: kulu 'heavy'

b. Normal: kuul

c. Focused: kuulu

Under the copy-delete view of metathesis, this focused form is actually the intermediate stage in which the copying, but not yet the deletion, has taken place. On the other hand, if *kulu* is mapped directly to *kuul* via a transposition rule like (4), then the derivational connection between the normal and focused forms is lost (i.e., they must be the result of completely different rules applying to the citation form). In addition, the fact that the focused form is *kuulu* and not **kulul* indicates that the vowel and not the consonant is copied.

These examples do not, however, lead to the conclusion that it is *always* the vowel that copies. Harris and Halle (2005), for example, propose a copying analysis of

the Spanish data in (7) (dashes indicate clitic boundaries).

(7) Spanish

a. véndan-lo 'Sell-PL it.'

b. véndan-lon

c. vénda-lon

In (7), the verbal plural suffix *-n* is optionally copied onto the following clitic, as in (7-b), or else metathesizes to that location, as in (7-c). That this suffix originates on the verb is evidenced by the fact that the plural on clitics is usually marked with *-s*. Harris and Halle (2005) propose a framework in which both (7-b) and (7-c) are the result of reduplication, but it is also possible to view (7-b) as an intermediate stage of the metathesis in (7-c), as was done for Kwara'ae.

Lastly, in Zoque the sequence jt metathesizes to t^j word-initially, but surfaces as jt^j everywhere else (Wonderly, 1951). These facts can be unified if the word-initial cluster is subject to both copying and deletion, but deletion does not apply to non-initial clusters.

A couple of languages also provide diachronic evidence for the intermediate form. Kiparsky (1976) attributes to Grammont (1948), Schwyzer (1953), Lejeune (1955), and Diver (1958) the view that diachronic Greek sonorant-glide metathesis (e.g., *phanjo: > phajno: > phaíno:, 'show') proceeded through a stage with two palatalized sonorants: $nj > n^jn^j > jn$). And Hock (1985) proposes an intermediate diachronic stage of metathesis that results from Latin glide epenthesis: sapiat > *sapya > *sa^{y}p^{y}a > saypa > Spanish, sepa, 'would know'.

In addition to languages with evidence of intermediate metathesis forms, languages that demonstrate independent use of the component copy and deletion processes also support the decomposition analysis. An example is found in the Ethiopian language Arbore. Hayward (1984) reports a metathesis process in which the second part of a geminate vowel switches positions with a following laryngeal that is itself followed

by a non-laryngeal consonant. The rule for this process is shown in (8). For readability, let H represent the set of [-cons, -son] sounds and C represent the set of [+cons] sounds. Examples of the process are shown in (9) - note that the laryngeal also becomes voiced intervocalically.

(8) $V_1H \rightarrow HV_1 \ / \ V_1\_\_C$

(9) Arbore

a. /zeehs+e/ ↦ [zeɦese] 'caused to melt'

b. /sooh+ne/ ↦ [soɦone] 'twisted into rope'

c. /kee?+t+e/ ↦ [ke?ete] 'planted'

The same language has an epenthesis process in which a vowel is copied between a sequence of a vowel plus a laryngeal and a non-laryngeal, as in (10). Examples are shown in (11).

(10) $\emptyset \rightarrow V_1 \ / \ V_1H\_\_C$

(11) a. /gile?+n+e/ ↦ [gile?ene] 'begged'

b. /burah+t+e/ ↦ [buraɦate] 'scattered'

c. /se?+t+aw/ ↦ [se?etaw] 'my cow'

d. /leh+t+atto/ ↦ [leɦetatto] 'that ewe'

Thus Arbore has two different processes that both generate a V_1HV_1C sequence. Under the copy-delete view of metathesis, the examples in (9) are the result of copying followed by deletion (i.e., $V_1V_1HC \mapsto V_1V_1HV_1C \mapsto V_1HV_1C$), while the examples in (11) are the result of just copying (i.e., $V_1HC \mapsto V_1HV_1C$).

As another example, Najdi Arabic has a metathesis process by which a $C_1aC_2C_3at$ template surfaces as $C_1C_2aC_3at$ when C_2 is a guttural (i.e., uvular, pharyngeal, or laryngeal) consonant (al Mozainy, 1981; al Mozainy et al., 1985), as shown in (12-a). The same dialect also exhibits an independent process that deletes /a/ in the context

C__CaC (Abboud, 1979), as in (12-b).

(12) Najdi Arabic

a. naʕʝat ↦ nʕaʝat ‘ewe’

b. /sakan-/ ‘dwell’ skanat ‘she dwelled’

In another dialect, Negev Bedouin Arabic, when a prefix ending in /a/ is attached to a CC-initial template, /a/-epenthesis breaks up the two consonants if the first one is guttural (Blanc, 1970). An example is shown in (13).

(13) Negev Bedouin Arabic

ja+ħlam ↦ jaħalam ‘he dreams’

Blevins and Garrett (1998) propose that the Najdi Arabic deletion process (CaCaC ↦ CCaC) and the Negev Bedouin epenthesis (aʕC ↦ aʕaC) existed before the dialectal split, and that both dialects have the epenthesis. The difference is that Najdi employs the subsequent deletion, with a result that looks like metathesis (aʕC ↦ aʕaC ↦ ʕaC).

Still more evidence comes from analyses of dialect variation that are simplified by the assumption of separate copy and deletion processes. For example, Mills and Grima (1980) propose that what appears to be $C_1VC_2 \mapsto C_1C_2V$ metathesis in Leti and Moa is in fact the result of ordered epenthesis and syncope rules. Sample derivations are shown in (14). Note that the epenthesis process is in fact copying, since the inserted vowel matches the stem vowel.

(14) Leti and Moa

	/ulit/	/arak/	/irun/
Epenthesis	uliti	araka	irunu
Syncope	ulti	arka	irnu
	‘skin’	‘liquor’	‘nose’

They extend this analysis to words that end in VC by positing an abstract underlying [+cons] segment, as in the Leti derivations in (15). This abstract segment (represented by X) is also deleted, which triggers compensatory lengthening of the preceding vowel.

(15) Leti

	/tuXin/	/weXin/
Epenthesis	tuXini	weXini
Syncope	tuXni	weXni
X-Deletion	tu:ni	we:ni
	'coco shell'	'pretty'

Their evidence for this analysis is that it accounts for a difference between Leti and Moa with a simple reordering of these component rules. In Moa, from the same UR the correct SR is derived by ordering X-deletion before syncope. This bleeds the application of syncope, which only targets vowels between consonants.

(16) Moa

	/tuXin/	/weXin/
Epenthesis	tuXini	weXini
X-Deletion	tuini	weini
Syncope	–	–
	'coco shell'	'pretty'

Thus a surface difference between the two languages can be explained with simple rule ordering, but only if the metathesis effect is the result of two separate processes. Another difference between Leti and Moa that can be explained with this assumption is what happens to nouns when the third person possessive morpheme is attached. The morpheme is *-n* in Leti and *-ni* in Moa. As shown in (17), the morpheme triggers epenthesis and syncope in Leti but only syncope in Moa (shown in (18)).

(17) Leti

	/tutu+n/	/ati+n/	/mata+n/
Epenthesis	tutunu	atini	matana
Syncope	tutnu	atni	matna
	'its point'	'his heart'	'his eye'

(18) Moa

	/tutu+ni/	/ati+ni/	/mata+ni/
Syncope	tutni	atni	matni
	'its point'	'his heart'	'his eye'

This analysis reduces the observed differences between the two languages to a difference in the morpheme - the presence of a vowel in the Moa suffix makes epenthesis unnecessary. Without the independence of the epenthesis and deletion processes, the explanation would have to be that Leti has metathesis in all cases and Moa has metathesis in some cases and syncope in others. The copy-delete analysis of metathesis thus unifies the observed facts for two languages that are expected to behave similarly.[4]

Also in the domain of language variation, Blevins and Garrett (1998) suggest that the difference between Polish and Ukrainian exemplified in (19) is explained if Polish developed copy plus delete but Ukrainian developed only the copying.

(19)

Proto-Slavic	Polish	Ukrainian	
*melko	mleko	molokó	'milk'

Other proposals have been put forth that differ from the copy-delete analysis but still treat metathesis effects not as movement but as the result of more elementary operations. Demers (1974), for example, proposes that both of the Lummi forms in (20) share the underlying form /c'əʔsə+t+ŋ+s/. The different surface forms are not

[4] van der Hulst and van Engelenhoven (1995) also presents a non-metathesis account for Leti grounded in Government Phonology (Kaye et al., 1985; Charette, 1988, 1990; Harris, 1990), in which the variable position of a vowel is due to delinking/reassociation to a syllable template. A Correspondence Theory account is presented in the study of Hume (1998) mentioned above.

the result of metathesis, but unstressed vowel deletion.[5]

(20) Lummi

a. c'əstɪɲs 'he's getting hit'

b. c'səstɪɲs 'someone hit him'

And McCarthy (1989), under an assumption of planar segregation of vowels and consonants, proposes that CV metathesis effects are the result of the vowel spreading across a consonant. Hume (1991) argues against this analysis based on incorrect predictions it makes for Maltese Arabic[6], but still proposes an autosegmental, lexical phonology account of the data in (21) in which metathesis is the result of deletion and epenthesis.[7]

(21) Maltese

a. korob 'to groan (perfective)'

b. yo-korb-u 'they groan (imperfective)'

In (21), a syncope rule that deletes unstressed vowels in non-final open syllables applies to yV-korob (V is an unassociated vowel slot on the skeletal tier) to dervie yV-krob, but this only deletes the V slot associated to the vowel melody [o]. The vowel melody itself is left floating and reassociates to the prefix V slot, to derive yo-krob. On the next cycle syncope applies again to give yo-krb-u, which again leaves the [o] melody floating. In the post-lexical cycle, another V slot is epenthesized before the unsyllabified medial sononant, to give yo-kVrb-u. The floating [o] associates to this new V slot to give yo-korb-u. Thus, once again, what looks like metathesis is analyzed as the result of independently-motivated rules of deletion and insertion.

---

[5] The underlying ʔ is deleted by a separate process.

[6] Specifically, Hume (1991) shows based on an interaction of metathesis and guttural assimilation that the metathesis must take place *after* plane conflation, not before as assumed by McCarthy (1989).

[7] See also Besnier (1987) for a similar autosegmental account of Rotuman metathesis.

Given the collective evidence in support of the decomposition view of metathesis, the next section will turn to the computational analysis and show how the component copying and deletion processes can be modeled with SL functions.

4.2 Local Metathesis

Consider again the example of Rotuman metathesis and the proposed rule for the copying component, repeated in (22) and (23), respectively.

(22) Rotuman

a. hosa $\mapsto$ hoas 'flower'

b. hula $\mapsto$ hual 'moon'

c. tiko $\mapsto$ tiok 'flesh'

(23) $\emptyset \rightarrow \mathrm{V}_2 \ / \ \mathrm{V}_1 \_\mathrm{CV}_2\#$

This rule is similar to the insertion rule considered in the previous chapter, though the difference between copying and general insertion needs to be acknowledged. Whereas general insertion was handled by simply pairing the inserted string with $\emptyset$ to fill out the $x_i \rightarrow y_i$ portion of the rule, copying requires matching y_i to some portion of u or v. In (23), this is indicated with indexing that identifies the inserted vowel as being identical to the word-final vowel. This requirement that distinguishes copying from general insertion is formalized in (24).

(24) $\emptyset \rightarrow y \ / \ u \_\_v$, where $u = u'yu'' \vee v = v'yv''$

The condition on (24) requires the inserted string to be a substring of either the left or right context of the insertion point. A rule of this form can still be submitted to the construction procedure for a Strictly Local transducer. Figure 4.1 shows $\mathcal{T}_{SL}$ for the Rotuman copy process. Note that this FST is minimized, for ease of reading.[8]

[8] The construction procedure for SL FSTs presented in the previous chapter does not create the minimal transducer. An alternative construction that does produce the minimal one is included in

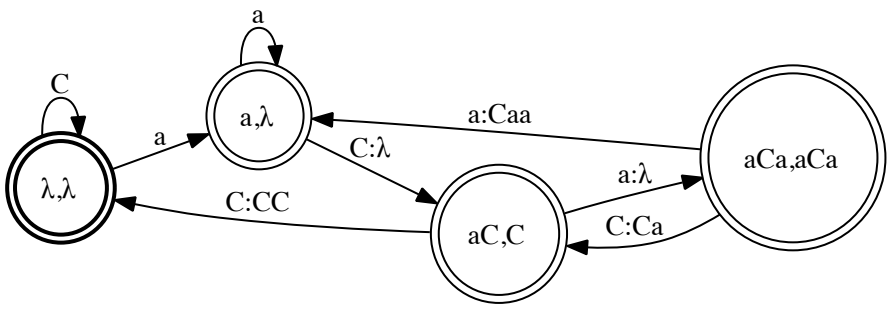


Figure 4.1: $\mathcal{T}_{SL}$ for Rotuman copying, $k = 4$, $\Sigma = \{\text{C, a}\}$

The FST in Figure 4.1 was constructed with the alphabet $\Sigma = \{\text{C, a}\}$, with C representing any consonant and /a/ being the only vowel. Though any vowel can serve as the left context of the copying, the epenthesized vowel must match the one in the right context (i.e., VCa is rewritten as VaCa for any V and any C). And this is true for all vowels, not just /a/ (i.e., VC$i \mapsto$ ViCi, VC$u \mapsto$ VuCu, etc.). Thus Figure 4.1 actually presents a subgraph of the complete FST for Rotuman copying, as the complete graph includes separate states and transitions for all possible C, V, and V_1 combinations.

Since this is a word-final process, the copied vowel is inserted via the final output function. For example, the input form aCa will conclude in state aCa, at which point the current output will be just a. The final output function will append aCa to this output, for the correct mapping of $aCa \mapsto aaCa$. The second step to achieve metathesis is to delete the original vowel that the copy matches. For Rotuman, this process corresponds to the rule in (25).

(25) $\quad \text{V}_2 \rightarrow \emptyset \ / \ \text{V}_1\text{V}_2\text{C}\_\#$

$\mathcal{T}_{SL}$ for this rule is shown in Figure 4.2, again minimized and again with an abbreviated alphabet of {C, a}.

---

the appendix.

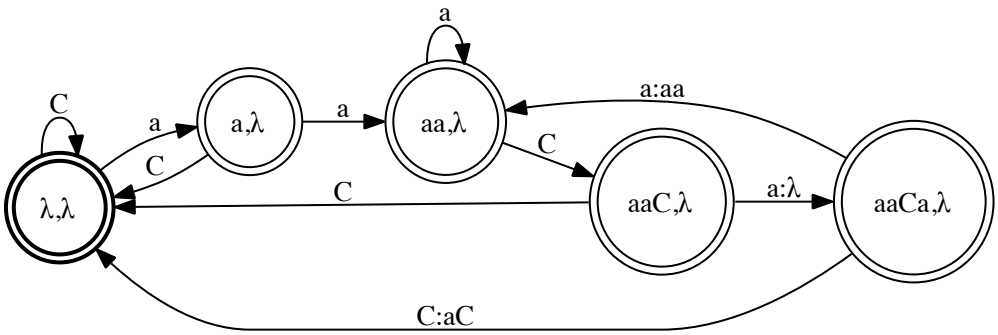


Figure 4.2: $\mathcal{T}_{SL}$ for Rotuman deletion, $k = 5$, $\Sigma = \{\text{C, a}\}$

Figure 4.1 maps *aCa#* to *aaCa#* and Figure 4.2 maps *aaCa#* to *aaC#*. Their combined effect is to map *aCa#* to *aaC#*, which corresponds to word-final CV metathesis.

Examples like in Rotuman and Kwara'ae have traditionally been categorized as local metathesis, because the segments that change position are adjacent to one another. Additional examples of local metathesis are found synchronically in Chawchila (Newman, 1944), Yuman languages (Langdon, 1976), Western Munster Irish (Malone, 1971), Munyo and Rendille (Heine, 1980), Lithuanian (Seo and Hume, 2001), Kui (Winfield, 1928), Dɛg (Crouch, 1994), Bisayan dialects (Zorc, 1977), Koṇekor Gadaba (Bhaskararao, 1980), Helong and Dawanese (Steinhauer, 1996), Korean, Tagalog (Ultan, 1978), and Kurmanji (Kahn, 1976), and diachronically in West Somerset English (Elworthy, 1875), the South-Central group of Dravidian languages (Krishnamurti, 1955, 1978; Subrahmanyam, 1983), Mandaic languages (Malone, 1971, 1985), and Attic and Ionic Greek (Ultan, 1978). See Blevins and Garrett (1998), Hume (2000), and Buckley (2011) for additional examples.

But metathesis in general is not restricted to adjacent segments. Another category, often called long-distance metathesis, involves non-adjacent segments surfacing in each other's positions. The next section turns to these long-distance patterns.

4.3 Long-Distance Metathesis

This section presents examples of metathesis around intervening segments and shows that in fact such processes are also SL provided that the number of segments between the segments that metathesize is bounded. This condition appears to be met in all synchronic cases.[9] An example from Cuzco Quechua (Davidson, 1977) is shown in (26). In this process, a glide and a liquid metathesize around an intervening vowel.

(26) Cuzco Quechua

yuraq ↦ ruyaq 'white'

Under the copy-delete analysis, this metathesis mapping is the result of the four processes shown in (27).

(27) a. $\emptyset \rightarrow$ r / __Vr

b. r $\rightarrow \emptyset$ / rV__

c. $\emptyset \rightarrow$ y / yrV__

d. y $\rightarrow \emptyset$ / __rVy

Note that the form of these rules assumes they will be applied in the order listed. In particular, the context of the *y* insertion (27-c) and deletion (27-d) assumes that the *r* insertion and deletion has already taken place. It is true, however, that the same final result could be obtained if the two *y* rules were applied before the two *r* rules (in which case the *y* would be included in the context of the *r* rules). The lack of independent evidence to determine which of these two orderings is correct favors the formulation of the rules shown in (28).

---

[9] Mielke and Hume (2001) actually argue that all synchronic metathesis is local, and long-distance cases like the ones presented in this section tend to be non-regular or are otherwise amenable to a non-movement analysis. Since, as will be shown, synchronic long-distance metathesis is still SL, whether or not these processes should be considered metathesis is tangential to the main point. However, if Mielke and Hume are correct, the evidence is strong that synchronic phonological movement is very restricted.

(28) a. $\emptyset \rightarrow$ r / __(y)Vr

b. r $\rightarrow \emptyset$ / r(y)V__

c. $\emptyset \rightarrow$ y / y(r)V__

d. y $\rightarrow \emptyset$ / __(r)Vy

With the rules in (28), many possible orderings will result in the correct $yVr \mapsto rVy$ mapping (i.e., any ordering so long as (28-a) precedes (28-b) and (28-c) precedes (28-d)). And just as in Rotuman, each of these rules can be modeled with a SL FST. Thus we have no computational reason to distinguish cases of local and long-distance metathesis: both are SL. What is crucial to the Cuzco Quechua metathesis being SL, however, is that the amount of material that intervenes between the two segments involved in the metathesis is bounded. In this case, the intervening material is a single vowel, so it is bounded by length 1. This bound allows us to identify the k-value of the SL function, which for the rules in (28), is either 3 or 4. Additional examples of SL metathesis with intervening segments are found in Gashowu (Newman, 1944), Yuman languages (Langdon, 1976), Madagascan dialects, Gayo, Agde French, Caribbean French Creole, Algarve Portuguese (Ultan, 1978), Salishan languages (Noonan, 1997), and Hebrew (Horowitz, 1960). Again see Hume (2000) for additional examples.

Are all long-distance metathesis patterns SL? A non-SL version of long-distance metathesis would be a process like the one in Cuzco Quechua in which there is no upper bound on the amount of material that intervenes between the two segments that metathesize. Reviews of the typology of metathesis (Ultan, 1978; Blevins and Garrett, 1998, 2004; Hume, 2000, 2001; Buckley, 2011; Chandlee et al., 2012) suggest that such patterns do not occur synchronically. Possible diachronic cases include Ilocano, Spanish, and Breton. In the Tagalog > Ilocano data in (29) and (30) (from Anttila (1989)), the segments /t/ and /s/ appear to metathesize regardless of how many segments intervene.

(29) Tagalog > Ilocano

a. taŋis > saŋit 'weep'

b. tubus > subbot 'redeem'

c. tigis > sigit 'decant'

d. tamis > samqit 'sweet'

(30) a. gatos 'trillion' > gasut 'hundred'

b. tastas > satsat 'rip'

Another, often-cited example is the $r...l > l...r$ change in Spanish (Penny, 1991) and Breton (Ultan, 1978):

(31) Old Spanish > Spanish

a. parabla > palabra 'word'

b. periglo > peligro 'danger'

c. miraglo > milagro 'miracle'

(32) Leonese > Breton

a. mervel > melver 'to die'

b. teûrel > teûler 'to throw'

c. brélim > blérim 'grindstone'

Ultan (1978) suggests that this pattern is influenced by an independently-motivated process of regressive dissimilation, whereby $r...r > l...r$, which is extended by analogy to words with $r...l$. Similarly, Wanner (1989) proposes, at least in the case of Spanish, that the /l/ was first changed to /r/ to create a preferred stop-/r/ onset cluster, and then the other /r/ was changed to /l/ due to liquid dissimilation. Regardless of the correct account of these long-distance metathesis cases, as diachronic processes they do not disprove the claim that synchronic metathesis is SL.

There is, however, another process called displacement that is often discussed along with metathesis but differs in that only a single segment (or feature or secondary articulation) moves from its original position. The movement in these displacement

processes appears to be less restricted than in metathesis. The next section will discuss the extent to which these processes can be categorized as SL.

4.4 Displacement

Among displacement patterns there is a prevalence of liquid movement, and a careful examination of how these processes vary reveals the crucial distinction between SL and non-SL displacement. First, Lipski (1990) describes an /r/-displacement phenomenon in many Spanish dialects, as exemplified in (33).

(33) Spanish

a. abarcar > abracar 'to include'

b. cabestro > cabresto 'halter'

c. cobertor > corbetor 'bedspread'

The leftward movement of /r/ in these examples is not as unbounded as it appears at first glance. The movement in (33-a) is in fact identical to local metathesis, as the /r/ just shifts to the other side of the preceding vowel. It has already been established that such a process is SL. In (33-b) and (33-c), the /r/ moves farther, but still not unboundedly far; it moves to the corresponding position in the adjacent syllable (i.e., from onset to onset or coda to coda). Since languages restrict the amount of material that can appear in a syllable, there is in fact a bound on how many segments intervene between the /r/'s original and final positions. Thus (33-b) and (33-c) are the result of the processes represented in (34) and (35), respectively.

(34) Onset-to-onset

a. Copy: $\emptyset \rightarrow$ r / C__V(C)Cr

b. Delete: r $\rightarrow \emptyset$ / CrV(C)C__

(35) Coda-to-coda

a. Copy: $\emptyset \rightarrow$ r / V__CVr

b. Delete: r $\rightarrow \emptyset$ / VrCV__

There are also cases of rightward movement, as in the examples in (36).

(36) Spanish

a. albarda > albadra 'saddle'

b. fraude > faudre 'fraud'

Again this movement is either local, as in (36-a), or targets the onset of the adjacent syllable, as in (36-b). A corresponding pair of rules like those in (34) could then describe this movement. Lipski notes that rightward movement is less common, and indeed he provides no cases of rightward coda-to-coda movement.

As was shown in the discussion of local and long-distance metathesis, rules of the form in (34) and (35) can be modeled with SL functions. Lipski (1990) provides a large number of examples of this phenomenon, and all of them are either local (like (33-a) and (36-a)) or target the adjacent syllable position (like (33-b), (33-c), and (36-b)).[10] Indeed, Lipski notes that, 'a maximum of two syllables is ever involved in /r/-transposition' (96). Spanish /r/-displacement thus serves as a useful example of a process that at first glance may appear less constrained than the cases of local and long-distance metathesis detailed above, but in fact is still Strictly Local.

A nearly identical pattern is found in Luchonnais Gascon (Grammont, 1906; Dumenil, 1987). Examples (from Dumenil (1987)) are presented below.

(37) Latin > Gascon

a. capra > craba 'goat'

b. cambra > cramba 'bedroom'

c. tendru > trendo 'tender'

[10] There is a single exception (out of over 100 examples) in *Tribucio* > *Tiburcio*. But perhaps this is actually the result of two movements, *Tribucio* > *Tibrucio* > *Tiburcio*. Regardless, the movement still targets the adjacent syllable and so a SL function could model it.

Dumenil (1987) suggests that the displacement of /r/ could be the result of two local movements, such that the change in (37-a) was actually *capra* > *carpa* > *crapa*. The evidence for this analysis is a few examples for which the intermediate form (e.g., *carpa*) is in free variation with the final form (e.g., *crapa*). But as this free variation is only evident in a few cases, Dumenil instead generalizes the process as the liquid moving to the onset of the preceding syllable. Grammont (1906) and Blevins and Garrett (1998) likewise describe the Bagnéres-de-Luchon pattern exemplified in (38) as the liquid moving one syllable to the left.

(38) Latin > Bagnéres-de-Luchon

a. kámera > *kámbra > crambo ‘room, chamber’

b. kápra > crabo ‘goat’

c. *eskombrá:re > escrumba ‘to sweep’

Still another case of liquid displacement is attested diachronically in Italian (Vennemann, 1988), shown in (39), and as free variation, shown in (40) (Vennemann, 1996).

(39) Old Italian > Italian

a. fabula > fabla[11] > flaba > fiaba ‘fairy tale’

b. fundula > fundla > flunda > fionda ‘catapult’

(40) a. Tuscan: vetro/vreto ‘glass’

b. Calabria: crapestu/capestro ‘rope’ frinesta/finestra ‘window’

Vennemann schematizes the pattern as #CVCLV > #CLVCV and #CVCVCLV > #CLVCVCV. The first case (as in (39) and (40-a)) is again movement to an adjacent syllable onset. The second case skips over a syllable (as in (40-b)) to the first syllable

[11] Vennemann posits this form (and *fundla*) to account for the loss of the vowel.

onset.[12] The clearest generalization is that the movement always targets the first syllable onset. The same pattern is reported by (Rohlfs, 1950) for a South Italian dialect of Greek. Examples are shown in (41).

(41) Classical > South Italian Greek

a.	kopros > kropo	'dung'
b.	gambros > grambo	'son-in-law'
c.	kapistrion > krapisti	'halter'

Again, examples (41-a) and (41-b) are consistent with the adjacent-syllable analysis, while (41-c) instead suggests the movement targets the *first* syllable.[13] The question of interest for the SL status of this pattern is what happens in a word with four or more syllables. If the movement only occurs in words of two or three syllables, the movement is bounded and the process can be described with a SL function. But if, no matter how far into the word the /r/ originates, the movement occurs and targets the first syllable, then this pattern is not SL. To see why, consider the copy and deletion rules for the Greek pattern, shown in (42).

(42) a. Copy: $\emptyset \rightarrow \text{r} \;/\; \#\text{C}\_\_x\text{Cr}$

b. Delete: $\text{r} \rightarrow \emptyset \;/\; \#\text{Cr}x\text{C}\_\_$

In these rules, x represents a string of unbounded length that intervenes between the original location of the copy and its copied location (the initial onset). Recall the defining property of SL functions: the same suffix of length $k-1$ implies the same

[12] One could posit that the movement in *finestra* did target the adjacent syllable, but since *nr* is not a good onset cluster, it continued on to the initial onset: *finestra* > *finresta* > *frinesta*. But this doesn't explain the example of *capestro*, since presumably *pr* is a valid cluster. It is interesting to note that Vennemann cites the following pair for Old Italian: *capestro* > *capresto*. Perhaps, then, the actual forms for Calabria are *capestro* and *capresto* > *crapesto*, in which case the adjacent syllable analysis can be maintained. Though see Coffman (2013).

[13] Penny (1991) also gives two examples of what looks like an opposite process for Latin > Spanish, in which /r/ moves to the onset of the *last* syllable: *praesepe* > *pesebre*, *crepare* > *quebrar*. With just two examples, however, it is unclear what to make of these cases.

tails. This property can be used to show that the processes in (42) are not SL for any k. Using the deletion process to demonstrate, we consider two input strings $CrxC$ and CxC, where $|x| = k - 2$ for some $k \in \mathbb{N}$. Since both strings end in xC, both strings have the same suffix of length $k-1$.[14] But, they have different tails. If $CrxC$ is extended by r, the output string will be extended by λ, since the r is deleted according to (42-b). But if CxC is extended by r, the output string will be extended by r as well, since this r is not a target for deletion. Thus (r, λ) is in the tails of $CrxC$ while (r, r) is in the tails of CxC. Since the choice of k was arbitrary, it follows that no matter what value is assigned to k, the two strings will have the same suffix of length $k - 1$ but different tails. This means the property never holds, and so the process is not SL. A similar argument would show that the copy process is also not SL.

Both processes are, however, subsequential. More specifically, the copy process is right subsequential, while the deletion process is left subsequential. Consider the SFST in Figure 4.3.

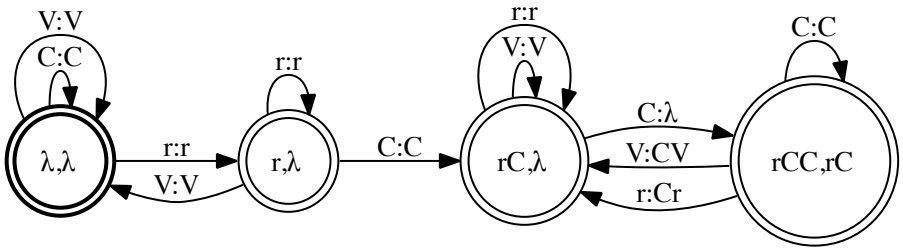


Figure 4.3: SFST for Greek copy process, $\Sigma = \{C, r, V\}$

Recall that a mapping is right subsequential if the mapping that results from processing the input string right-to-left is left subsequential. Consider the input string $CxCr$, which is subject to the Greek copy process, (42-a). Processing this string right-to-left is equivalent to processing the *reversed* string left-to-right. As demonstrated in (43),

[14] Here we are applying the definition for ISL functions, though a parallel argument proves the processes are also not OSL.

using this reversed string as input to the SFST in Figure 4.3 gives the output $rCxrC$. Since the input was revered, this output must also be reversed.

(43)

1	Input string:	$CxCr$
2	Reversed input string:	$rCxC$
3	Output for reversed input string:	$rCxrC$
4	Reversed output string:	$CrxCr$

Since the mapping from line 2 to 3 ($rCxC \mapsto rCxrC$) is left subsequential (it is described by the SFST in Figure 4.3), the mapping from line 1 to 4 ($CxCr \mapsto CrxCr$, which corresponds to (42-a)) is right subsequential.

Note that the SFST in Figure 4.3 is not a Strictly Local one, because the transitions do not always lead to the state that corresponds to the most recent input or output. In particular, the x string in the rule's context is processed between the rC and rCC states. It is necessary for the SFST to remain in these two states in order to 'remember' the fact that a prior rC substring has been found (and therefore another r must be inserted before the final consonant).[15] The SFST moves to state rCC when it reads a C, just in case this C is the final one. If it is in fact the final one, the copied r is inserted via the final output function, which appends rC to the output. If more material follows (another r or a V), the SFST returns to state rC, indicating that this was not the final C, but still remembering that the trigger for r-copying (rC) was previously found. Moving out of these states to other states that correspond to the most recent input or output (e.g., states for CrV or VVr), which is required by SL transducers, would force the FST to 'forget' that it has already seen this rC. This is the very nature of SL transduction - memory is limited to a fixed bound of previous segments. General memory such as 'rC was seen at some arbitrary previous point' is not possible.

[15] Remember this SFST will process reversed strings, so by 'final consonant' is actually meant 'initial onset'.

Moving on to the deletion component of Greek /r/-displacement, consider the SFST in Figure 4.4, which describes the mapping of (42-b). This process is left subsequential, which means it can be described with the SFST without having to reverse the strings. The SFST first determines if the input string begins with Cr, since only these strings are subject to the deletion process. If the string does not begin with Cr, the SFST proceeds to state ? and outputs all remaining input unchanged. Otherwise it looks for the next Cr cluster, deleting the r once it is located. Thus an input string like $CrxCr$ will be mapped to $CrxC$. Combing the mappings of Figures 4.3 and 4.4 gives $CxCr \mapsto CrxCr \mapsto CrxC$, which correctly models Greek /r/-displacement.

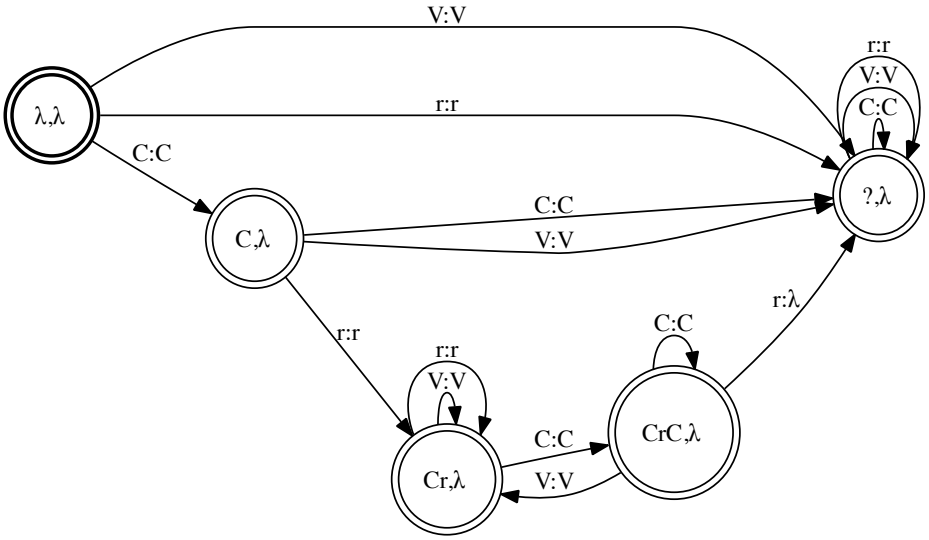


Figure 4.4: SFST for Greek deletion process, $\Sigma = \{C, r, V\}$

Thinking about it in a different way, in the case of the deletion SFST, the first r serves as a cue that another r will follow that needs to be deleted. So the SFST must locate this second r, but it does not need to hold on to any material as it does so. As a result, the unboundedness of the x string is not a problem. In the case of copying, the situation is the opposite: an r may or may not need to be epenthesized, but this time the position of the epenthesized r *precedes* the trigger of that epenthesis, and the two

are separated by the unbounded x. Thus the SFST would have to hold onto all of the material in x until it locates the trigger r, which is why the unbounded length of x is a problem. Reversing the string solves this problem, because once again the original r will proceed the location of the copy.

More generally, Chandlee and Heinz (2012) call this type of copying 'pre-pivot' copying, since the copy precedes the original. They show that such a process is right subsequential if the amount of material being copied is bounded and the left context of the copy's location is bounded. These conditions are met by the Greek copying, since only a single segment is copied and it is placed in the initial onset, making the left context the bounded string $\#C$. For the copying to be left subsequential, three conditions must be met: 1) the string between the original and copy is bounded, 2) the string being copied is bounded, and 3) the right context of the original is bounded. The alternative, 'post-pivot' copying, in which the copy follows the original, is left subsequential if at least the second two conditions are met. Metathesis patterns that involve pre-pivot copying have a corresponding process of post-pivot deletion, and patterns with post-pivot copying have a corresponding pre-pivot deletion process. The same conditions hold for the deletion processes to be either left or right subsequential. Thus, as discussed above, the post-pivot Greek deletion is left subsequential because only a single segment is being deleted and the right context of the segment to be deleted is λ (i.e., bounded by 0).[16]

Liquid displacement has thus served as a useful case study that distinguishes SL displacement in which the movement is bounded (e.g., Spanish, Gascon, and Bagnéres-de-Luchon) and non-SL but subsequential patterns in which the movement is not (e.g., Italian and Greek, pending further data). Additional examples of displacement involving segments other than liquids can now be discussed with this distinction in mind.

Several languages demonstrate aspiration or /h/-displacement, such as Marathi (Bloch, 1915; Turner, 1969; Ultan, 1978), as shown in (44).

[16] This doesn't mean there can't be any material after the deleted segment. It means the right context that *identifies* the segment to be deleted is empty.

(44) Marathi

a. kam^{h}kh > kham^{h}k ‘armpit’

b. om^{h}t^{h}h > hom^{h}t^{h}

c. məhisi > mhəis ‘buffalo’

d. pətthər > phəttər ‘stone’

Ultan (1978) describes the process as movement from a word-final or word-medial position to the word-initial sound, but Blevins and Garrett (2004) describe it as ‘aspiration (or breathy voice) has regularly shifted to word-initial position from the onset of a second syllable’ (14). This latter description is consistent with (44-c) and (44-d), but not (44-a) or (44-b) in which the movement is from coda to onset of the same syllable. In the absence of an example in which the aspiration involves distinct, non-adjacent syllables, the pattern appears to be SL. The same is true for similar processes in Romani ((45)) and Greek ((46)).[17]

(45) Old Indo-Aryan > Romani (Matras, 2002)

a. duddha > *dhud > thud ‘milk’

b. bandh > phand ‘to shut’

(46) Greek (Lejeune, 1972; Steriade, 1982)

a. /pro+hódos/ → [p^{h}roódos] ‘gone’

b. /tétr+hippos/ → [tét^{h}rippos] ‘with four horses’

Next, Langdon (1976) and Hinton and Langdon (1976) report a process of ‘glottal-stop inversion’ in the La Huerta dialect of (Tipai) Diegueño, by which an

[17] Aspiration movement has also been reported for Sanskrit, in which root-final aspiration appears instead on the root-initial consonant (e.g., /budh/, ‘know’ is sometimes [bhud]). It was first reported by Whitney (1896), who does not use the term metathesis, and indeed both Anderson (1974) and Janda and Joseph (1989) argue against a movement-based account of this data.

initial glottal stop shifts to the position immediately before the verbal root.[18]

(47) La Huerta Diegueño

/ʔnʲ+m+ka+náp/ ↦ [nʲmkaʔnáp] 'you told me'

Whether this process is SL or just subsequential depends on whether there is an upper bound on the number of segments between the word-initial position and the root (in (47) the root is *nap*). If there is such a bound, the process can be modeled with a SL FST, provided the alphabet includes a symbol that identifies the first segment of the root (e.g., a bracket, such that the input would be ʔnʸmka[nap). If not, the process is still subsequential: it could be modeled with left subsequential post-pivot copying followed by right subsequential pre-pivot deletion.

Another example with displacement of glottalization as a secondary articulation is found in the Salishan language Shuswap, in which 'the glottalized resonant of a suffix yields its glottalization to the final resonant of a stressed root' (Kuipers (1974), p. 30). Examples are presented in (48) and (49).

(48) Shuswap

a.	kʔʷul	'make'
b.	-(e)wʔs	'road'
c.	x-kʔʷúlʔ-ws-m	'make a trail'

(49)

a.	kʔʷuy	'long object lies'
b.	-(e)wʔs	'road'
c.	c-x-kʔʷúyʔ-ws	'log lying across the road'

This process is SL if there is some bound on the number (and size) of the suffixes that can attach to the root. Such a bound might generally be plausible, if we assume a root can only have a single suffix for tense, aspect, plurality, and whatever other features

[18] Hinton and Langdon (1976) actually identify the 2-subject/1-object prefix in La Huerta as $n^{y}m$ʔ-, but still propose the same movement of ʔ to pre-root position.

are encoded morphologically in a given language. But as the examples above show, Shuswap has 'lexical suffixes' which are used not for inflection but in a manner similar to compounding. Establishing a limit on the number of additions to a compound is less likely. Allowing again for a bracket or other symbol to mark the end of the root (e.g., xkʔʷúl]wʔsm), this process could be modeled with right subsequential pre-pivot copying and left subsequential post-pivot deletion. Thus it is subsequential but not SL.

The last example that will be discussed is a process of pharyngeal displacement attested in the interior Salishan language Colville (Mattina, 1979), by which a pharyngeal in the root is shifted to a stressed lexical suffix, as shown in (50).

(50) Colville

a.	pʕas	'scared'	c-ps-ʕáyaʔ	'senseless'
b.	q'ʷʕáy	'black'	q'ʷəy-ʕás	'black man'
c.	pʕáw	'he ran down'	pw-ən-cʕát-əlx	'they make noise running down'

The correct generalization of this process is not entirely clear. In the majority of Mattina (1979)'s examples, the root pharyngeal becomes the onset of the next syllable to the right, as in (50-a) and (50-b). If that is always the case, then this process is formally identical to the rightward shift of /r/ in Spanish, which was already determined to be SL. In (50-c), the pharyngeal crosses two morpheme boundaries, but still only one syllable boundary if the loss of the root vowel results in resyllabification (i.e., pwən.cʕá.təlx). Indeed Bessell (1998) suggests that the domain of this movement may be the foot.

The crucial (missing) data point would be one in which the pharyngeal moves across multiple syllable boundaries. In that case, the process could be derived with post-pivot copying of the root pharyngeal into the onset position of a following stressed syllable (assuming stress is marked), followed by pre-pivot deletion of the original. The functions that model these copy and deletion processes would not be SL, but they

would be left and right subsequential, respectively. Alternatively, since the pharyngeal movement is clearly related to stress, one could posit that stress in this language is signaled with a pharyngeal quality on the vowel. Under that analysis, there would be no movement at all, as the loss of the root pharyngeal would be simply a reflex of the stress shift to the suffix. Indeed, the movement does not occur unless the stress is shifted away from the root.[19]

Whether or not the La Huerta, Shuswap, and Colville displacement processes are SL is of particular interest, since if not they are the only cases found to date of synchronic non-SL movement (both metathesis and displacement). Under the assumption that only synchronic processes have to be learned, the absence of non-SL synchronic metathesis or displacement lends support to the proposed learning mechanism for SL processes that will be presented in Chapter 6.

4.5 Summary

This chapter has shown that local and long-distance metathesis processes are Strictly Local. In addition, a review of displacement processes suggests that synchronic displacement is also SL, though further investigation is needed of the La Huerta, Shuswap, and Colville processes. Non-SL displacement characterized by unbounded movement is also attested, but appears to be limited to the diachronic domain. Table 4.1 summarizes the types of displacement discussed in this chapter.

These findings raise an interesting question: if unbounded movement is possible, why is there still a prevalence for SL diachronic movement? If it turns out that the few non-SL processes, both diachronic and synchronic, can be explained as the result of (or due to interaction with) other factors, then a strong restriction for phonological movement to be local can be established and serve as a further distinction between the kinds of patterns observed in phonology versus syntax.

[19] Blevins and Garrett (1998) also suggest a non-movement account of Colville pharyngeal displacement as vowel-retracting harmony.

Table 4.1: Summary of Displacement Processes

	Diachronic	Synchronic
SL	Spanish Gascon Bagnéres-de-Luchon Marathi Romani	Greek aspiration
Non-SL	Italian Greek liquids	La Huerta? Shuswap? Colville?

As a final note, this chapter has presented both computational and empirical evidence to support the view that metathesis is not an exceptional phonological process. The empirical evidence suggests that both metathesis and displacement patterns can be decomposed into copying, a particular type of insertion, and deletion. Such an analysis was proposed as an alternative to a transposition analysis, in which the reordering of segments is a single operation. The computational evidence is that these component copy and deletion processes are SL just like the more common insertion and deletion processes discussed in the previous chapter. It turns out, however, that the SL status of metathesis does not depend on this decomposition. Rotuman metathesis, for example, can also be modeled with a SL-4 substitution process. The SL FST in Figure 4.5 models the rule in (51).

(51) $V_1C_1 \rightarrow C_1V_1$ / V __#

Thus, analyzing local or long-distance metathesis as transposition instead of copying followed by deletion does not require any additional formal power, since the pattern is SL under both analyses. This is the kind of insight gained by directly modeling input-output mappings, since as was mentioned at the beginning of this chapter, accounting for metathesis in both SPE and OT required an increase in the formal complexity of the grammar.

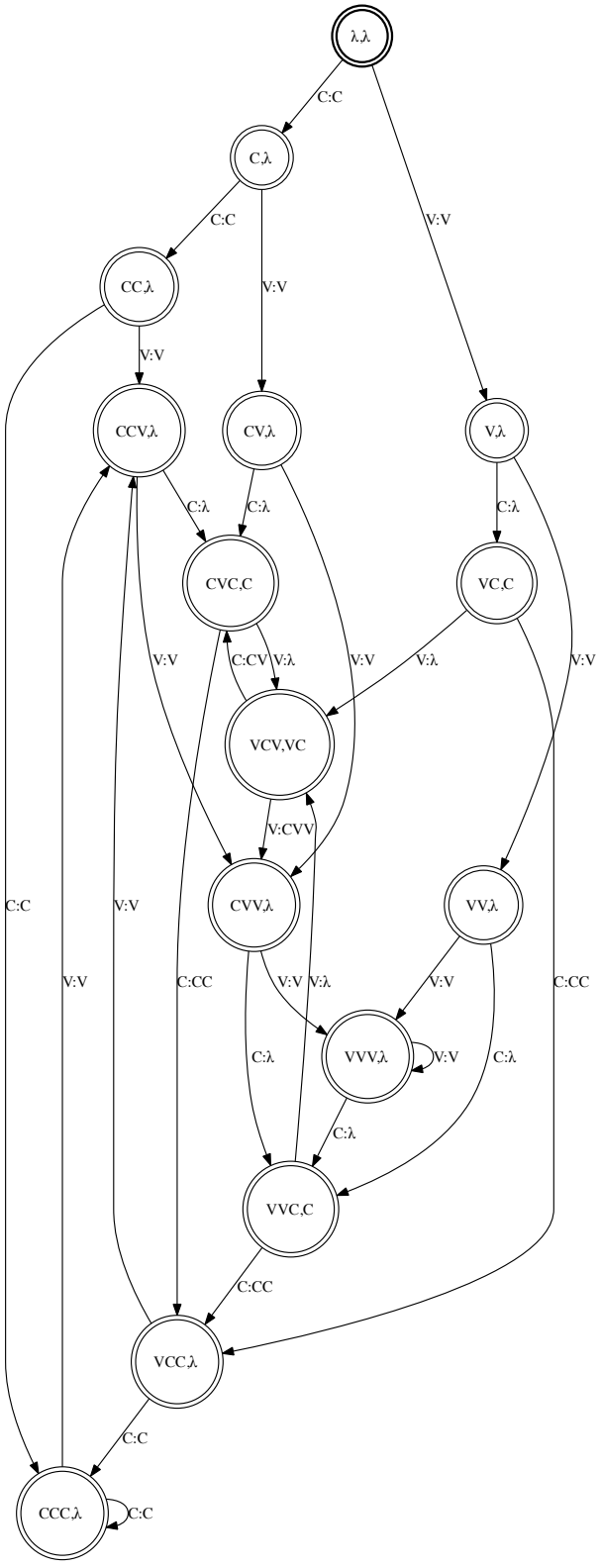


Figure 4.5: $\mathcal{T}_{SL}$ for Rotuman metathesis, $k = 4$, $\Sigma = \{\text{C, V}\}$

It would be interesting to also seek additional experimental evidence for the psychological reality of the decomposition of metathesis. Such evidence would also speak to a larger question of interest, which is whether or not individual processes have psychological reality independently of the larger grammar. Since the answer to that question touches on one of the key distinctions between the SPE and OT models of the phonological grammar, pursuing it could lead to strong implications beyond just the correct account of metathesis patterns.

The decomposition analysis does, however, unify metathesis and displacement in a clearer way than the transposition analysis. The intervening material between the original and final positions of a displaced segment may be unbounded, and a transposition rule would have to include this unbounded material in the target and structural change. For example, the Greek liquid displacement, as transposition, would have the rule in (52), where again x is a string of segments of unknown length.

(52) $\quad C\textit{x}Cr \rightarrow Cr\textit{x}C \;/\; \#\_\_$

Unlike the transposition rule in (51), the rule in (52) includes an unbounded amount of material in the target and structural change. As mentioned at the beginning of this chapter, the use of a multiple-segment target and structural change is what initially distinguished metathesis from other processes and inspired alternative analyses that avoided that expansion of the rule formalism. That in turn casts suspicion on an analysis requiring targets and structural changes of *unbounded* length. One possibility is that metathesis, as it has been defined in this chapter (i.e., both local and long-distance) is transposition, while displacement is copying and deletion. But viewing both metathesis and displacement as copy-delete has the advantage of unifying all cases of phonological movement.

The next chapter turns to the modeling of processes outside of phonology proper, in particular, the word formation processes of affixation and reduplication. It will be shown that the analysis of copying presented in this chapter extends to partial

reduplication cases, again provided there is a bound on the amount of material copied and the material that intervenes between the original and the copy.

Chapter 5

WORD FORMATION PROCESSES

The previous two chapters demonstrated the range of phonological processes that can be classified as Strictly Local. This chapter turns to word formation processes and demonstrates the extent to which they too are SL. In particular, the treatment of metathesis as involving a component of copying suggests a natural connection to reduplication patterns. Also like metathesis, reduplication has been accounted for in a variety of ways that generally aim to simplify a seemingly complicated process. Such analyses include, among others, attaching a reduplicative template to the CV-skeleton (Marantz, 1982), loops over a representation of a word's precedence relations (Raimy, 1999, 2000), OT constraints requiring correspondence between the base and the reduplicant (McCarthy and Prince, 1995), and an OT constraint REDUP that generally favors reduplication structures (Zuraw, 2002).

Given that morphological processes involve larger structural units than phonological processes, it would not be surprising if they likewise involved greater computational complexity. But this chapter will show that both general affixation and many types of partial reduplication are also SL. It is not the case, however, that all reduplication is SL, nor even subsequential, and these exceptions will also be discussed.

5.1 Affixation

Both prefixation and suffixation can be handled straightforwardly by SL FSTs. Figure 5.1 shows $\mathcal{T}_{SL}$ for English *un−* prefixation (e.g., *undo, untie, uncover*, etc.). This time the symbol ? represents any segment in the inventory. As soon as the FST reads in a single symbol (i.e., as soon as it rules out the empty string input), it prefixes

un− to that symbol and then moves to a state where all remaining input is outputted unchanged.

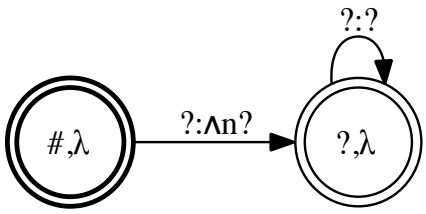


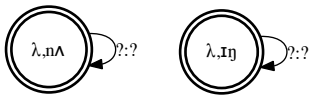
Figure 5.1: $\mathcal{T}_{SL}$ for English *un*-prefixation, $k = 2$, $\Sigma = \{?\}$

Likewise, suffixation is a simple matter of outputting the entire input unchanged and then attaching the suffix via the final output function. The FST in Figure 5.2 models English −*ing* suffixation (e.g., *singing, reading, watching*, etc.).[1]

As for infixation, it too is SL provided there is a bound on the amount of material between the edge of the string and the infix's position. As an example, consider *um*-infixation in Tagalog, which derives the finite form of a verb (Orgun and Sprouse, 1999; Klein, 2005):

---

[1] These FSTs for prefixation and suffixation are constructed with the assumption that the prefix/suffix should not be appended to the empty string. Without that assumption, these operations could be achieved with the single state machines shown below.



Note that the prefixation FST (on the left) is right subsequential. To prefix *un*− to an input string like 'do' (/du/), that string is first reversed (*ud*), then outputted unchanged (by the ?:? loop), and then the reversed prefix (*n*ʌ) is appended to the end of the output. This entire output is then reversed: $(ud + n\text{ʌ})^r = \text{ʌ}ndu$, 'undo'.

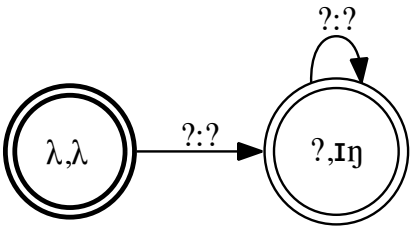


Figure 5.2: $\mathcal{T}_{SL}$ for English *ing*-suffixation, $k = 2$, $\Sigma = \{?\}$

(1) Tagalog

sulat	'write'	sumlat	'to write'
gradwet	'graduate'	grumadwet	'to graduate'
mahal	'expensive'	*mumahal	

Figure 5.3 shows $\mathcal{T}_{SL}$ for this infixation process. The alphabet for the FST is $\Sigma =$ {C, V, M}, where V is a vowel, M is a labial sonorant, and C is any other consonant.

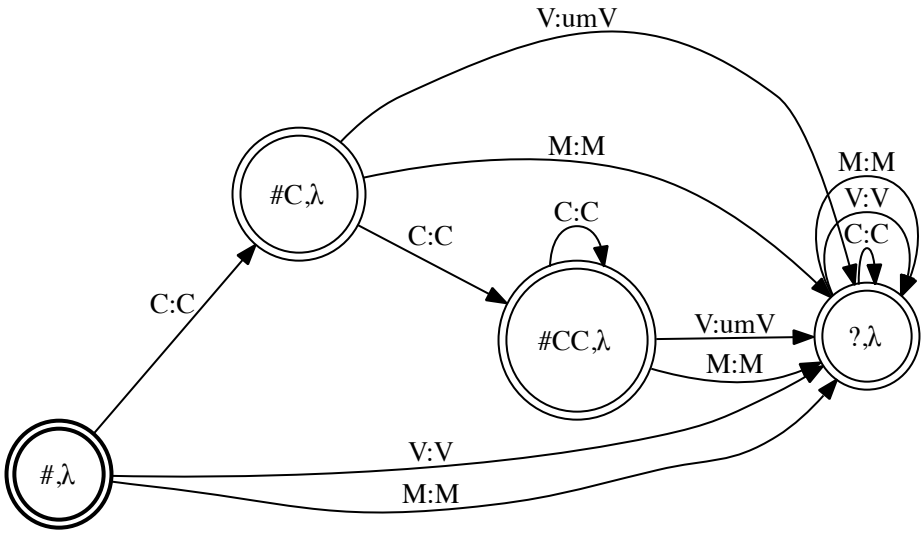


Figure 5.3: $\mathcal{T}_{SL}$ for Tagalog *um*-infixation, $k = 4$, $\Sigma = \{C, V, M\}$

The FST inserts the infix in the word-initial context #C__V, as in *sumlat*, or #CC__V, as in *grumadwet*.[2] The process is not permitted after an initial vowel or labial sonorant, as shown with **mumahal*. For ease of reading, in the FST the state ? represents the point at which it has been determined whether or not the infixation is licit. After this point, all input is outputted unchanged, as indicated by the loops at this state.

Despite the fact that this Tagalog infixation process has contextual variability and blocking, it is still SL because the position of the infix is a fixed distance from the word boundary (in this case, either 1 or 2 segments). An infixation process without such a fixed position would be one, for example, in which the infix is place in the exact center of the string. The hypothesis that all affixation is SL predicts the non-existence of such a process, a prediction shared by OT treatments of affixation that restrict the position of the affix using edge alignment constraints (McCarthy and Prince, 1996).

Since, as this section has shown, general affixation is a Strictly Local process, the SL status of reduplication processes will hinge on the nature of the copying. The next section presents the kinds of copying involved in reduplication and determines which can be classified as SL.

5.2 Reduplication

Reduplication has traditionally been divided into two categories, full (or total) and partial. In partial reduplication, some portion (the reduplicant) is copied from the word (the base). In full reduplication, the entire word is copied. These two categories will be discussed in turn.

5.2.1 Partial reduplication

In partial reduplication patterns, some portion of a word is copied and then affixed to it. For an example of reduplicative prefixation, we can again look to Tagalog

[2] According to Orgun and Sprouse (1999) and Klein (2005), in the case of #CC__V, the infix can also variably appear in #C__CV. Modeling free variation is beyond the scope of this thesis, but see Chapter 7 for comments on this future direction.

(Blake, 1917):

(2) Tagalog

súlat ‘write’ susúlat ‘will write’

In this example, the future tense of a verb is derived by copying the first CV of the base and then prefixing it. We can schematize this process as in (3), where the dash indicates the morpheme boundary between the reduplicant affix and the base and x refers to the remainder of the base after the initial CV.

(3) C_1V_1-C_1V_1x

A rule for this reduplication process is shown in (4)[3], and a SL FST that models it is shown in Figure 5.4.

(4) $\emptyset \rightarrow C_1V_1 \ / \ \#\_\_C_1V_1$

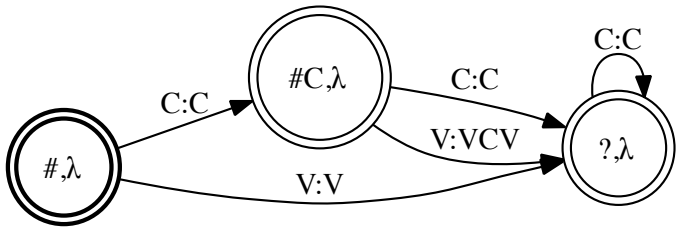


Figure 5.4: Schematized $\mathcal{T}_{SL}$ for Tagalog partial reduplication, $k = 3$, $\Sigma = \{C, V\}$

Note that Figure 5.4 is just a schema for the complete FST for the Tagalog reduplication pattern. The complete FST contains a branch like Figure 5.4 for each possible CV combination. Despite the larger size of this complete FST, however, it is still SL. And

[3] The rules presented in this chapter are meant to highlight the connection between SL copying in metathesis and reduplication, but no theoretical claim is being made here that rules are the correct grammatical representation of reduplication.

as in Figure 5.3 the ? state in Figure 5.4 represents the point at which the prefixation has taken place and so the rest of the input is outputted unchanged.

To get a sense of what the complete FST looks like, Figure 5.5 presents the FST for Tagalog, assuming an alphabet of only {s, a, u}. The crucial transitions are from state $\#s$: if the next input symbol is a vowel (either 'a' or 'u'), the environment for partial reduplication (i.e., CV-initial words) has been fulfilled. The outgoing 'a' and 'u' transitions from this state include in the output the copied *sa* and *su*, respectively.

The partial reduplication schema in (3) is an example of what has been called local reduplication, since the reduplicant is affixed adjacent to the portion of the base it is copied from. Another category that meets this condition is schematized in (5), in which the reduplicant is a suffix copied from the end of the base. An example from Marshallese is shown in (6) (Byrd, 1993).

(5) $x\mathrm{C}_1\mathrm{V}_1\mathrm{C}_2\text{-}\mathrm{C}_1\mathrm{V}_1\mathrm{C}_2$

(6) Marshallese

ebbok	'to make full'	ebbok-bok	'puffy'
lɤŋɔŋ	'fear'	lɤŋɔŋ-ŋɔŋ	'very afraid'

This process copies the final CVC of the base and then suffixes that copy to the base. A rule for this copying is shown in (7).

(7) $\emptyset \rightarrow \mathrm{C}_1\mathrm{V}_1\mathrm{C}_2 \;/\; \mathrm{C}_1\mathrm{V}_1\mathrm{C}_2\_\!\_\#$

And a SL FST that models the process is shown in Figure 5.6. As with general suffixation (Figure 5.2), the reduplicated suffix is appended via the final output function in Figure 5.6, but only to strings that end in CVC (i.e., strings that end in state CVC). And as with the Tagalog reduplication, the pattern in Marshallese is considered local because the reduplicant is affixed adjacent to the portion of the base it was copied from.

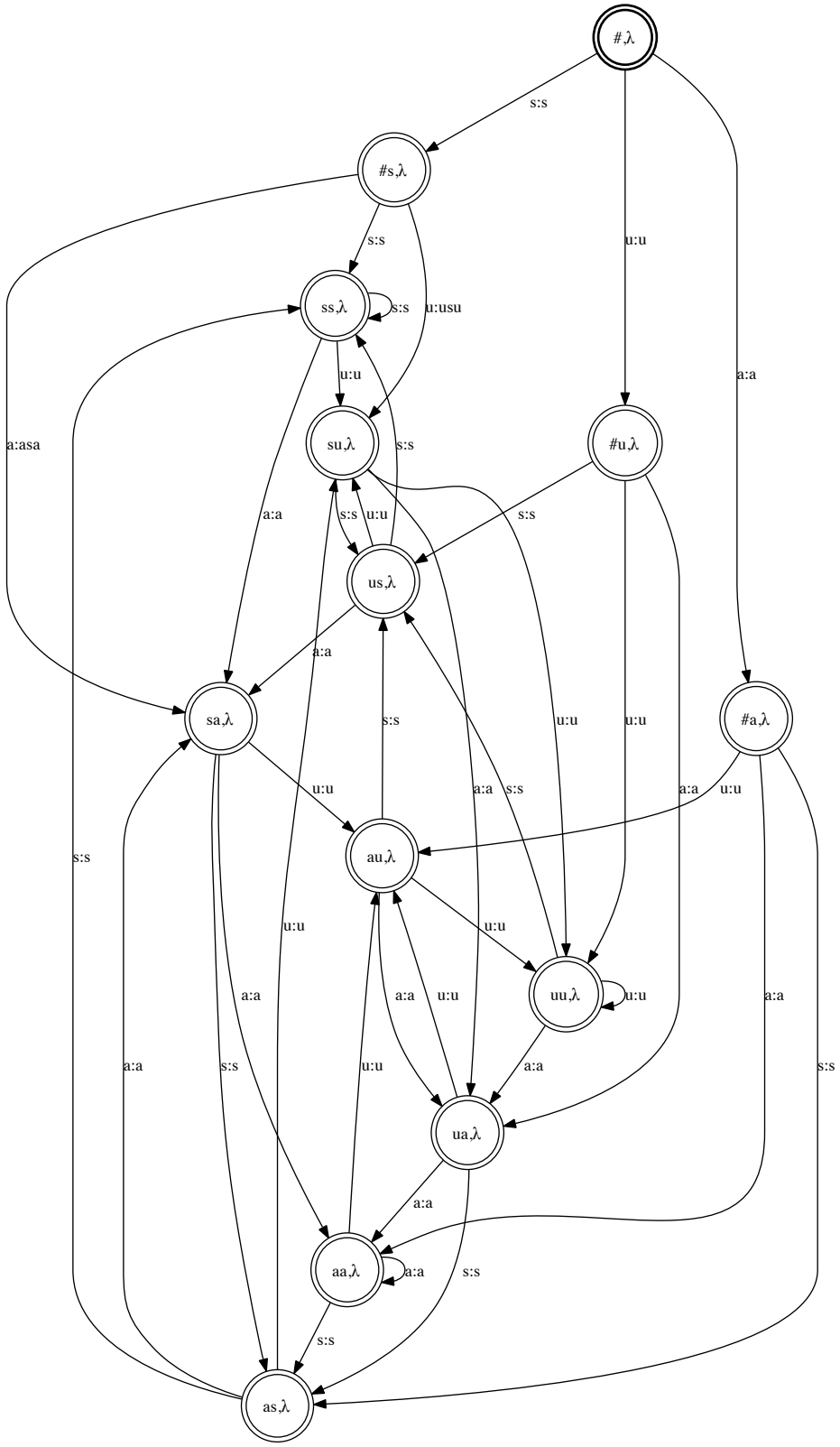


Figure 5.5: $\mathcal{T}_{SL}$ for Tagalog partial reduplication, k =3, Σ = {s, a, u}

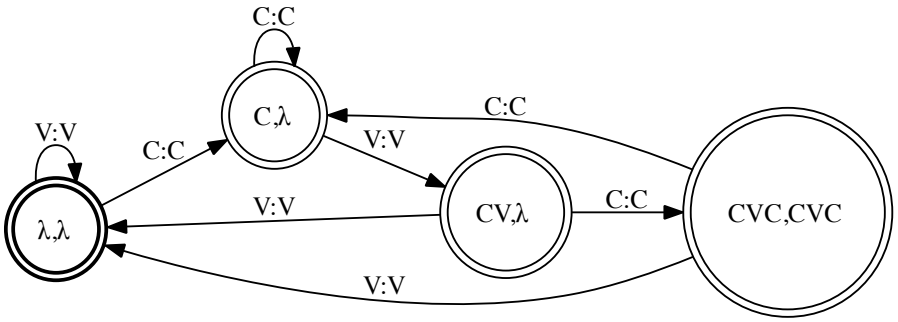


Figure 5.6: Minimized $\mathcal{T}_{SL}$ for Marshallese partial reduplication, $k = 4$, $\Sigma = \{\text{C, V}\}$

Lastly, a reduplicant can also serve as an infix, such as in the example in (8) from Pima (Riggle, 2006a).

(8) Pima

sipuk ‘cardinal’ sispuk ‘cardinals’

(9) $\text{C}_1\text{V-C}_1\text{-}x$

As shown in the schema in (9), this process copies the initial consonant and infixes it after the initial CV of the base. A rule for this process is shown in (10), and the SL FST is shown in 5.7.

(10) $\emptyset \rightarrow \text{C}_1 \ / \ \#\text{C}_1\text{V}\_\_$

Figure 5.7 maps all CV-initial inputs to outputs with the initial C inserted after the first vowel. Like the Tagalog infixation example in the previous section, in Pima reduplicative infixation the distance between the infix and the edge of the word (which is also the material it was copied from) is bounded, so this process can be modeled with a SL function.

The examples of local reduplicative prefixation and suffixation contrast with their non-local counterparts, in which the reduplicant is copied from one end of the

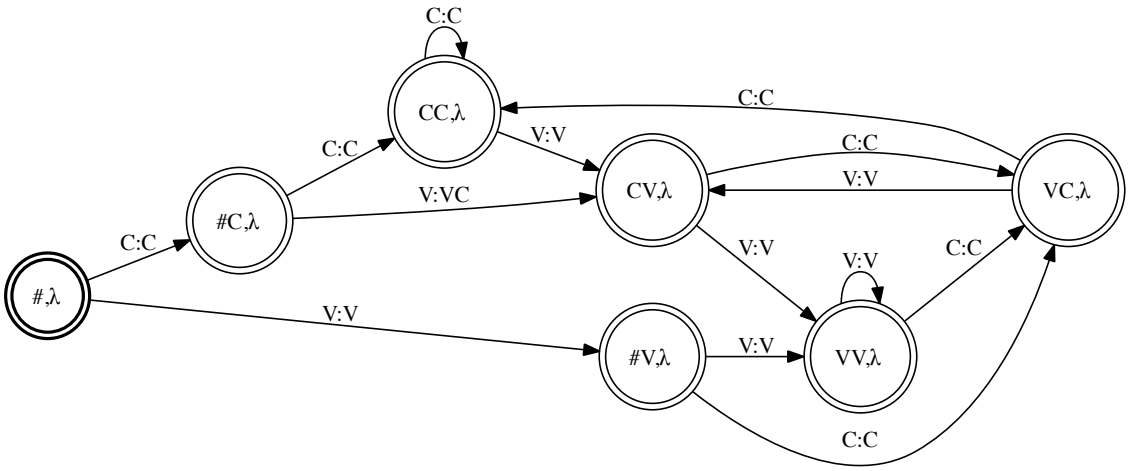


Figure 5.7: $\mathcal{T}_{SL}$ for Pima partial reduplication, $k = 3$, $\Sigma = \{\text{C, V}\}$

string but affixed to the opposite end (Riggle, 2003). An example of non-local suffixation from Koryak (Bogoras, 1969; Kenstowicz, 1979; Krause, 1980) is shown in (11) and schematized in (12).

(11) Koryak

mɪtqa ‘oil’ mɪtqa-mɪt ‘oil’ (absolute)

(12) $\text{C}_1\text{V}_1\text{C}_2x\text{-C}_1\text{V}_1\text{C}_2$

And a rule for this copying process is shown in (13).

(13) $\emptyset \rightarrow \text{C}_1\text{V}_1\text{C}_2 \ / \ \text{C}_1\text{V}_1\text{C}_2x\_\#$

What makes this process non-local is the x string that intervenes between the position of the reduplicant and the portion of the base that it copies. Since the length of this x is unbounded, we cannot identify a k-value for this process. In other words, there is no k such that a FST that only keeps track of the previous $k-1$ symbols can distinguish strings that begin with CVC from those that do not.

Though Koryak partial reduplication is not SL, it is left subsequential. The SFST in Figure 5.8 models the rule in (13).

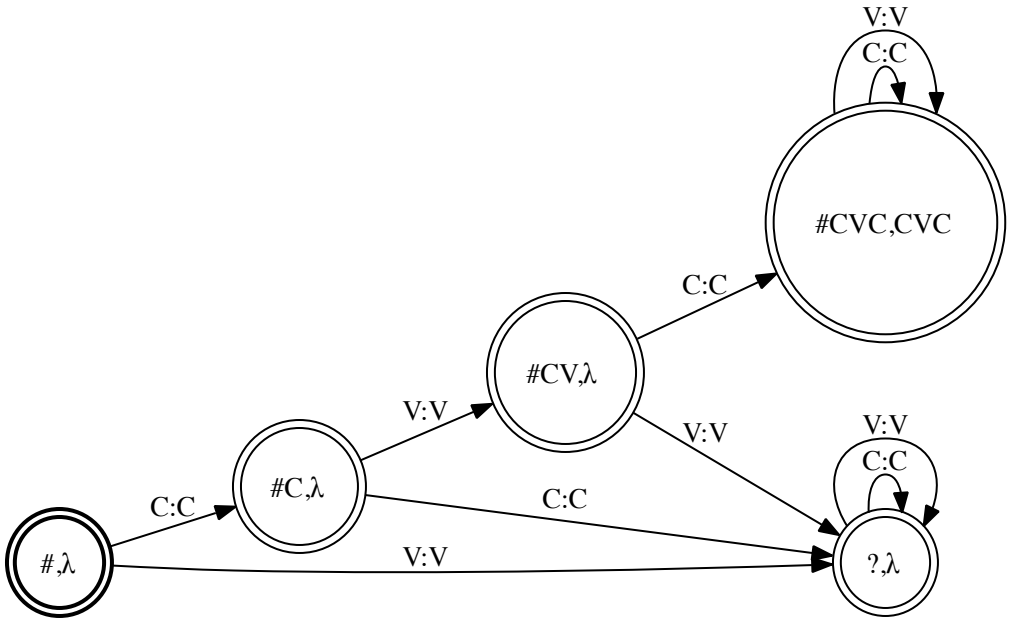


Figure 5.8: SFST for Koryak partial reduplication, $\Sigma = \{\text{C, V}\}$

Within the first three input symbols, this SFST distinguishes strings that begin with CVC from those that do not. From that point on these two types of strings are handled by different states (labeled #CVC and ?, respectively), and those in the former category have a CVC suffix appended to their output strings via the final output function. One might question how this FST differs from the one in Figure 5.4 above, which also has a ? state and is Strictly Local. The difference is that in Figure 5.4, the ? state represents all $k-1$-factors except the ones already included in the FST. This means the FST will not encounter the same $k-1$-factor in two different states. In Figure 5.8, however, there is no k such that the FST could *not* encounter the same $k-1$-factor in states #CVC and ?, since both states have loops for C and V. This means it keeps track of information other than just the previous $k-1$ symbols: it distinguishes whether the

input string began with CVC (by being in state #CVC) or not (by being in state ?). Thus Figure 5.4 is a SL FST, but Figure 5.8 is not.

The Koryak example shows that non-local reduplicative suffixation is not SL, but it is still subregular. More will be said about non-SL but still subsequential processes in Chapter 7. Some researchers have addressed the seemingly exceptional nature of non-local copying like that in Koryak by classifying it as something other than reduplication (Nelson, 2005), under the assumption that reduplication is by default local (i.e., the 'edge-in association' of Marantz (1982)). However, the question of whether non-local reduplication is truly reduplication is tangential to the issue of whether or not it is SL. As an attested process, non-local copying sheds light on the extent to which morphological processes are Strictly Local.

It is unclear whether cases of non-local reduplicative prefixation exist.[4] This would be a process like the one schematized in (14), in which a portion of the string starting from the end is copied and then used as a prefix.

(14) $\mathrm{C_1V_1}$-$x\mathrm{C_1V_1}$

Such a process is again not SL, but still subsequential. More precisely, it is right subsequential because (as discussed in the previous chapter) it is pre-pivot copying. Given an input string of the form $x\mathrm{C_1V_1}$, reversing this string gives $\mathrm{V_1C_1}x^r$. Consider a SFST that models the rule in (15).

(15) $\emptyset \rightarrow \mathrm{V_1C_1}$ / $\mathrm{V_1C_1}x$__#

We know such a SFST exists because this rule differs minimally from the Koryak example in (13). This SFST would map $\mathrm{V_1C_1}x^r$ to $\mathrm{V_1C_1}x^r\mathrm{V_1C_1}$. The reverse of this output is $\mathrm{C_1V_1}x\mathrm{C_1V_1}$, the expected output for non-local prefixation.

It is also unclear whether non-SL reduplicative infixation exists. This would (presumably) be a process in which some portion of the base is copied and then infixed

[4] It is not part of the factorial typology of Riggle (2003)'s OT analysis.

at an arbitrary position in the string. Here 'arbitrary' means there is no bound on the amount of material between the word-boundary and the position of the infix. Such a process would certainly not be SL, and in fact it would not even be subsequential. It was possible to model non-local prefixation and suffixation with SFSTs because the position of these affixes is fixed with respect to the word-boundary. The lack of this fixed position would force the FST to keep track of an unbounded number of input symbols, which it cannot do (i.e., it could not then be *finite* state).[5] The existence of such a pattern would pose a challenge for the subregular status of morpho-phonology, and therefore an unequivocal example would be both surprising and interesting.

Table 5.1 summarizes the findings for partial reduplication patterns. The next section turns to full reduplication.

Table 5.1: Summary of Computational Properties of Partial Reduplication Patterns

	SL	Left subsequential	Right subsequential
Local prefixation	✓	✓	✓
Local suffixation	✓	✓	✓
Local infixation	✓	✓	✓
Non-local prefixation			✓
Non-local suffixation		✓	

5.2.2 Full reduplication

In addition to partial reduplication, many languages exhibit full (or total) reduplication, in which the entire string appears twice. In a classic example from Indonesian, the plural of a noun is derived from the singular via full reduplication (Sneddon, 1996):

(16) Indonesian

buku 'book' buku-buku 'books'

[5] See Jardine (2013) and Heinz and Lai (2013) for a similar explanation and formal proofs for the non-subsequentiality of tonal plateauing and (unattested) sour grapes vowel harmony, respectively.

Full reduplication is not Strictly Local. Nor for that matter is it subsequential, or even regular, because no finite state machine can model the infinite mapping of this process. It is true that if every person's lexicon is finite, there will be a longest word, and setting k to the length of that longest word plus one would permit a SL FST to model full reduplication. Each word would correspond to a path in the FST and each state would append its own $k-1$ factor to the end of the output. For example, consider a language of strings of a's in which the longest word is of length 4. The SL FST in Figure 5.9 models full reduplication in this language.

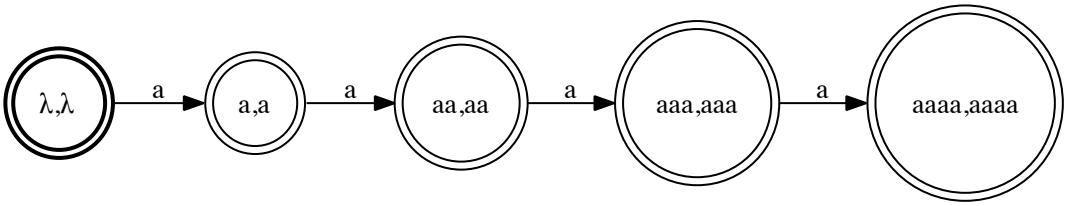


Figure 5.9: $\mathcal{T}_{SL}$ for full reduplication in $\mathcal{L} = a^{\leq 4}$

This FST will correctly map $a \mapsto a\text{-}a$, $aa \mapsto aa\text{-}aa$, $aaa \mapsto aaa\text{-}aaa$, and $aaaa \mapsto aaaa\text{-}aaaa$. If this language added a word of length 5, the k-value is increased to 6 and the FST could be extended to accommodate full reduplication of this word as well. And so on, up to whatever the longest word ends up being. But this is precisely the difference between modeling just the input that has been seen so far and modeling the infinite mapping the data represents. Speakers of a language with full reduplication can do more than just produce the reduplicated form for any word in their lexicons. They can produce the reduplicated form for any *possible* word, including longer words that are not yet in their lexicons. It is this generalization that a FST like Figure 5.9 misses, and that no FST can model. This important distinction between modeling finite data and infinite mappings is the basis for the approach to learning SL functions presented in the next chapter.

Thus full reduplication is a counter-example to the hypothesis that morphology is also subregular. Instead, full reduplication is context-sensitive. Computationally, then, it has more in common with syntax than phonology (Chomsky, 1956, 1959; Kobele, 2006). That fact could be taken as evidence that full and partial reduplication are not really two subtypes of a single process. Instead, full reduplication can be considered as a particular case of syntactic copying, while morphological reduplication is always partial. More generally, the computational classifications presented in this chapter highlight which processes merit further attention and what typological gaps warrant scrutiny. We have seen that some morphological processes (e.g., general affixation, local partial reduplication) are Strictly Local, while others (e.g., non-local partial reduplication) are not SL but still subsequential, or even non-regular (e.g., full reduplication).[6] While some treatments aim to unify the various forms reduplication can take (e.g., Raimy (2000)), perhaps we should also investigate further how they differ.

With this understanding of the range of phonological and morphological processes that can be modeled with SL functions, the next chapter will present an algorithm that learns such processes by using their defining property of strict locality as an inductive principle.

[6] The computational properties of templatic morphology remain to be investigated.

Chapter 6

LEARNING STRICTLY LOCAL FUNCTIONS

One motivation for restricting the range of possible phonological patterns is to investigate whether the nature of the restrictions can explain how humans learn phonology. Restrictions in the form of computational properties serve to structure the hypothesis space of a learner in such a way that logically possible mappings that are consistent with the data but do not have the computational property in question are not even considered. This chapter will demonstrate that strict locality can delimit a learner's hypothesis space in just this way.

As already discussed in Chapter 2, the regular relations are not believed to be learnable within the Gold paradigm (Gold, 1967), so the finding that phonological mappings are regular (Johnson, 1984; Koskenniemi, 1983; Kaplan and Kay, 1994) does not provide a clear approach to the learning problem. In contrast, the subsequential class of subregular functions *is* identifiable in the limit from positive data by the algorithm known as OSTIA (Onward Subsequential Transducer Induction Algorithm) (Oncina et al., 1993). OSTIA generalizes from a finite set of input-output string pairs and converges on a SFST that describes the mapping they represent. Since all of the phonological mappings that have been shown to be Strictly Local are by definition also subsequential, it follows that OSTIA can learn them. And by extension, perhaps humans generalize from their language data in the same manner in which OSTIA does (i.e., by assuming their target is a subsequential mapping).

However, when tested on a phonological example with natural language data, OSTIA did not in fact converge on the correct target function (Gildea and Jurafsky, 1995, 1996). In addition, there exist subsequential functions that do not resemble the kinds of mappings we find and expect to find in natural language phonology. Together

these facts suggest that the phonological learner is assuming more than just subsequentiality when it converges on a phonological mapping. This chapter proposes that the learner instead assumes that its target is a Strictly Local function. As a demonstration, a learning algorithm is presented that identifies the ISL functions in the limit by using strict locality as an inductive principle. A comparable algorithm for learning OSL functions is being left for future work, but the chapter will end with some discussion of how such an algorithm can be developed.

The algorithm that will be presented is called the Input Strictly Local Function Learning Algorithm (ISLFLA). As it is based on OSTIA, the next section will first present in detail how OSTIA generalizes from positive data to a subsequential function. It will be shown, however, that the restriction on the ISLFLA's hypothesis space (i.e., the class of ISL functions) allows it to converge on its target faster and with less data than OSTIA.

6.1 The Onward Subsequential Transducer Induction Algorithm (OSTIA)

OSTIA is essentially a two stage algorithm: 1) build a prefix tree and 2) merge states. Prefix trees will be discussed in detail in this and the next section. State merging is widely used in grammatical inference; for other examples see Angluin (1982), Heinz (2009), and de la Higuera (2010).

As mentioned above, the input data to OSTIA is a set of input-output string pairs that sample the target function. Consider the example subsequential function in (1), from Oncina et al. (1993).

(1)

$$t(a^n) = \begin{cases} \lambda & n = 0 \\ b^{n+1} & n\ odd \\ b^n c & n\ even \end{cases}$$

The domain of this function is the set of strings of a's of any length. If the length of the input string is even, the output is an equal number of b's plus a 'c'. If the length of

the input string is odd, the output is a string of b's that is one longer than the input. A data set T that samples this function is shown in (2).

(2) $T = \{(\lambda, \lambda), (a, bb), (aa, bbc), (aaa, bbbb), (aaaa, bbbbc)\}$

The first step of the algorithm is to create a finite state representation of this data set. This representation is a particular type of SFST called an onward tree subsequential transducer (OTST). The OTST for the data set in (2) is shown in Figure 6.1. Notice that when the tree is constructed each input string is augmented with the end-of-word marker #, which leads each state to a final state.

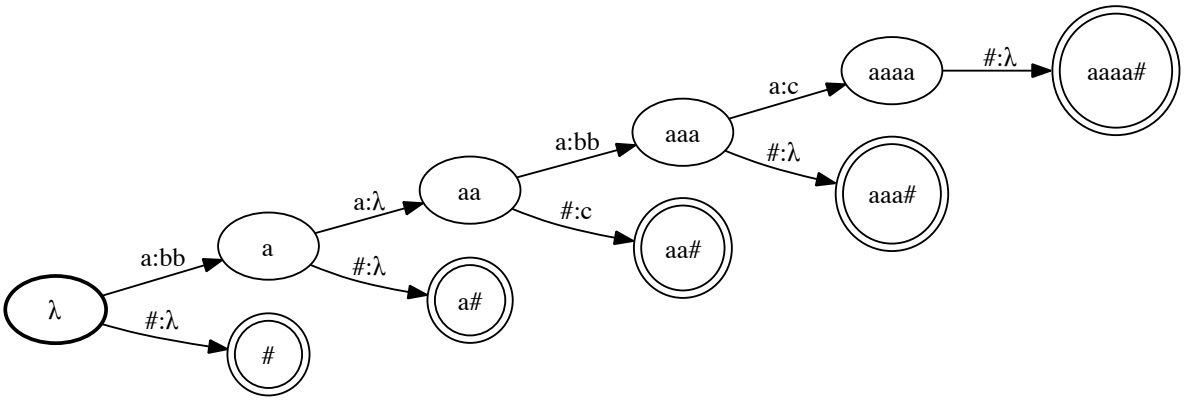


Figure 6.1: OTST for the example data set in (2), $\Sigma = \{a\}$, $\Gamma = \{b, c\}$

This representation is called a tree because it branches out from the initial state (the 'root') and, more importantly, it contains no loops. What this means is that the FST represents *only* the data set from which it was built. It does not generalize beyond that set and therefore cannot produce an output for any input aside from those in the data. Lastly, the *onward* designation of the OTST indicates that it produces output on the transitions as soon as it has enough information to do so. For example, the 'a' transition from the root produces the output *bb*, because (based on the data set) all input strings that begin with *a* are mapped to output strings that begin with *bb*.

Formally, onwardness is achieved when the longest common prefix of all outputs of all transitions leaving a state is λ.

Since the target subsequential function can produce an output for any input in its domain, it is necessary to generalize beyond the OTST. OSTIA achieves this generalization via state merging, a process that combines states while preserving transitions. State merging can be thought of as a means of reducing information. Given a characteristic sample of the target transduction, the OTST created from that sample contains the maximum amount of information about the transduction. Merging states is a way of discarding all but the necessary information to get the transduction right.

For example, in Figure 6.1, states a and aaa are both reached on a transition labeled $a : bb$. These states will be merged in the course of OSTIA's run, so that in the result (see Figure 6.2 below) only state a remains. This is possible because according to the target function, it does not matter whether the transducer has seen one 'a' or three; what *does* matter is whether it has seen an even or odd number of a's.

State merging takes place in both an inner and an outer loop of the algorithm. The outer loop considers all possible pairs of states for merging. If the merging introduces non-determinism, the inner loop attempts to remove it by an operation called *pushback* followed by additional state merges. The *pushback* operation will be discussed further in §6.2. Removing non-determinism is necessary to ensure that the final FST the algorithm converges on (as well as all intermediate FSTs) is still subsequential. Therefore if the inner loop determines that it cannot remove the non-determinism, the original (outer loop) merge is rejected, and the algorithm moves on to the next pair of states. The final output of OSTIA for the data in (2) is shown in Figure 6.2. The reader can verify that this SFST indeed describes the function in (1).

It is proven in Oncina et al. (1993) that this algorithm can identify any subsequential function in the limit from positive data. However, as stated above, a study by Gildea and Jurafsky (1995, 1996) found that given natural language data OSTIA was unable to learn a phonological mapping. The next section discusses this result.

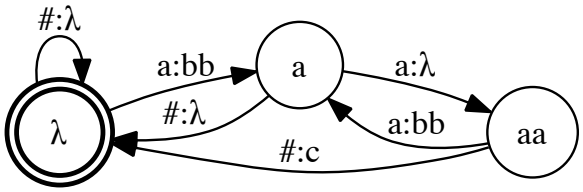


Figure 6.2: Final output of OSTIA for the data set in (2)

6.1.1 Modifying OSTIA to learning phonological mappings

Gildea and Jurafsky (1995, 1996) (hereafter GJ) tested OSTIA using a corpus of natural language data to see if it would learn the English flapping rule in (3).

(3) $t \rightarrow ɾ$ / V́ (ɪ) __V

Flapping is a substitution process by which a /t/ surfaces as a flap when it follows a stressed vowel and an optional /ɪ/ and precedes an unstressed vowel. GJ found that the output of OSTIA when given natural language data did not correctly describe the mapping of this rule. Why? OSTIA is guaranteed to learn in the limit any subsequential function, but to succeed it requires a characteristic sample of the target function. In the case of flapping this characteristic sample is not present in a natural language dictionary. More specifically, the mapping that describes *only* the flapping process is a *total* function. This means that every state in the flapping SFST has a transition for every symbol in the alphabet. But a natural language corpus does not contain the necessary data for OSTIA to generalize a total function, because the corpus reflects more than just the flapping process. It also reflects the phonotactic constraints that dictate the language's valid surface forms. Thus OSTIA will not find evidence for every transition from every state, because certain sequences of alphabet symbols are prohibited. As a result, it does not converge on the correct transducer that models just the flapping process.

GJ's solution was to equip OSTIA with three learning heuristics that are largely

assumed in phonological theory. These include 1) faithfulness (corresponding underlying and surface segments tend to be similar), 2) community (segments within a natural class behave alike), and 3) context (phonological rules refer to variables in their contexts). Their modified version of OSTIA successfully learned the flapping rule, as well as several others.

Their study reveals the value of these particular learning assumptions and of prior knowledge in general. It was also successful in terms of getting OSTIA to succeed with natural language data. Another desirable objective, however, is to obtain theoretical results. For instance, it is also important to show that a learner succeeds not just on a set of test cases, but on the entire class of functions those test cases are drawn from. Such a proof would remove any concern that the learner succeeded based on some unknown property of the particular examples on which it was tested. OSTIA itself can provably learn the class of subsequential functions, but no such proof is provided for GJ's modified OSTIA. In particular, their faithfulness bias is incorporated with an alignment operation over the input-output data pairs that alters the onwardness property of the initial tree transducer. But the onwardness property plays a crucial role in Oncina et al. (1993)'s proof of OSTIA's correctness.

This chapter proposes a different approach to learning phonological mappings. In particular, the next section introduces the ISLFLA, a learning algorithm that provably learns the class of Input Strictly Local functions by using strict locality as an inductive principle to generalize from positive data. The learner's use of the strict locality property is similar to the context bias employed by GJ's learner, but the structure of the ISL class also allows for a theoretical result of identification in the limit (presented in §6.8). The range of processes that have been shown to be ISL and the success of this learning strategy together suggest that the human phonological learner generalizes in the same manner as the ISLFLA when learning local processes.

6.2 The ISL Function Learning Algorithm (ISLFLA)

Like OSTIA, the ISLFLA first builds an onward prefix tree representation of the data set and then merges states to converge on a FST that describes the target function. The two algorithms differ, however, in the state merging criterion. OSTIA, as described above, merges any two states provided that any resulting non-determinism can be removed without altering the transduction represented by the data. The ISLFLA likewise insists on maintaining determinism (and therefore subsequentiality), but it also requires the two states that are merged to share a suffix of length $k-1$. To clarify what this means: each state is a string, and more specifically each state is a prefix of one of the input forms in the data set. This is true just by construction of the initial prefix tree. The suffixes of these strings represent the most recent symbols on the path leading into that state. For example, consider the branch of a tree for the input form *aabc*, shown in Figure 6.3. Note this figure depicts a prefix tree, not a tree transducer.

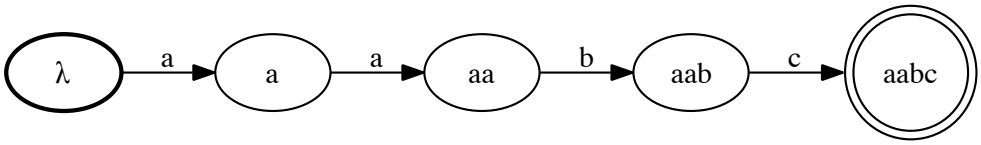


Figure 6.3: Branch of a prefix tree for input *aabc*

The suffixes of each state correspond to the incoming paths of varying lengths. For example, the suffixes of state *aabc* include λ, *c*, *bc*, *abc*, and *aabc*, which correspond to the incoming paths of length 0, 1, 2, 3, and 4, respectively. The ISLFLA will only merge states that were reached on the same path of length $k-1$, which is equivalent to having the same suffix of length $k-1$. So for a target function with a k-value of 2, states *a* and *aa* in Figure 6.3 could be merged, since both have the same suffix of length 1.

This state merging strategy is motivated by the fact that the states in the target $\mathcal{T}_{ISL}$ are $k-1$-factors, and no other distinctions among states need to be maintained

to capture a SL transduction. In other words, it is not necessary to maintain the entire path leading into a state, because the last $k-1$ symbols will suffice. It follows that this state merging criterion means the ISLFLA can only learn SL functions. And it learns specifically ISL (not OSL) functions because the states in the prefix tree represent prefixes of the *input* (not the output) strings.

A *non*-SL function could not be learned by this strategy, since the $k-1$-factors do not provide enough information for non-SL patterns. It could not, for example, learn the function in (1), because there is no k such that a SL-k FST could distinguish strings of even and odd lengths. Note that this tactic of merging states that share a suffix of length $k-1$ was previously applied to learning local phonotactics by Heinz (2007)[1] and local processes by Chandlee and Koirala (2014) and Chandlee and Jardine (2014). The difference between the latter two works and the current results will be discussed in §6.3.4 below.

6.2.1 The algorithm

This section describes in detail how the ISLFLA works. The pseudo-code is presented below. Like OSTIA, the data is submitted to the learner in the form of a prefix tree transducer (PTT), defined in Definition 6.

Definition 6 (Prefix Tree Transducer)**.** *Given $T = \{(w, w') \mid f(w) = w'\}$ for a function f, a prefix tree transducer for T is $PTT(T) = (Q, \Sigma, \Gamma, q_0, \delta)$, where Σ, Γ are the input and output alphabets, respectively, and*

- $Q = \bigcup\{\texttt{Pref}(w\#)|(w, w') \in T\}$
- $q_0 = \lambda$
- $\delta = \{(u, a, \lambda, ua) \mid u, ua \in Q\} \cup \{(w, \#, w', w\#) \mid (w, w') \in T\}$

The algorithm's first step is to submit this PTT to a function that makes it onward. Following Oncina et al. (1993), this onward function starts from the leaves

[1] Though locality was expressed in terms of n-grams rather than SL languages, the idea is the same.

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

Algorithm 1: Pseudo-code for the ISLFLA

of the PTT and iteratively advances the longest common prefix of the outputs of all outgoing transitions toward the root. Once this is done, the algorithm begins merging states, gradually collapsing the tree down to the SL FST.

Like OSTIA, the learner merges states in a pair of nested loops. The outer loop iterates through the state set of the tree, which is ordered lexicographically, and merges each state q with the state that corresponds to its suffix of length $k-1$. This is the first key difference between the ISLFLA and OSTIA - unlike OSTIA, the ISLFLA does not search the tree for a state to merge q with, because it knows exactly which state to merge it with (i.e., all q will be merged with $\texttt{Suff}^{k-1}(q)$).

Similar to OSTIA, however, the ISLFLA uses an inner loop to remove any non-determinism that results from the outer loop merge. Consider the situation depicted in Figure 6.4, in which the merge of q with $q' = \texttt{Suff}^{k-1}(q)$ has created non-determinism for input 'a'.

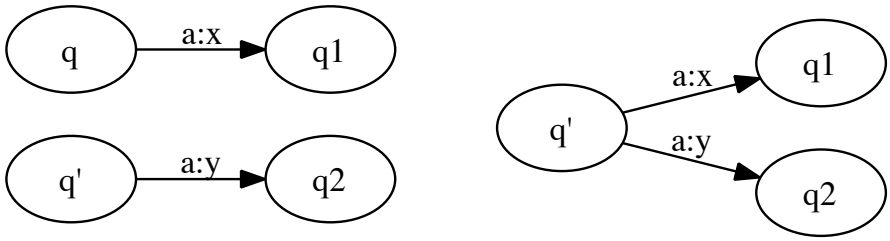


Figure 6.4: Non-determinism after merging q and q'

The learner detects the presence of such a situation by searching the transition function δ for two edges with the same origin state and the same input symbol (i.e., (q, a, x, q_1) and (q, a, y, q_2)). To remove this non-determinism, it first determines the longest common prefix of the two output strings x and y. Let $l = \texttt{lcp}(x, y)$ be this longest common prefix. This string l will be retained as the output from q' on 'a'; whatever is left of x and y after l is removed will be 'pushed back' toward the leaves of the tree. More specifically, $s = l^{-1}x$ and $t = l^{-1}y$ will be prefixed to the output strings of all transitions leaving states q_1 and q_2, respectively. This situation is depicted in Figures 6.5 and 6.6.

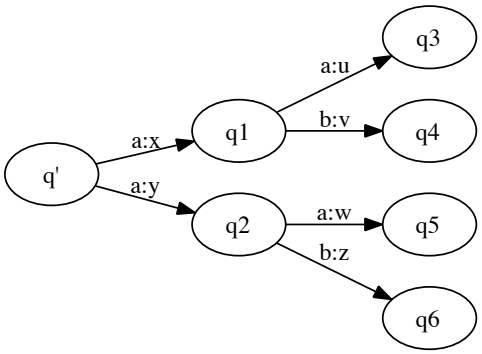


Figure 6.5: Before *pushback*

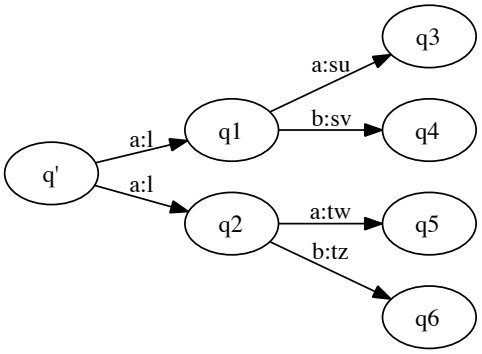


Figure 6.6: After *pushback*: $x = ls$ and $y = lt$

After *pushback*, states q_1 and q_2 can be merged to remove the non-determinism. It is easy to see in Figure 6.6 that this merge will create additional non-determinism, since both q_1 and q_2 have transitions on inputs 'a' and 'b'. The inner loop will continue to iterate until all non-determinism is removed, and then the outer loop will move on to the next state in the queue. When the outer loop reaches the end of the queue, the algorithm halts and outputs the FST.

Another important difference between the ISLFLA and OSTIA is the ability to reject state merges. As explained above, OSTIA will reject (and subsequently undo) some state merges if it finds that the resulting non-determinism cannot be removed with the inner loop procedure of *pushback* + *merge*. More precisely, it will reject any merge that leads to the situation depicted in Figure 6.7.

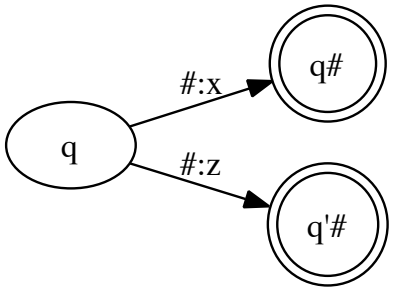


Figure 6.7: Irreparable non-determinism

Because the destination states of the two transitions leaving q in Figure 6.7 do not have any outgoing transitions (which is true of all states reached on #), the operation of pushing back the longest common prefix of x and z cannot apply. Thus there is no way to remove this non-determinism, and so OSTIA will reject whatever state merging led to this scenario. The ISLFLA, however, does not reject any merges. If it were to reject a merge, the possibility would arise that two states with the same suffix of length $k-1$ would remain distinct in the final FST the learner outputs. Such an FST would not satisfy the definition of being Strictly Local. Of course the situation in Figure 6.7 is problematic for the ISLFLA as well, since its final output must also

be deterministic. Therefore, when it enters the inner loop it first checks that the non-determinism it needs to remove is not of the form in Figure 6.7. If it is, the algorithm halts and outputs a message indicating that it does not have sufficient data to identify the correct SL FST. Of course, the learner will also fail if the function the data samples is not actually ISL, though it cannot distinguish these two scenarios.

The next section will present a proof that the ISLFLA will always identify the correct $\mathcal{T}_{ISL}(f)$ provided the data set contains a *characteristic sample* for f and f is ISL. The proof relies in part on the fact that given such a sample (which will also be defined), the ISLFLA will *never* encounter the fatal situation in Figure 6.7.

6.2.2 Identification in the limit

This section presents a proof that the algorithm ISLFLA identifies the class of ISL functions in the limit from positive data, in the sense of Gold (1967). Before going through the formal proof, it may be useful to discuss the intuitions behind it. Essentially the proof follows from the fact that given a characteristic sample for the target function (which will be defined below), the algorithm merges *all* and *only* states with the same suffix of length $k-1$. The result will be a FST that meets the definition for $\mathcal{T}_{ISL}(f)$.

It is easy to see that merges in the outer loop only involve states with the same suffix of length $k-1$. This is also the case for merges that take place in the inner loop. Consider the scenario depicted in Figure 6.8, in which q is a state created by an outer loop merge. After *pushback*, states s and t will be merged. If $x = \texttt{Suff}^{k-1}(q)$, then both s and t must have xa as a suffix. Since $|xa| = k$, then it follows that $\texttt{Suff}^{k-1}(s) = \texttt{Suff}^{k-1}(t)$. It also follows that additional states merged to remove non-determinism resulting from the merge of s and t will have the same suffix of length $k-1$. To show that all states with the same suffix of length $k-1$ will be merged, the proof will show that the ISLFLA will *never* encounter the situation in Figure 6.7, provided the data set includes a characteristic sample defined as follows.

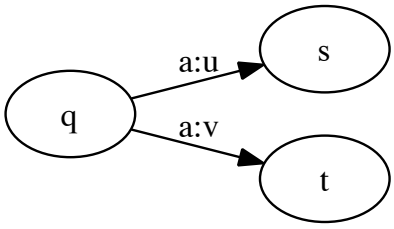


Figure 6.8: Non-determinism from an outer loop merge

Definition 7 (Characteristic Sample)**.** *Given a k-ISL function f and $\mathcal{T}_{ISL}(f)$, let $Q' = \{q \in Q$ such that $\sigma(q) \neq \lambda\}$. A characteristic sample for f is $S = S' \cup S''$, where*

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

The first part of Definition 7 says that the sample must include all possible substrings up to length k paired with their respective outputs according to f. The second part says that for those substrings of length k that end in a state in $\mathcal{T}_{ISL}$ that has a non-empty final output string mapped to it by σ, the sample must also include input-output pairs for the substring plus each of the symbols in the alphabet. This means that for those ISL functions for which σ maps the empty string to all states, the characteristic sample consists of only the first part. One such example will be presented in the next section.

Notice that this definition for a characteristic sample assumes that the target function is total. Though SL functions are not total by definition, this thesis has been treating the mappings of phonological processes as total functions for the reasons mentioned in Chapter 3. Whether the ISLFLA can identify partial functions, and what a characteristic sample would be in those cases, is not yet known.

The following theorem establishes the result of identification in the limit for the ISLFLA. Its proof follows.

Theorem 4. *The ISLFLA identifies the class of ISL functions in the limit.*

Proof. It must be shown that given an ISL function f and a dataset D such that S $\subseteq$ D, where S is a characteristic sample as defined in Definition 7, the output of the ISLFLA is $\mathcal{T}_{ISL}$.

Let $PTT(D) = (Q_{PTT}, q_{0PTT}, \Sigma, \Gamma, \delta_{PTT})$ be the input to the ISLFLA and $\mathcal{T} = (Q_{\mathcal{T}}, q_{0\mathcal{T}}, \Sigma, \Gamma, \delta_{\mathcal{T}})$ be its output. First it is shown that $Q_{\mathcal{T}} = \Sigma^{\leq k-1}$. By definition of $PTT(D)$ (Definition 6) and S (Definition 7), $\Sigma^{\leq k-1} \subseteq Q_{PTT}$. Since the ISLFLA only merges states with the same suffix of length $k-1$, $\Sigma^{\leq k-1} \subseteq Q_{\mathcal{T}}$. Since it does not exit until all states q have been merged with $\texttt{Suff}^{k-1}(q)$, $Q_{\mathcal{T}} = \Sigma^{\leq k-1}$.

Next it is shown that given S, it will never be the case that two states q_1 and q_2 are merged such that $\delta_1(q_1, \#) \neq \delta_1(q_2, \#)$. Let $\delta_1(q_1, \#) = z$ and $\delta_1(q_2, \#) = x$ with $z \neq x$ and $q_1 = \texttt{Suff}^{k-1}(q_2)$. By definition of ISL functions (Definition 4, Chapter 3), $\texttt{tails}_f(q_1) = \texttt{tails}_f(q_2)$, so if $z \neq x$ it must be the case that q_2 does not have transitions for all $a \in \Sigma$. This is because the only way for the output strings of the outgoing transitions of q_2 to differ from those of q_1 is if fewer transitions were present on q_2 when the PTT was made onward. (By definition of S we know q_1 has transitions for all $a \in \Sigma$.) But since $\texttt{tails}_f(q_1) = \texttt{tails}_f(q_2)$, we also know that $z = ux$ for some $u \in \Gamma^*$.

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

It remains to show that $\forall q \in Q_{\mathcal{T}}, a \in \Sigma, \delta_2(q, a) = \texttt{Suff}^{k-1}(qa)$. Since state merging preserves transitions, this follows from the construction of $PTT(D)$. By Definition 4 (Chapter 3), $\mathcal{T} = \mathcal{T}_{ISL}$. □

6.2.3 Complexity results

This section establishes complexity bounds on the runtime of ISLFLA and the size of the characteristic sample for ISL functions.[2]

6.2.3.1 Time complexity

Let n denote the number of states of the PTT; n is at most the sum of the lengths of the input strings of the pairs in the sample. Let m be the length of the longest output string in the sample. We show that the time complexity is in $O(n \cdot m \cdot k \cdot |\Sigma|)$. First, making the PTT onward can be done in $O(m \cdot n)$: it consists of a depth-first parsing of the PTT from its root, with a computation at each state of the `lcp` of the outgoing transition outputs after the recursive computation of the function (see de la Higuera (2010), Chap. 18, for details). As the computation of the `lcp` takes at most m steps and has to be done for each state, the complexity of this step is effectively in $O(m \cdot n)$.

For the two loops, we need to find a bound on the number of merges that can occur. States q such that $|q| < k$ do not yield any merges in the outer loop. All other states q' are merged with $\texttt{Suff}^{k-1}(q)$, in the outer or inner loop. The number of merges is thus bounded by n. Computing the suffix of length $k-1$ of any word can be done in $O(k)$ with a correct implementation of strings of characters.

The test of the inner loop can be done in constant time and so can the merge and *pushback* procedures. After each merge, the test of the inner loop needs to be done at most $|\Sigma|$ times. As computing the `lcp` has a complexity in $O(m)$, the overall complexity of the two loops is in $O(n \cdot m \cdot k \cdot |\Sigma|)$.

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

6.2.3.2 Size of the characteristic sample

Let $\mathcal{T}_{ISL} = (Q, q_0, \Sigma, \Gamma, \delta, \sigma)$ be the target transducer for f. We define $m = \texttt{max}\{|u| \mid (q, a, u, q') \in \delta\}$ and $p = \texttt{max}\{|v| \mid (q, \#, v) \in \sigma\}$. The first part of the

[2] The results in this section were established in collaboration with Jeffrey Heinz and Rémi Eyraud.

characteristic sample, S', covers all and only the states of the target: the set of input strings of these pairs is thus linear in n and the set of output strings is at most $n \cdot m + p$. Thus the size of S' is at most $n \cdot (n + n \cdot m + p) = O(n^2 \cdot m + n \cdot p)$.

Concerning the second part S'' of the characteristic sample, its cardinality is at most $n \cdot |\Sigma|$ (in the rare case where $Q' = Q$). Each input string of S'' is of size $k + 1$ and each output string is at most of length $(k + 1) \cdot m + p$. The size of S'' is thus in $O(n \cdot |\Sigma| \cdot (k \cdot m + p))$. Therefore, the size of the characteristic sample is in $O(n \cdot |\Sigma| \cdot k \cdot m + n^2 \cdot m + n \cdot |\Sigma| \cdot p)$, which is clearly polynomial in the size of $\mathcal{T}_{ISL}$.

We observe that both of the bounds established are improvements over OSTIA, which has a complexity of $O(n^3(m + |\Sigma|) + nm|\Sigma|)$ (de la Higuera, 2010). This result is not surprising, since ISL functions have less generative power than subsequential functions, and it demonstrates how greater a priori knowledge enables learning with less time and data.

6.3 Demonstrations

This section presents four demonstrations of the ISLFLA using natural language examples. These demonstrations include two examples of substitution processes - German final devoicing and English flapping - as well as one example of deletion (from Greek) and one epenthesis process (from Dutch).

6.3.1 Substitution

6.3.1.1 German final devoicing

Final devoicing, attested in German (as well as Russian, Dutch, Polish, and several other languages), is a process by which a word-final voiced obstruent surfaces as its voiceless counterpart.

(4) German

a. /ba:d/ ↦ [ba:t] ‘bath’

b. /sa:g/ ↦ [sa:k] ‘say’

c. /wald/ ↦ [walt] ‘forest’

A rule for this process is shown in (5), where D is a voiced obstruent and T is a voiceless obstruent.

(5) D → T / __#

The ISLFLA was given an artificial corpus of input-output string pairs for the final devoicing mapping. Since $k = 2$, the input forms were all possible words up to length 3 over the alphabet $\Sigma = \{D, T, N\}$, where D and T are defined as above and N is a sonorant. Each input was paired with a corresponding output form in which the above rule was applied if the necessary context was present in the input form. The total corpus consisted of 40 string pairs. The output of the learner, shown in Figure 6.9, is the correct $\mathcal{T}_{ISL}$ for the SL-2 function that models final devoicing. Note that the actual FST that the learner outputs has transitions on # that lead to a final state. The output on these # transitions corresponds to the final output string assigned to that state (since that transition is only taken if the end of the input has been reached). Such a representation differs from, but is equivalent to, the FSTs presented in previous chapters in which all states are final and the final output string is included in the state label. The FSTs presented in this section have therefore been converted to be consistent with those in previous chapters.

6.3.1.2 English flapping

The second test case is the process of flapping in English, exemplified in (6) and described with the rule in (7).

(6) English

a. /bʌtɾ̩/ ↦ [bʌɾɾ̩] ‘butter’

b. /wɑtɾ̩/ ↦ [wɑɾɾ̩] ‘water’

(7) t → ɾ / V́ __V

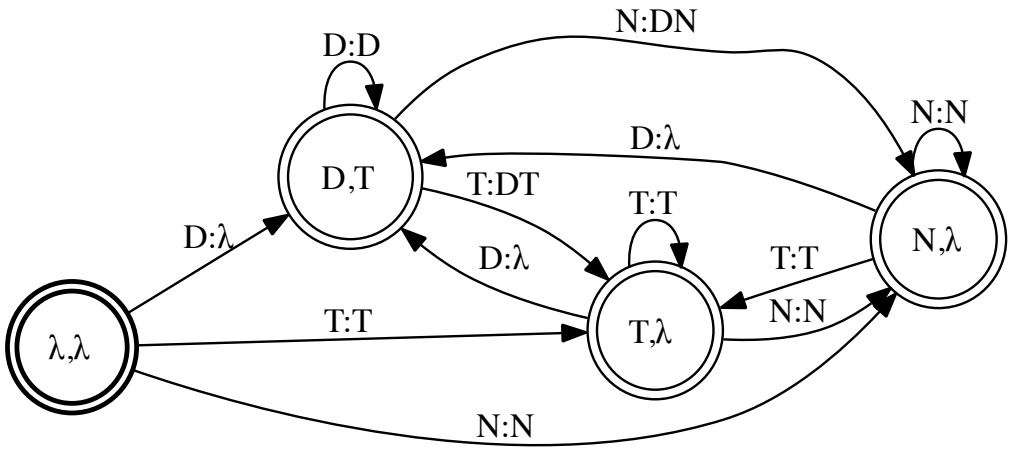


Figure 6.9: ISLFLA output for German final devoicing test, $k = 2$, $\Sigma = \{\text{D, T, N}\}$

The ISLFLA was given an artificial corpus of input-output pairs for this rule. This time $k = 3$, so the inputs were all possible words up to length 4 over the alphabet $\Sigma = \{\text{V, v, t, ?}\}$, where V is a stressed vowel, v is an unstressed vowel, and ? is any segment in English except V, v, or t. This resulted in a corpus of 340 pairs. The output of the ISLFLA is shown in Figure 6.10. Note this FST is minimized for readability. This output is the correct $\mathcal{T}_{ISL}$ for the SL-3 function that models English flapping.

6.3.2 Greek fricative deletion

The test case for deletion is a process in Greek by which an interdental fricative is deleted when it precedes a voiceless coronal fricative (Joseph and Philippaki-Warburton, 1987). This process is described by the rule in (8).

(8) $\{\theta, \eth\} \rightarrow \emptyset \; / \; \_\{\text{s, } \theta\}$

The corpus for this process was again artificially generated, using the alphabet $\Sigma = \{\theta, \eth, \text{s, ?}\}$, where ? represents any segment aside from θ, ð, or s. All possible strings up to length 3 (since $k = 2$) were created from this alphabet, resulting in a corpus of

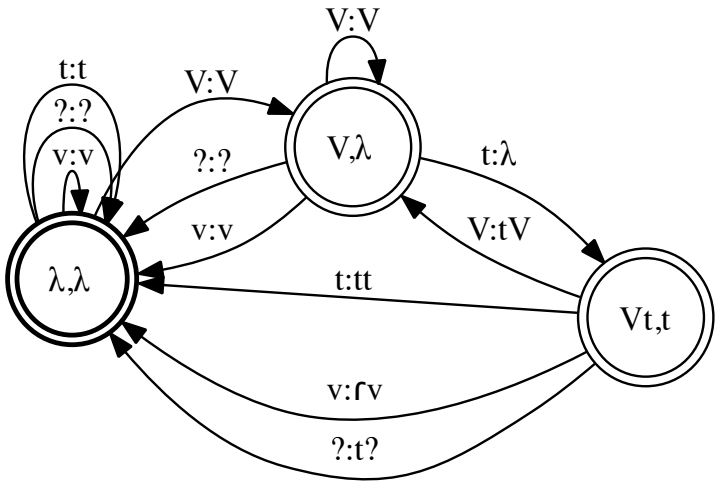


Figure 6.10: ISLFLA output (minimized) for English flapping test, $k = 3$, $\Sigma = \{$t, V, v, ?$\}$

53 pairs. The output of the learner is shown in Figure 6.11 and is the correct $\mathcal{T}_{ISL}$ for the SL-2 function that models this deletion process.

6.3.3 Dutch schwa epenthesis

The final test case is a process of ə-epenthesis found in Dutch. As described with the rule in (9), this process inserts a schwa between a liquid and a [-coronal] consonant (Warner et al., 2001). Examples are shown in (10).

(9) $\emptyset \rightarrow$ ə / {l, r} __[-coronal]

(10) Dutch

a. /mɛlk/ ↦ [mɛlək] 'milk'

b. /ʊɪlx/ ↦ [ʊɪləx] 'willow'

c. /hʏlp/ ↦ [hʏləp] 'help'

Again $k = 2$ for this process, but this time only input forms up to length 2 are needed for the learner to identify the correct function. This is a case in which the characteristic

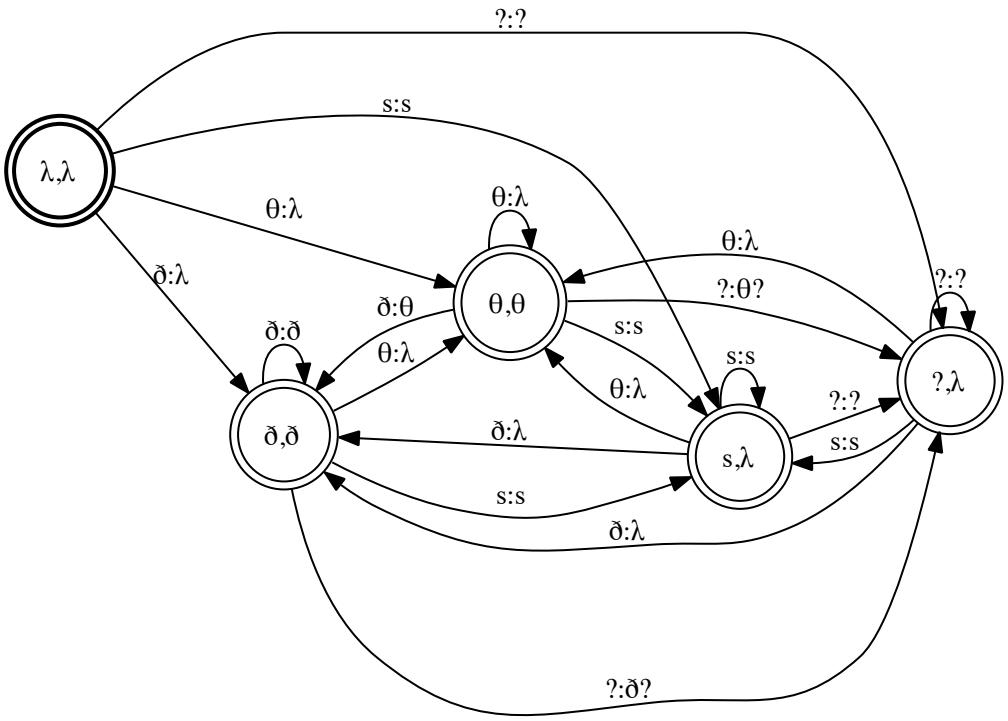


Figure 6.11: ISLFLA output for Greek deletion test, $k = 2$, $\Sigma = \{\theta,\eth,s,?\}$

sample includes only part one of Definition 7, because the final output function of $\mathcal{T}_{ISL}$ maps the empty string to all states (i.e., in Definition 7, $Q' = \emptyset$). As the proof of Theorem 4 showed, the need for strings of longer length stems from the need to avoid merging states with different final output strings. When all final output strings are λ, this conflict is avoided.

All possible inputs up to length 2 were generated using the alphabet $\Sigma = \{$l, r, K, ?$\}$, where K represents any [-coronal] consonant and ? represents any other segment in the Dutch inventory aside from those already in the alphabet. These input forms were paired with output forms according to the epenthesis rule to form a corpus of 21 pairs. The learner outputs the correct $\mathcal{T}_{ISL}$ for the SL-2 function that models this epenthesis process, as shown in Figure 6.12.

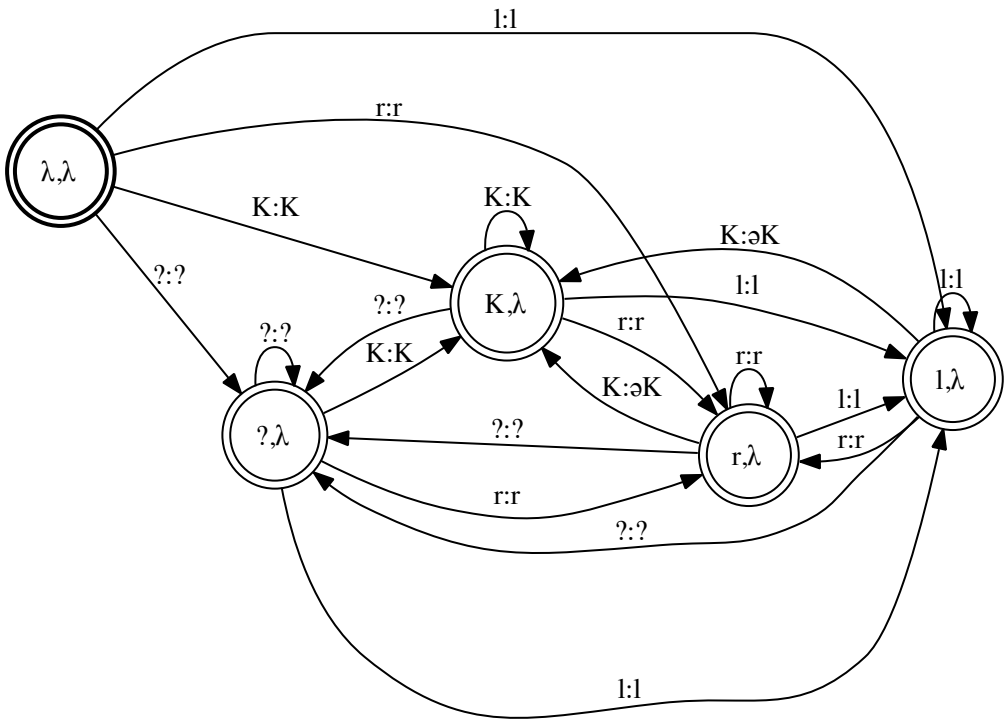


Figure 6.12: ISLFLA output for Dutch epenthesis test, $k = 2$, $\Sigma = \{$l, r, K, ?$\}$

6.3.4 Discussion of learning results

This chapter has presented a learning algorithm for local processes that uses strict locality as a learning bias to generalize an ISL function for a target phonological mapping from positive data that reflects that mapping. It also provided a proof that given sufficient data the learner will learn any ISL function. It is clearly necessary and desirable to extend these results to OSL functions. Though that extension will be left for future work, it appears that the learning of OSL functions will require only a single change to the current ISLFLA. Recall that the key difference between the FSTs that model ISL and OSL functions is in the transition function. For ISL functions, the

destination state of each transition corresponds to the most recent $k-1$ symbols of the input. For OSL functions the destination state of each transition corresponds to the most recent $k-1$ symbols of the output. Just by nature of how the initial prefix tree transducer is constructed by the learner (i.e., the states are prefixes of the input strings in the data set), the learner only has access to the input side of the transduction when it performs state merging. As a result, the FST the learner converges on is for an ISL function.

The proposed modification that should enable the algorithm to learn OSL functions is for it to merge states that were reached by the same $k-1$ *output* symbols. This will presumably require a different construction for the initial prefix tree transducer, so that the states are prefixes of the output strings of the data set. Also, as the method of removing non-determinism involves shifting the transition outputs, care will have to be taken so that the destination states always correspond to the most recent output symbols. Development of this version of the learner, as well as the corresponding proof that it can learn any OSL function, will be an important next step in the current research program.

To more clearly articulate the contributions of the current learning results, it is useful to compare them to the results for OSTIA (Oncina et al., 1993), Gildea and Jurafsky (1995, 1996)'s modified OSTIA, and the learners for local processes proposed in Chandlee and Koirala (2014) and Chandlee and Jardine (2014). First, the fact that all SL functions are subsequential functions and the fact that OSTIA can learn any subsequential function together mean that OSTIA can learn any SL function. And indeed it can, provided it is given a characteristic sample as defined in Oncina et al. (1993). However, the other relevant fact is that the SL functions are a *proper* subset of the subsequential functions, which means OSTIA can learn many functions that are not SL. So if the human phonological learner used only subsequentiality (and not strict locality) as a learning bias, then these subsequential but non-SL functions would be within its reach. Consider again the rule in (11) from Chapter 1.

(11) /i/ → [u] / x __(x is a string with an even number of obstruents)

This process could not be represented with a SL FST, but it could be represented with a subsequential FST, because a subsequential function (like the one in (1)) can distinguish between strings with an even and an odd number of obstruents. But such a process is unattested, and many if not most phonologists would consider it implausible. Following Heinz (2007), the view advocated in this thesis is that the range of 'possible' phonological patterns is delimited in part by the manner in which they are learned. Under this view, if humans use only subsequentiality as an inductive principle when learning phonological patterns, the absence of processes like (11) in the typology is left unexplained. In contrast, the proposal that the phonological learner assumes not just subsequentiality but also strict locality explains both how local processes can be learned and why processes like (11) are not found (i.e., because they can't be learned). In addition, because the ISLFLA allows for only a subset of the state merges that OSTIA attempts, and because it does not search for a state to merge another state with (i.e., it merges each state with its own suffix of length $k-1$), the ISLFLA can converge on the correct FST in less time and with less data than OSTIA, as was shown in §6.2.3 above.

Next, while it is important to note that the study of GJ and the current work had different objectives, a comparison of the approaches is instructive for understanding how the ISLFLA fits into a larger model of phonological learning. As discussed above in §6.1.1, GJ modified OSTIA so that it could learn phonological rules from natural language data, with the primary objective of showing the usefulness and importance of their incorporated learning biases - community, faithfulness, and context. In fact the strict locality learning bias of the ISLFLA is an alternative implementation of their context bias, as it too exploits the fact that phonological changes are defined in terms of some amount of material to the left and right of the segment affected by the change. The other two biases in their study, community and faithfulness, addressed problems OSTIA faced when using natural language data. As discussed above, the issue with a

natural language corpus is that on account of the language's phonotactic constraints it does not provide sufficient information for generalizing a total function. In particular, the prefix tree transducer built from such a corpus will not have transitions from all states for all alphabet symbols, reflecting the fact that not all symbols can follow all other symbols in a given language (e.g., the word-initial t state in a tree built from an English corpus would not have an outgoing transition for 'l', because the sequence $\#tl$ is not permitted in English).

These missing transitions affect the way the outputs are shifted when the prefix tree is made onward. Recall that onwardness ensures that the longest common prefix (`lcp`) of all output strings of all transitions leaving a state is λ. If a state is missing transitions for one or more alphabet symbols, then this `lcp` will be determined based on a smaller number of strings than if the state had transitions for all symbols. This is problematic when trying to generalize a total function from such data, because the `lcp` will be computed based on incomplete information. In other words, the `lcp` is the output that the FST can be confident about producing after a certain amount of input. The missing transitions can be thought of as counterexamples that, if present, would cause the FST to output a different `lcp`.

GJ handle this problem with their faithfulness bias, which forces a certain alignment between the input and output strings of the data set. This alignment prevents the incorrect shifting of outputs that OSTIA allows, but it also means the prefix tree transducer is not onward. As mentioned above, the assumption that the transducer is onward is important for the proof that the learner learns any function in the class. So a modification to the algorithm that improves its performance on natural language data comes as the expense of identifying the class of functions it learns.

The artificial corpora that were constructed for the tests of the ISLFLA consisted of all possible strings over the alphabet up to a certain length. This means that, unlike in a natural language corpus, there were no restrictions on the sequences of symbols in a string, which in turn means the prefix tree transducers were not missing any crucial transitions. Thus the algorithm succeeded without a forced alignment like that

employed by GJ. However, when the ISLFLA is run with natural language data it makes the same mistakes that OSTIA does when adjusting the outputs on the transitions (i.e., it moves more output than it should). On the one hand, this suggests a need for the ISLFLA to be modified in some fashion similar to GJ's modifications to OSTIA so that it too can handle natural language data, though preferably in a way that maintains the result of identification in the limit. On the other hand, this apparent shortcoming of the ISLFLA actually speaks to the larger question of how the phonological learning problem should be factored. In both the tests of the ISLFLA presented above and GJ's tests of OSTIA, the learner is being tasked with learning a single phonological mapping (e.g., final devoicing, flapping, etc.). And both learners approach this task with an essentially blank slate: OSTIA has no existing knowledge of the language this mapping is a part of, and the ISLFLA is given only the value of k.[3] But the state of the human phonological learner is not believed to be so impoverished when it begins to learn phonological processes.

There is evidence from the experimental language acquisition literature that knowledge of a language's phonotactics precedes knowledge of processes. Phonotactic knowledge is evident as early as nine months (Friederici and Wessels, 1993; Jusczyk et al., 1993b,a, 1994), while knowledge of alternations/processes comes later (Tesar and Prince, 2003; Hayes, 2004; Kerkhoff, 2004). In addition, based on evidence from artificial language learning experiments, Pater and Tessier (2003) suggest that knowledge of phonotactics can assist in the learning of processes. If that is the case, testing learners like OSTIA or the ISLFLA with natural language corpora makes little sense, since the task in that case is to *simultaneously* learn both the process in question *and* phonotactic knowledge it has already acquired. The question then is how to augment the learner with this previously acquired knowledge. One option is to give it data of the form used in the demonstrations presented above (i.e., data for all possible sequences

[3] An obvious question is how can it know the value of k for the function it is learning before it has learned it. It remains an issue for future work whether a learner can figure out both the function and the k-value.

of segments) to simulate that it is ignoring the phonotactic restrictions it has already figured out. If, however, the human phonological learner is in fact *using* its existing knowledge of phonotactics when learning processes, then computational models like the ISLFLA should incorporate and exploit that knowledge more directly.

Alternatively, it might be possible to enable the ISLFLA to induce the missing data from the given data. How? Recall the Suffix Substitution property of the SL languages (discussed in Chapter 2), which guarantees that u_1xv_2 is in a SL-k language if u_1xv_1 and u_2xv_2 are both in the language and $|x| = k - 1$. The defining property of SL functions (i.e., the same suffix of length $k - 1$ implies the same tails) is similar in spirit to the SSC, and the learner could use it to create additional data from the existing data and thereby reduce the needed characteristic sample. The details of such a mechanism need to be worked out, but it suggests potential for the ISLFLA to succeed with natural language data.

The third and final learning bias employed by GJ is called community, and it encodes the fact that segments within natural classes tend to behave similarly. In other words, it is not a coincidence that final devoicing targets the set of segments {b, d, g, z, ʒ, dʒ}, since these segments comprise the natural class of voiced obstruents. Neither OSTIA nor the ISLFLA have the ability to deduce that if a process applies to {b, d, ʒ} then it is likely to also apply to {g, z, dʒ}. GJ equipped OSTIA with a bias that allowed it to make just such a deduction.

On the one hand, the tests of the ISLFLA presented above avoided this issue entirely by using reduced alphabets (e.g. {T, D, N} for German). This was done primarily for readability of the resulting FSTs. In fact the learner will also succeed on alphabets that correspond to the complete phoneme inventory of each language. As a small test case, consider the output of the ISLFLA for final devoicing with the alphabet $\Sigma = \{b, d, g, V\}$, shown in Figure 6.13.

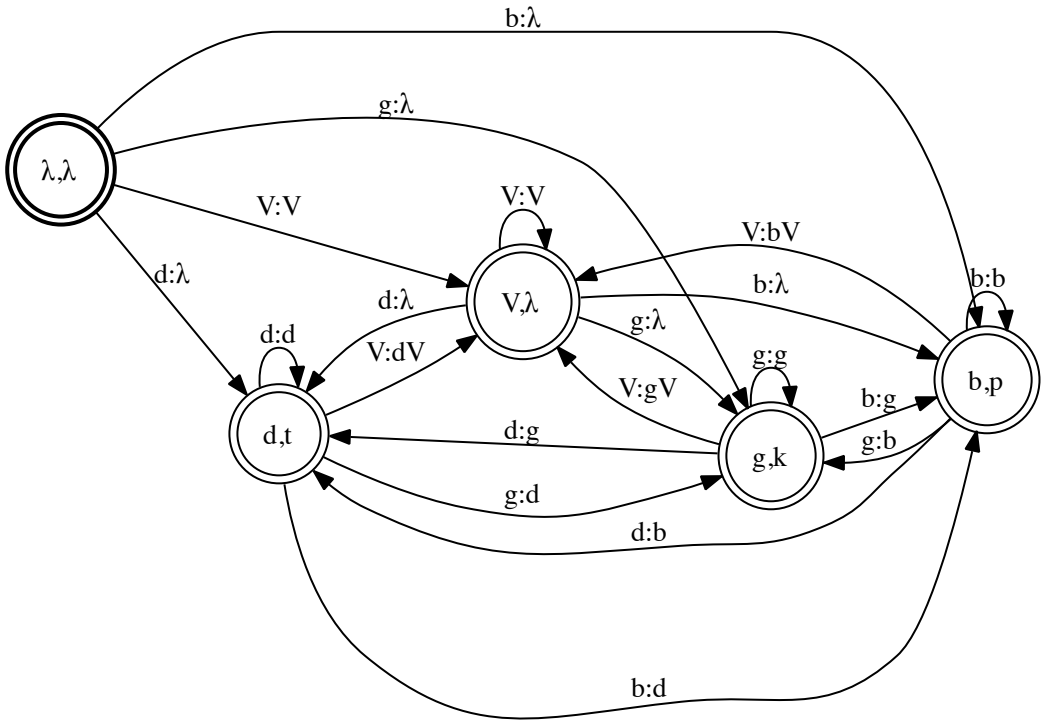


Figure 6.13: ISLFLA output for German final devoicing with $\Sigma = \{b, d, g, V\}$

As can be seen in the figure, the ISLFLA again correctly learns final devoicing, including the fact that d $\mapsto$ t, b $\mapsto$ p, and g $\mapsto$ k. This shows that its success on the {T, D, N} alphabet was not due to the simplifying assumption that the process only affects D $\mapsto$ T. As before, what is crucial is that the data set includes the characteristic sample with all possible sequences from the alphabet. In this sense, GJ's modification to OSTIA that allows it to generalize natural classes also serves as another mechanism for filling in missing transitions. The fact remains, however, that the ISLFLA neither uses nor learns any knowledge of natural classes of segments. By assumption, the human learner already has knowledge of natural classes when it learns phonological processes - either because this knowledge is part of UG or else because it was acquired at an earlier stage of acquisition. So as with phonotactic knowledge, the ISLFLA should not necessarily be expected to learn natural classes at the same time it learns

processes. But, also as in the case of phonotactics, if in fact humans are exploiting their knowledge of natural classes when they learn processes, then the computational model should indeed incorporate and make use of this knowledge. As generalizations over segments, natural classes could serve as a useful and principled mechanism for inducing the crucial missing data.

This mechanism could be in the form of a simplicity bias, with the learner's line of reasoning proceeding something like this: for an observed process that targets both /b/ and /d/, it is simpler to assume that it also targets /g/. Some manner of correction must also be incorporated, in the event the learner observes a counter-example to an earlier assumption (i.e., it notices that in fact the process does not target /g/). Initially it would assume a process targets the largest natural class the observed segments belong to, but then it could gradually shrink that class to accommodate the full set of data. This is another area in which the use of feature alphabets could lead to important developments.

Lastly, it should be noted that the ISLFLA presented in this chapter has evolved from previous learning results presented in Chandlee and Koirala (2014) and Chandlee and Jardine (2014). The learning algorithms proposed in those works also involved the two stages of 1) building a prefix tree transducer and 2) merging states with the same suffix of length $k-1$. The former was the first to demonstrate the utility of this state merging strategy for learning mappings, but it could not guarantee that its output would be subsequential because the prefix tree was not onward and the algorithm lacked a principled mechanism for removing non-determinism. Chandlee and Jardine (2014) addressed these shortcomings by modifying OSTIA itself with the SL state merging criterion. Thus the learner's output was guaranteed to be both subsequential and SL, but only if the data set was closed under a certain property that prevents the problematic situation depicted in Figure 6.7 above. The stronger learning result of identification in the limit was possible for the current version of the ISLFLA because it discards OSTIA's searching strategy and instead merges each state directly with its own suffix of length $k-1$.

To conclude, this chapter has presented a learning algorithm that demonstrates the utility of strict locality as an inductive principle when generalizing from finite data. If human learners likewise generalize in this way when learning local phonological mappings, then these results suggest a reason why these mappings have this property in the first place: without it they'd be unlearnable. The larger implication is that phonological learning is modular, since clearly non-SL processes will also fall outside of the purview of this learner. The next chapter will include a discussion of how these results might be extended to the learning of non-SL processes.

For formal learning theory, the main contribution of this chapter is that the class of Input SL functions is identifiable in the limit from positive data. For language acquisition, this result demonstrates some utility for the property of strict locality when generalizing from positive data, though further work is needed to better understand how strict locality interacts with other aspects of the total system of phonological learning.

Chapter 7

DISCUSSION

This chapter will further discuss the contributions of this thesis, as well as remaining issues and areas for future work. This discussion is divided into three sections: §7.1 quantifies the empirical coverage of SL functions by reporting the results of a survey of the P-Base database (v.1.95, Mielke (2004)) and discussing the potential of extending the SL analysis to the non-SL exceptions, §7.2 further discusses the implications of the learning results presented in the previous chapter, and §7.3 addresses the relationship between computational explanations of typology and other explanatory factors like phonological naturalness.

7.1 Empirical Coverage of SL Functions

Chapters 3, 4, and 5 demonstrated the range of phonological and morphological processes that can be modeled with SL functions. To better quantify the empirical coverage of strict locality, a review was conducted of the approximately 5500 patterns from over 500 languages in the P-Base database (v.1.95, Mielke (2004)) to determine how may of them are SL-describable. A process was considered SL if a k-value could be identified such that a SL-k transducer could be constructed to model the process. By this measure, 94% of the processes in the database are *necessarily* SL. The remaining 6%, which includes suprasegmental processes, across-word-boundary and pre/post-pausal processes, and vowel and consonant harmony, are either non-SL or SL given additional assumptions. These cases will be addressed further below. The results of this survey lend weight to the claim that strict locality is a strong property of many phonological processes, but it is *not* meant to reflect the cross-linguistic proportion of SL to non-SL

processes. Indeed this particular database was not intended to represent the distribution of local and long-distance processes.[1] And in fact, as will now be discussed, future work could reveal that some of the processes in the remaining 6% are also SL.

First, this thesis has focused on segmental phonology, putting aside processes like stress and tone assignment. At the phonotactic level, Heinz (2007) and Rogers et al. (2013) have shown that most stress patterns are SL for some k, which means the set of well-formed strings according to the surface stress pattern is a SL language. This finding suggests some potential for the mappings that correspond to stress assignment to likewise be describable with SL functions. Consider again a rule of the form in (1).

(1) $\quad A \rightarrow B \ / \ C \_\_ D$

One way to think of the process that this rule represents is as a repair for a violation of the phonotactic constraint *CAD. Such a constraint can itself be modeled with a SL language, where $k = |CAD|$. So there is some correspondence between SL phonotactics and SL processes. Stress assignment (and tonal) patterns comprise only 0.6% of the patterns in P-Base, but other databases exist that provide a more comprehensive cross-linguistic view of stress assignment.[2] As for tonal processes, it is already known that they are not all SL, as Jardine (2013) discusses one category (tonal plateauing) that is not even subsequential. But as with stress assignment, the interest will lie in determining whether non-SL patterns are the exception or the norm in suprasegmental phonology.

Second, approximately 3% of the processes in P-Base take place across a word boundary or are triggered by a pause. An example of the former is found in Slovene, as shown in the examples in (2). The rule for this process is shown in (3) (Herrity, 2000).

---

[1] The sample reflected what grammars were present in the library of Ohio State University (circa 2004), where it was originally created.

[2] e.g., UD Phonology Lab Stress Pattern Database (http://phonology.cogsci.udel.edu/dbs/stress/), StressTyp (http://www.unileiden.net/stresstyp/)

(2) Slovene

a. /z/ brátom ↦ z brátom 'with the brother'

b. /z/ míze ↦ z míze 'from the table'

c. /z/ stréhe ↦ s stréhe 'from the roof'

(3) /z/ → [s] / __## [-voice]

An example of a pre-pausal process is found in Bengali, as reported by Ray (1966). The rule is shown in (4)

(4) Bengali

/b^h/ → [β] / before pause or juncture

Processes such as these were classified with the 6% that are not *necessarily* SL, because they are only SL if the FST alphabet can be augmented with symbols to represent certain boundaries. Indeed it is easy to see that the process in (3) targets a sequence like *z##b*, which is bounded by length 4. The current construction of a SL FST assumes a mapping from UR ↦ SR in which the UR and SR are single words. But if that assumption is relaxed and if the alphabet includes a symbol for a between-word boundary, then we can model (3) with a SL FST.[3] This between-word boundary symbol needs to be distinct from the morpheme boundary symbol, which is typically either + or -, since some processes, like the one in (5), may be triggered by morpheme boundaries but not between-word boundaries.

(5) Catalan (Wheeler, 1979)

a. ∅ → d / {n, l} __+r

b. /bɔl+ɾ+ɛ+m/ ↦ [buldɾɛm] 'we shall want'

c. /ofɛn+ɾ+ɛ+m/ ↦ [ufəndɾɛm] 'we shall offend'

[3] The FST will actually be SL-3, since the between-word boundary is now a single symbol.

This extension of the UR $\mapsto$ SR mapping from single words to multiple-word strings was also assumed in SPE. According to that theory, the input to the phonological component of the grammar is not in fact single words, but the sentences (i.e., strings of formatives) that were constructed by the syntactic component (Chomsky and Halle, 1968, pg. 9). Thus the existence of processes triggered by various boundaries (e.g., morpheme, word, between-word) is expected, and the SL FST formalism should be able to handle them. But a question is raised regarding how many different types of boundaries to allow. The answer relies on the correct description of the processes that are being modeled - provided the boundary has psychological reality (i.e., speakers clearly only apply the process in the presence of the boundary) its inclusion in the alphabet is justified. The prediction is that the processes are still SL regardless of which type of boundary they involve, though it would be interesting to investigate further whether there are differences among the various types of boundary processes. In particular, since the boundary type can correspond to different interfaces (phonology-morphology, phonology-syntax, morphology-syntax), it may be the case that the computational properties of these processes also differ. If not, this would be further evidence that the prevalence of the SL restriction extends beyond phonology proper.

As for processes triggered by a pause, such as (4), these too can be accommodated by adding a designated symbol to the alphabet to represent 'pause', though there is less theoretical support for doing so. It is unclear what component of the grammar is involved in such alternations, since clearly the pauses in a string are not represented underlyingly. For present purposes, it suffices to note that IF we allow for a 'pause' symbol in the string, the process is SL, though more needs to be understood about how incorporating such discourse factors affects the predicted range of what is phonologically possible.

Third, just over 2% of the patterns in the database are vowel harmony patterns. Vowel harmony is among the most prominent examples of non-local or long-distance processes, but its SL status in fact depends on whether or not the language has transparent vowels. Consider the two sets of vowel harmony rules in (6) and (7), where

α is the harmonizing feature. Rules (6-a) and (7-a) are for progressive (left-to-right) harmony, and rules (6-b) and (7-b) are for regressive (right-to-left) harmony.

(6) a. $V_{[-\alpha]} \rightarrow [+\alpha] \ / \ V_{[+\alpha]}(C)(C)\_\_$

b. $V_{[-\alpha]} \rightarrow [+\alpha] \ / \ \_\_(C)(C)V_{[+\alpha]}$

(7) a. $V_{[-\alpha]} \rightarrow [+\alpha] \ / \ V_{[+\alpha]}C_0\_\_$

b. $V_{[-\alpha]} \rightarrow [+\alpha] \ / \ \_\_C_0V_{[+\alpha]}$

The rules in (6) are for a language that allows only single-consonant codas and onsets, but they could be modified to accommodate more or less restrictive syllable structures. The idea is that the number of consonants that will intervene between the two vowels involved in the harmony process is bounded by the length of the maximum allowed coda + onset. The two vowels are not adjacent, but they are also not arbitrarily far apart, and so both of these rules could be modeled with OSL functions.

On the other hand, the rules in (7) cannot be modeled with SL functions, because the string represented by C_0 is unbounded. So the SL status of vowel harmony without transparent vowels depends on the correct generalization of the process: if (6) is correct then the process is SL, but if (7) is correct, then it is not. Vowel harmony is also not SL when the language has transparent vowels that do not participate in the harmony process. Once transparent vowels are introduced, there is no longer an upper bound on the amount of material that can intervene between the two harmonizing vowels.

The remaining patterns in the database include substitution processes that are conditioned by the presence or absence of another element in the string. An example from Sacapulteco is shown in (8) and exemplified in (9) (Dubois, 1981) (note: /-ʃ/ is the passive morpheme).

(8) /-ʃ/ → [s] / after stem containing {ts, ts', s}

(9) Sacapulteco

a. /sik'i+ʃ/ ↦ [sik'is] 'it was read'

b. /tsili+ʃ/ ↦ [tsilis] ‘it was returned’

c. /ts’ono+ʃ/ ↦ [ts’onos] ‘it was asked for’

d. /ku?um+asa+ʃ/ ↦ [ku?masas] ‘it was moved’

This consonant harmony process is not SL for any k, because the entire stem (which is unbounded in length) must be checked for the presence or absence of one of the triggering segments {ts, ts’, s}. Two things can be said about such processes. One, a large percentage of them are subsequential (Luo, 2013), which means they are subregular (Mohri, 1997). Two, the phonotactic restrictions that correspond to them can be modeled with a subregular language - just not a SL one. Heinz (2010) shows that the Strictly Piecewise (SP) languages can model the set of well-formed strings derived from long-distance consonant harmony processes. Recall the Subregular Hierarchy of formal languages, which was discussed in Chapter 2 and is shown again in Figure 7.1.

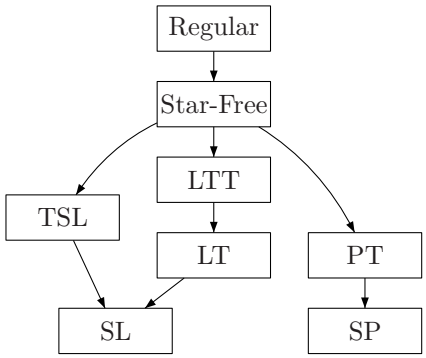


Figure 7.1: Subregular Hierarchy of formal languages (LTT = Locally Threshold Testable, PT = Piecewise Testable, TSL = Tier-based Strictly Local, LT = Locally Testable, SP = Strictly Piecewise, SL = Strictly Local)

The Strictly Piecewise (SP) languages differ minimally from the SL languages in that they are described with grammars of *subsequences* of length k instead of *substrings*. The difference is that a subsequence is not necessarily contiguous. So the substrings of length 2 of the string *abca* include {#a, ab, bc, ca, a#}, while the subsequences

of length 2 include {a...b, a...c, a...a, b...c, b...a, c...a}. In other words, subsequences encode restrictions in terms of *precedence* instead of *contiguity*.

The Sacapulteco process in (9) indicates that the language has a phonotactic restriction against the subsequences ts...ʃ, ts'...ʃ, and s...ʃ.[4] The set of well-formed surface forms is accepted by the FSA in Figure 7.2. In this FSA, S = {ts, ts', s} and ? represents any segment except for S and ʃ.

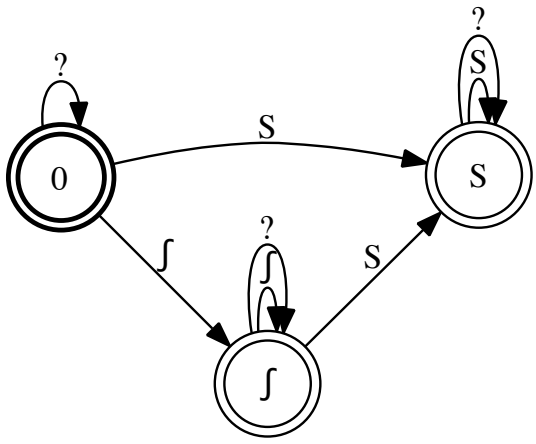


Figure 7.2: FSA for surface forms in Sacapulteco, Σ = {?, S, ʃ}

The FSA has separate states for S and ʃ, and once state S is reached there is no way to reach state ʃ. In this way it models the restriction against S preceding ʃ. As with the SL FSAs, however, this FSA does not model the process by which a ʃ that *does* follow a S becomes [s]. For that we need some functional counterpart to the SP languages, which to date does not exist. Though defining such a functional class is being left for future work, this thesis hopefully points to a fruitful approach: to combine the properties of subsequential transduction with the properties of SP languages. The resulting functional class has the potential to model long-distance consonant harmony processes such as (9), as well as long-distance dissimilation processes, which have also

[4] A similar restriction holds in Navaho (Sapir and Hoijer, 1967).

been shown to be largely subsequential (Payne, 2013). Long-distance harmony and dissimilation processes such as these are similar to the non-SL displacement cases discussed in Chapter 4, except they involve substitution instead of copying and deletion.

Likewise, the non-SL vowel harmony patterns, which are also subsequential (Gainor et al., 2012; Heinz and Lai, 2013), might be describable with a functional counterpart to the Tier-based Strictly Local (TSL) languages, which also has yet to be defined (see Heinz et al. (2011) for TSL phonotactics). The TSL languages are those for which SL constraints are defined over a specified tier of segments.[5] In the case of vowel harmony, this tier would contain only harmonizing vowels, and a SL-2 constraint would prohibit adjacent vowels that disagree in the harmonizing feature. This to-be-defined class of TSL functions will also be needed to model long-distance dissimilation with blocking, such as the Latin liquid dissimilation process exemplified in (10) (Jensen, 1974; Odden, 1994).

(10) Latin

a. /navalis/ ↦ navalis ‘naval’

b. /solalis/ ↦ solaris ‘solar’

c. /floralis/ ↦ floralis ‘floral’

Comparing (10-a) and (10-b) we see that a /l/ that follows another /l/ surfaces as [r], unless an /r/ already appears somewhere between the two /l/’s, as in (10-c). As discussed by Heinz (2010), the phonotactics of this pattern cannot be modeled with a SP language. The reason is that the change in (10-b) cannot be captured with a constraint against the subsequence *l…l*, since that constraint is violated by the well-formed *floralis*. With a k-value of 3, the subsequence *l…r…l* could be permitted, while *l…l…l* is ungrammatical. But this approach makes the wrong predictions, since a word like *floralisralis*, which should be well-formed, contains both *l…r…l* and *l…l…l*. More generally, the SP grammars cannot require the presence of a subsequence. They can

[5] Similar to autosegmental representations in phonology.

only state which subsequences are and are not permitted. But as with vowel harmony with transparent vowels, once a tier is defined (in this case a tier for liquids) the constraint can be reinterpreted as SL-2, with the substring **ll* prohibited.

It has been suggested that the SL, TSL, and SP languages are sufficient to model phonotactics (Heinz, 2013). By hypothesis, their functional counterparts - the SL functions presented in this thesis and the as yet undefined SP and TSL functions - will be sufficient to model processes.

While the primary topic of interest in this thesis has been phonological processes, it was also shown that the SL analysis extends to some types of morpho-phonological and morphological processes. The treatment of metathesis, for example, included cases that repair phonotactic violations or are triggered by prosodic factors like stress. But it also included cases like the Rotuman complete/incomplete distinction, which appears morphological in nature but is derived via CV-metathesis. Thus metathesis processes appear to span the phonology-morphology boundary. And Chapter 5 demonstrated that certain processes that are unequivocally morphological, like affixation and partial reduplication, are also SL. But there are also clear examples of morphological processes that are not SL, such as non-local partial and full reduplication. The previous research on computational properties of natural language has largely focused on phonology and syntax, while relatively less is known about morphology. It will be an interesting area of future research to determine where morphology is situated with respect to phonology and syntax in terms of the computational complexity of its processes. In particular, the computational properties of templatic morphology could lead to important insights. As morphological analysis of surface forms is a key area of natural language processing research, this could also be an area in which greater connections can be established between theoretical and applied computational linguistics.

One last direction in which to expand the empirical coverage of the SL FST formalism is free variation or rules that apply optionally. The deterministic nature of the SL FSTs requires each input string to be mapped to exactly one output, but certain phenomena are better modeled with a mapping from a single underlying form

to two (or more) surface forms. In addition, the choice of surface form is often modeled stochastically, with certain factors making one option more likely than another. One possible adaptation of the current SL FSTs that would maintain subsequentiality is the p-subsequential transducers studied by Mohri (1997). These transducers are still deterministic except for the final output function, which allows for up to p possible final output strings per state. Another route would be to combine the properties of SL languages with some type of non-deterministic weighted transducer, in which the outgoing transitions of each state are assigned weights corresponding to the probability of that particular path extension. These weights would be determined by training the FST on some representative data set. Since the learning algorithm presented in Chapter 6 relies in part on the assumption that its target is deterministic, a different algorithm would be needed to learn this new type of weighted transducer.

7.2 Implications of Learning Results

This thesis has demonstrated that the property of strict locality can be used as an inductive principle to learn a phonological mapping, but the learning algorithm presented in Chapter 6 is not meant to be a complete model of phonological learning. It was shown that this learner can learn a mapping corresponding to a single process, but the question remains whether the human learner targets processes individually or the total phonological grammar. The research reviewed in Chapter 2 on learning within OT assumes the latter scenario: equipped with the universal set of constraints, the OT learner is tasked with learning the grammar (i.e., the ranking) that directly maps the UR to the SR. An important question for future work is how strict locality can be used to learn not just individual processes but the total grammar.

Given a set of SL processes, their respective SL functions could be combined into a single function by composition. A learner like the ISLFLA could then be tasked with learning this total grammar function given data that samples it. However, the feasibility of this approach depends on whether SL functions have the property of *closure under composition*. Recall from Chapter 2 that the regular relations are closed

under composition; this means that the relation that results from composing two or more regular relations is guaranteed to also be regular. For SL functions, the question is whether the composition of two SL functions is also SL. In the case of OSL functions, the answer is no. Since closure properties serve as guarantees that the result of an operation (in this case composition) will always have the property in question, the existence of even one counter-example disproves the closure property.

A counter-example to the closure under composition of OSL functions can be constructed using the long-distance nasal assimilation example from Kikongo, shown in (11) (Meinhof, 1932; Dereau, 1955; Webb, 1965; Ao, 1991; Odden, 1994; Piggott, 1996; Rose and Walker, 2004). This process causes a suffix voiced stop to surface as nasal when the stem contains a nasal.

(11) Kikongo

a. /tu+kun+idi/ ↦ [tukunini] 'we planted'

b. /tu+nik+idi/ ↦ [tunikini] 'we ground'

The mapping in (11) cannot be modeled with a SL function, because the trigger of the process (a nasal in the stem) can be arbitrarily far away from the target in the suffix. However, the same mapping can be achieved as the composition of the two processes in (12) and (13).

(12) $[-\text{nasal}] \rightarrow [+\text{nasal}] \;/\; [+\text{nasal}]\ \_\_$

(13) $\{[-\text{cons}], [-\text{voice}]\} \rightarrow [-\text{nasal}] \;/\; \_\_$

Given the input /tunikidi/, applying the rule in (12) left-to-right produces the output [tunĩkĩnĩ]. Using this output as the input to the rule in (13) gives the output [tunikini]. Both of these processes are SL ((12) is OSL and (13) is both ISL and OSL). But the composition of (12) and (13) gives the mapping of /tunikidi/ ↦ [tunikini]. As stated

above, this mapping is not SL.[6]

It remains to be seen whether the ISL functions are closed under composition. Even if they are not, the ISLFLA can still be tested on cases in which the result of composing two (or more) ISL functions is ISL, since the lack of a closure property only means this is not guaranteed to be the case. It does not mean that such cases do not exist. Success on this kind of test case would lend further support to the suggestion that humans can learn phonology by generalizing in the way the ISLFLA does, but cases in which the composed function is not SL will certainly merit further attention. Do such cases exist in natural language, and, if so, how do humans learn them?

Notice that the two processes in (12) and (13) are similar in spirit to the copy-delete decomposition of phonological movement reviewed in Chapter 4. The two steps of copying and deletion were employed to model long-distance movement, and likewise the spreading (which can be seen as copying of a feature) and removal of nasality in this case achieves long-distance assimilation. The similarity between these cases raises two additional questions, one linguistic and one formal. The linguistic interest is whether all so-called long-distance processes can in fact be modeled as the composition of a series of local operations, as has been suggested in the literature (Flemming, 1995; Ní Chiosáin and Padgett, 1997; Bessell, 1998; Walker, 1998; Ní Chiosáin and Padgett, 2001). The formal interest is whether *any* regular relation and/or subsequential function can be decomposed into two SL functions.

It has already been established by Elgot and Mezei (1965) that any regular function is equivalent to the composition of two subsequential functions, one left and one right (see also Heinz and Lai (2013), who use this theorem to categorize various vowel harmony patterns). The left subsequential function reads the input left-to-right and possibly marks it up in some manner, and then the right subsequential function reads the output of the first function right-to-left and removes the markings. This is very similar to the treatment of nasal assimilation above - the first function marks up

[6] For a formal proof see the appendix.

segments with the nasal feature and the second one removes that marking from the segments that should not have it. Marking up segments in this way introduces abstract intermediate representations that never surface, so the contributions of such analyses tie into the debate in phonology on the right level of abstraction (see Kiparsky, 1968; Hyman, 1970; Kenstowicz and Kisseberth, 1977).

Of course the closure under composition property is only important under the assumption that processes are NOT learned individually. If they are learned individually, then the strategy of the ISLFLA is sufficient. But the distinction between learning individual processes versus an entire grammar really reflects a larger open question of the degree to which individual processes have psychological reality. The fact that German speakers apply final devoicing to a nonce word that ends in a voiced obstruent could be taken as evidence that these speakers 'know' the process of final devoicing. But this empirical fact is equally compatible with the explanation that they are merely submitting the nonce word to the entire grammar and the output has a word-final voiceless obstruent. Thus it would be an interesting future research direction to devise psycholinguistic studies that can provide evidence that processes have some autonomy in terms of mental representation. A careful examination of acquisition data could also provide insight into the intermediate states of the incomplete phonological grammar. For example, to what degree are the 'incorrect' grammars that children use on their way to acquiring the correct grammar best described as individual processes being missing, applied in the wrong order, etc.?

Perhaps the true role of strict locality in learning will only be revealed in the context of the total picture of phonological learning. The assumption for now is that it indeed plays a role, since otherwise its prominence among phonological processes would be reduced to coincidence. To put it another way, the findings of this thesis raise a question: why are so many processes SL? The proposed answer is that it is because those are the processes that can be learned. If there is no role for strict locality in learning, the question remains.

Of course this view is only compatible with a modular account of phonological

learning, since it is not the case that *all* processes are Strictly Local. Until comparable properties are identified for long-distance harmony and dissimilation processes, the details of this modular approach cannot be nailed down. The general idea, though, is that the complete phonological learner includes separate modules for separate classes of processes. This thesis has proposed that one of these modules uses the property of strict locality as an inductive principle to learn SL processes. Additional classes of processes will likewise be targeted by additional modules that employ corresponding inductive principles based on the computational property that defines their target class of processes. As discussed above, a working hypothesis is that there will be at least two such modules: one for SL processes and one for SP or TSL processes (or both). Given the evidence that separate learning modules are already needed for learning phonology and learning syntax (Lai, 2012, 2014), it seems reasonable to also posit separate learning modules within each of these domains.

7.3 Explanations of Typology

To return to the question posed at the beginning: what is a possible phonological process? This thesis has demonstrated the important role that computational properties play in accounting for the range of attested variation, but the complete picture of what is 'phonologically possible' will surely involve several interacting factors. In addition to computational properties, there seems to be a clear role for phonetic explanations, either optimization or diachronic reanalysis due to misperception, variation, etc. (e.g., Evolutionary Phonology (Blevins, 2004)). The need for some type of phonetic explanation is clear from the examples in (14).

(14) a. /n/ → ŋ / __{k, g}
b. /n/ → tʃ / __{k, g, b}

The rule in (14-a) is attested in many languages and has a clear phonetic explanation in that anticipatory co-articulation has caused the nasal to assimilate to the place feature of the following stop. As we saw in Chapter 3, this is a substitution process

that can be modeled with a SL-2 function. The rule in (14-b) can also be modeled with a SL-2 function, yet it has no clear phonetic motivation. As an analysis of some attested alternation, it is highly suspect. The fact that both of these processes - one expected and one not likely to be found in any language - are both SL suggests yet again that this class of functions is still too large to fully characterize phonological mappings. However, the needed restriction this time is not a computational one, since the two processes are computationally equivalent.

Recall again the rule schema that was used to generalize the kinds of processes that are SL, repeated in (15).

(15) $x_i \rightarrow y_i \ / \ U \ \_\_ V$

The use of an index on the x and y strings was justified by the fact that a target of a process cannot be changed into any arbitrary segment; its structural change is specified by the rule. In the case of final devoicing, the set X contains voiced obstruents and the set Y contains voiceless obstruents, but the definition of the rule includes pairing the strings in these sets appropriately ($(b,p),(d,t),(g,k)$, etc.). As part of the rule's definition, this prevents the FST from mapping b to k, but nothing prevents us from defining the pairings differently (e.g., $(b,t),(g,p),(d,k)$). In other words, nothing is incorporated into the formalism itself that can recognize which pairings are the correct ones. Likewise, nothing in the formalism indicates that {k, g} but not {k, g, b} is a reasonable set of right contexts for a change from /n/ to [ŋ].

Moving forward, then, it will be interesting to see how our understanding of phonetic naturalness can be translated into restrictions on the strings in X, Y, U, and V, as well as co-occurrence restrictions among the strings in these sets. Such restrictions would augment the current SL FST formalism in such a way that it combines computational and phonetic-based explanations of phonological typology. Two possible directions for establishing such restrictions are 1) identifying which k-factors are cross-linguistically marked, and 2) hypothesizing an upper bound on the k-value for

phonological processes. The goal would be to define a phonological subclass of the SL functions, which would still be within the scope of the ISLFLA proposed in Chapter 6.

Chapter 8

CONCLUSION

This dissertation has used computational analysis to propose a defining property of phonological and morphological processes with local triggers. It has defined a class of subregular functions and shown how these functions can model the mappings that underly such processes. The implication of this result is that this functional class delimits the expected range of variation for local processes. An explanation for why local processes are restricted in this way was provided through a demonstration that the defining property of strict locality can be used to learn these SL mappings. If the module of the phonological learner responsible for learning local processes is biased to only search the hypothesis space of Strictly Local mappings, then mappings without this property will not have a place in phonological grammars.

Since the computational analyses presented throughout the thesis describe the mapping from underlying to surface form directly, these results are independent of and compatible with both of the predominant theories of generative phonology, SPE and OT. Phonologists have used these theories to describe a wide range of attested phenomena, in the process revealing many insights into the mental representations of phonological grammars. As for the goal of identifying what is phonologically possible, the approach has been to generalize over the range of attested phenomena that have been described and establish guidelines for what is and is not there. In other words, conceptions of what is possible have been made based on the set of rules or constraints that have been proposed to account for attested patterns.

The alternative approach advocated by the current work is to assume that the set of possible processes is defined in part by their computational properties. For now, restricting our attention to local processes, the claim is that this set is the set of SL

functions, which are defined with a much stronger computational property than has been previously established for phonology. This claim leads to the hypothesis in (1).

(1) **SL hypothesis**: Local phonological processes are Strictly Local mappings.

Now the question is how to guarantee that the theory of phonology generates *only* the SL class. The answer will lie in a closer examination of the difference between the patterns we do and do not see and how the current formulation of rules and constraints (and their interactions) allow for the patterns we wish to rule out. This has always been a goal of research in generative phonology, but the findings of this thesis are meant to remove one of the uncertainties by identifying exactly the class we want to generate.

The combined evidence of the empirical coverage and learnability of SL mappings indicates that strict locality is the right notion of locality in phonological processes. Future work will seek comparable properties for long-distance processes, leading to a more complete characterization of the kinds of computations required by the phonological grammar.

As a working hypothesis, the SL hypothesis highlights the importance of looking for the exceptions. As was revealed by the discussion of displacement patterns in Chapter 4, the data collected about a language and the descriptions of its patterns typically do not reflect the goal of drawing computational distinctions. But the SL hypothesis can serve as a guiding principle for collecting the data that would crucially distinguish whether the pattern is SL or not. Such investigations will either uphold the hypothesis, or else provide informative exceptions. Under the assumption that the exceptions will be less probable than the rule, their appearance should spark further investigation into what other factors bring them about. Only through such examinations of the attested, the possible, and the improbable - examinations made possible by hypotheses like (1) - can we truly understand the nature of phonological processes.

BIBLIOGRAPHY

Abboud, P. F. (1979). The verb in northern Najdi Arabic. *Bulletin of the School of Oriental and African Studies*, 42:467–499.

al Mozainy, H. (1981). *Vowel alternations in a Bedouin Hijazi Arabic dialect: Abstractness and stress*. PhD thesis, University of Texas Austin.

al Mozainy, H., Bley-Vroman, R., and McCarthy, J. J. (1985). Stress shift and metrical structure. *Linguistic Inquiry*, 16:135–144.

Albright, A. and Hayes, B. (1999). An automated learner for phonology and morphology. Unpublished manuscript, UCLA.

Albright, A. and Hayes, B. (2002). Modeling english past tense intuitions with minimal generalization. In Maxwell, M., editor, *Proceedings of the 6th Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON)*.

Alderete, J., Brasoveanua, A., Merchant, N., Prince, A., and Tesar, B. (2005). Contrast analysis aids in the learning of phonological underlying forms. In Alderete, J., Han, C., and Kochetov, A., editors, *WCCFL 24: Proceedings of the 24th West Coast Conference on Formal Linguistics*, pages 34–42. Somerville, MA, Cascadilla.

Anderson, J. and Ewen, C. (1987). *Principles of Dependency Phonology*. Cambridge, Cambridge University Press.

Anderson, S. R. (1974). *The Organization of Phonology*. New York, Academic Press.

Anderson, S. R. (1981). Why phonology isn't 'natural'. *Linguistic Inquiry*, 12:493–539.

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

Anttila, A. (1997). *Deriving variation from grammar : A study of Finnish genitives*. John Benjamins.

Anttila, R. (1989). *Historical and Comparative Linguistics*. Amsterdam and Philadelphia, John Benjamins, 2nd edition.

Ao, B. (1991). Kikongo nasal harmony and context-sensitive underspecification. *Linguistic Inquiry*, 22:193–96.

Archangeli, D. (1985). Yokuts harmony: Evidence for coplanar representation in nonlinear phonology. *Linguistic Inquiry*, 16:335–372.

Archangeli, D. and Pulleyblank, D. (1987). Minimal and maximal rules: Effects of tier scansion. In *NELS 17: Proceedings of the 17th Annual Meeting of the North East Linguistic Society*, pages 16–35. GLSA, University of Massachusetts Amherst.

Archangeli, D. and Pulleyblank, D. (1994). *Grounded Phonology*. Cambridge, MA, MIT Press.

Bach, E. and Harms, R. T. (1972). How do languages get crazy rules? In Stockwell, R. P. and Macaulay, R. K., editors, *Linguistic Change and Generative Theory*, pages 1–21. Bloomington, Indiana University Press.

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

Baković, E. and Wilson, C. (2000). Transparency, strict locality, and targeted constraints. In Billerey, R. and Lillehaugen, B. D., editors, *WCCFL 19: Proceedings of the 19th West Coast Conference on Formal Linguistics*, pages 43–56. Somerville, MA, Cascadilla.

Beesley, K. R. and Karttunen, L. (2003). *Finite State Morphology*. Center for the Study of Language and Information.

Bennett, W. (2013). *Dissimilation, Consonant Harmony, and Surface Correspondence*. PhD thesis, Rutgers University.

Besnier, N. (1987). An autosegmental approach to metathesis in Rotuman. *Lingua*, 73:201–223.

Bessell, N. J. (1998). Local and non-local consonant-vowel interaction in Interior Salish. *Phonology*, 15:1–40.

Bhaskararao, P. (1980). *Koṇekor Gadaba: A Dravidian language*. Pune, Deccan College, Post-Graduate Institute.

Blake, F. R. (1917). Reduplication in Tagalog. *The American Journal of Philology*, 38(4):425–431.

Blanc, H. (1970). The Arabic dialect of the Negev Bedouins. In *Proceedings of the Israel Academy of Sciences and Humanities*, volume 4, pages 112–150.

Blevins, J. (2004). *Evolutionary Phonology*. Cambridge, Cambridge University Press.

Blevins, J. (2008). Consonant epenthesis: Natural and unnatural histories. In Good, J., editor, *Language universals and language change*, pages 79–109. Oxford, Oxford University Press.

Blevins, J. and Garrett, A. (1998). The origins of consonant-vowel metathesis. *Language*, 74(3):508–556.

Blevins, J. and Garrett, A. (2004). The evolution of metathesis. In Hayes, B., Kirchner, R., and Steriade, D., editors, *Phonetically Based Phonology*, pages 117–156. Cambridge, Cambridge University Press.

Bloch, J. (1915). *La Formation de la Langue Marathe*. Paris, E. Champion.

Boersma, P. (1997). How we learn variation, optionality, and probability. In *Proceedings of the Institute of Phonetic Sciences 21*.

Boersma, P. (1998). *Functional phonology: Formalizing the interactions between articulatory and perceptual drives*. PhD thesis, University of Amsterdam.

Boersma, P. (2000). Learning a grammar in functional phonology. In Dekkers, J., van der Leeuw, F., and van de Weijer, J., editors, *Optimality Theory: Phonology, Syntax, and Acquisition*. New York, Oxford University Press.

Boersma, P. and Hayes, B. (2001). Empirical tests of the Gradual Learning Algorithm. *Linguistic Inquiry*, 32:45–86.

Bogoras, W. (1969). Chukchee. In Boas, F., editor, *Handbook of American Indian Languages, Bureau of American Ethnology Bulletin 40, Part 2*, pages 631–903. Washington, DC, Government Printing Office.

Buckley, E. (2000). On the naturalness of unnatural rules. In *Proceedings from the 2nd Workshop on American Indigenous Languages. UCSB Working Papers in Linguistics*, volume 9.

Buckley, E. (2002). Rule naturalness and the acquisition of phonology. Talk given at the Second North American Phonology Conference, Concordia University, Montréal, April 25-28.

Buckley, E. (2003). Children's unnatural phonology. *Berkeley Linguistics Society*, 29:523–534.

Buckley, E. (2011). Metathesis. In van Oostendorp, M., Ewen, C., Hume, E., and Rice, K., editors, *The Blackwell Companion to Phonology*, volume 3. Wiley-Blackwell.

Byrd, D. (1993). Marshallese suffixal reduplication. In Mead, J., editor, *WCCFL 11: Proceedings of the 11th West Coast Conference on Formal Linguistics*.

Carpenter, A. C. (2010). A naturalness bias in learning stress. *Phonology*, 27:345–392.

Chandlee, J., Athanasopoulou, A., and Heinz, J. (2012). Evidence for classifying metathesis patterns as subsequential. In Choi, J., Hogue, E. A., Punske, J., Tat, D., Schertz, J., and Trueman, A., editors, *WCCFL 29: Proceedings of the 29th West Coast Conference on Formal Linguistics*. Somerville, MA, Cascadilla.

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 in Computational Morphology and Phonology (SIGMORPHON)*.

Chandlee, J. and Jardine, A. (2014). Learning phonological mappings by learning Strictly Local functions. In Kingston, J., Moore-Cantwell, C., Pater, J., and Staubs, R., editors, *Proceedings of the 2013 Meeting on Phonology*. LSA.

Chandlee, J. and Koirala, C. (2014). Learning local phonological rules. In *Proceedings of the 37th Penn Linguistics Colloquium*.

Charette, M. (1988). *Some Constraints on Governing Relations in Phonology*. PhD thesis, McGill University.

Charette, M. (1990). *Conditions on Phonological Government*. Cambridge, Cambridge University Press.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory 113124*, IT-2.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, 2:137–167.

Chomsky, N. (1965). *Aspects of the Theory of Syntax*. Cambridge, MA, MIT Press.

Chomsky, N. (1973). Conditions on transformations. In Anderson, S. and Kiparsky, P., editors, *A Festschrift for Morris Halle*, pages 232–286. Holt, Rinehart, and Winston, New York.

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

Churchward, C. (1940). *Rotuman Grammar and Dictionary*. Sydney, Methodist Church of Australasia, Department of Overseas Missions.

Clark, A. and Lappin, S. (2012). Computational learning theory and language acquisition. In Kempson, R., Asher, N., and Fernando, T., editors, *Philosophy of Linguistics*, pages 445–475. Elsevier.

Clements, G. N. (1976). *Vowel Harmony in Nonlinear Generative Phonology: An Autosegmental Model*. Bloomington, Indiana University Linguistics Club.

Coffman, I. (2013). Explaining long-distance liquid metathesis: Misperception vs. optimization. In Fainleib, Y., LaCara, N., and Park, Y., editors, *NELS 41: Proceedings of the 41st Annual Meeting of the North East Linguistic Society*, pages 113–126. GLSA, University of Massachusetts Amherst.

Crouch, M. (1994). The phonology of Dɛg, ms. GILLBT, Ghana.

Davidson, J. (1977). *A Contrastive Study of the Grammatical Structures of Aymara and Cuzco Quechua*. PhD thesis, University of California, Berkeley.

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

Demers, R. A. (1974). Alternating roots in Lummi. *International Journal of American Linguistics*, 40(1):15–21.

Dereau, L. (1955). *Cours de Kikongo*. Namur, A. Wesmael-Charlier.

Diver, W. (1958). On the prehistory of Greek consonantism. *Word*, 14:1–25.

Dresher, E. (1999). Charting the learning path: Cues to parameter setting. *Linguistic Inquiry*, 30:27–67.

Dresher, E. and Kaye, J. (1990). A computational learning model for metrical phonology. *Cognition*, 34:137–195.

Dubois, J. W. (1981). *The Sacapultec Language*. Ann Arbor, UMI.

Dumenil, A. (1987). A rule-account of metathesis in Gascon. *Lingvisticae Investigationes*, 11:81–113.

Eisner, J. (2000). Directional constraint evaluation in Optimality Theory. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 257–263. Morgan Kaufmann.

Elgot, C. C. and Mezei, J. E. (1965). On relations defined by generalized finite automata. *IBM Journal of Research and Development*, 9(1):47–68.

Elworthy, F. T. (1875). *The Dialect of West Somerset (English Dialect Society, Series D, Miscellaneous, 7)*. London, Trübner.

Flemming, E. (1995). Vowels undergo consonant harmony. Paper presented at the Trilateral Phonology Weekend, Stanford University.

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

Friederici, A. D. and Wessels, J. M. (1993). Phonotactic knowledge and its use in infant speech perception. *Perception & Psychophysics*, 54:287–295.

Gafos, A. (1996). *The Articulatory Basis of Locality in Phonology*. PhD thesis, Johns Hopkins University.

Gafos, A. and Lombardi, L. (1999). Consonant transparency and vowel echo. In Tamanji, P., Hirotani, M., and Hall, N., editors, *NELS 29: Proceedings of the 29th Annual Meeting of the North East Linguistic Society*, pages 81–96. GLSA, University of Massachusetts Amherst.

Gainor, B., Lai, R., and Heinz, J. (2012). Computational characterizations of vowel harmony patterns and pathologies. In Choi, J., Hogue, E. A., Punske, J., Tat, D., Schertz, J., and Trueman, A., editors, *WCCFL 29: Proceedings of the 29th West Coast Conference on Formal Linguistics*, pages 63–71. Somerville, MA, Cascadilla.

Gerdemann, D. and van Noord, G. (2000). Approximation and exactness in finite state Optimality Theory. In *Proceedings of the 5th Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON 2000)*.

Gildea, D. and Jurafsky, D. (1995). Automatic induction of finite state transducers for simple phonological rules. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 9–15. Association for Computational Linguistics.

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

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

Goldsmith, J. (1976). *Autosegmental Phonology*. PhD thesis, MIT.

Goldsmith, J. (1979). *Autosegmental Phonology*. New York, Garland.

Grammont, M. (1905-1906). La métathèse dans le parler de Bagnéres-de-Luchon. *Mémoires de la Société de Linguistique de Paris*, 13:73–90.

Grammont, M. (1948). *Phonétique du Grec Ancien*. Lyon, IAC.

Halle, M. (1959). Questions of linguistics. *Nuovo Cimento*, 13:494–517.

Halle, M. (1961). On the role of simplicity in linguistic description. In Jakobson, R., editor, *Structure of Language and Its Mathematical Aspects, Proceedings of the 12th Symposium in Applied Mathematics*, pages 89–94. American Mathematical Society.

Halle, M. (1962). Phonology in generative grammar. *Word*, 18:54–72.

Halle, M. (1964). On the bases of phonology. In Fodor, J. and Katz, J. J., editors, *The Structure of Language: Readings in the Philosophy of Language*. Englewood Cliffs, NJ, Prentice-Hall.

Halle, M. and Vergnaud, J.-R. (1978). Metrical structures in phonology. ms., MIT.

Hansson, G. (2001). *Theoretical and Typological Issues in Consonant Harmony*. PhD thesis, University of California, Berkeley.

Harkins, W. E. (1953). *A Modern Czech Grammar*. New York, King's Crown Press.

Harris, J. (1990). Segmental complexity and phonological government. *Phonology*, 7:255–300.

Harris, J. and Halle, M. (2005). Unexpected plural inflections in Spanish: Reduplication and metathesis. *Linguistic Inquiry*, 36(2):195–222.

Hayes, B. (2004). Phonological acquisition in Optimality Theory: the early stages. In Kager, R., Pater, J., and Zonneveld, W., editors, *Fixing Priorities: Constraints in Phonological Acquisition*. Cambridge University Press.

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

Hayes, B. and White, J. (2013). Phonological naturalness and phonotactic learning. *Linguistic Inquiry*, 44:45–75.

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

Hayes, B., Zuraw, K., Siptár, P., and Londe, Z. (2009). Natural and unnatural constraints in Hungarian vowel harmony. *Language*, 85(4):822–863.

Hayward, D. (1984). *The Arbore Language: A First Investigation, Including a Vocabulary*. Hamburg, Helmut Buske Verlag.

Healy, A. F. and Levitt, A. G. (1980). Accessibility of the voicing distinction for learning phonological rules. *Memory and Cognition*, 8(2):107–114.

Heine, B. (1980). *The Non-Bantu Languages of Kenya*. Berlin, Dietrich Reimer Verlag.

Heinz, J. (2005a). Optional partial metathesis in Kwara'ae. In *Proceedings of AFLA 12*, pages 91–102.

Heinz, J. (2005b). Reconsidering linearity: Evidence from CV metathesis. In Alderete, J., Han, C.-H., and Kochetov, A., editors, *WCCFL 24: Proceedings of the 24th West Coast Conference on Formal Linguistics*, pages 200–208. Somerville, MA, Cascadilla.

Heinz, J. (2007). *The Inductive Learning of Phonotactic Patterns*. PhD thesis, University of California, Los Angeles.

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

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

Heinz, J. (2013). The typology of phonological generalizations: A computational perspective. Talk given at Meeting on Phonological Typology, Oxford University, Somerville College, Oxford, United Kingdom, August 2013.

Heinz, J. (2014). (to appear) Computational theories of learning and developmental psycholinguistics. In Lidz, J., Synder, W., and Pater, J., editors, *The Cambridge Handbook of Developmental Linguistics*. Cambridge, Cambridge University Press.

Heinz, J. and Idsardi, W. (2011). Sentence and word complexity. *Science*, 333(6040):295–297.

Heinz, J. and Koirala, C. (2010). Maximum likelihood estimation of feature-based distributions. In *Proceedings of the 11th Meeting of the ACL Special Interest Group in Computational Morphology and Phonology (SIGMORPHON)*, pages 28–37. Association for Computational Linguistics.

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.

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. 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.

Hellberg, S. (1978). Unnatural phonology. *Journal of Linguistics*, 14:157–177.

Herrity, P. (2000). *Slovene: A comprehensive grammar*. London and New York: Routledge.

Hinton, L. and Langdon, M. (1976). Object-subject pronominal prefixes in La Huerta Diegueño. In Langdon, M. and Silver, S., editors, *Hokan Studies: Papers from the 1st Conference on Hokan Languages, held in San Diego, CA, April 23-25, 1970*, pages 113–128. Mouton.

Hock, H. H. (1985). Regular metathesis. *Linguistics*, 23:529–546.

Hoijer, H. (1933). *Tonkawa: An Indian language of Texas*. New York, Handbook of American Indian Languages 3.

Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2001). *Introduction to Automata Theory, Languages, and Computation*. Boston, Addison Wesley, 2nd edition.

Horowitz, E. (1960). *How the Hebrew Language Grew.* Jewish Educational Committee Press, New York.

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

Hume, E. (1991). Metathesis in Maltese: Implications for the Strong Morphemic Plane Hypothesis. In Sherer, T., editor, *NELS 21: Proceedings of the 21st Annual Meeting of the North East Linguistic Society*, pages 157–171. GLSA, University of Massachusetts Amherst.

Hume, E. (1995). Beyond linear order: Prosodic constraints and CV metathesis. In Gabriele, L., Hardison, D., and Westmoreland, R., editors, *FLSM VI: Papers from the 6th Annual Meeting of the Formal Linguistics Society of Mid-America*, volume 1, pages 15–26. Indiana University Linguistics Club.

Hume, E. (1998). Metathesis in phonological theory: the case of Leti. *Lingua*, 10.

Hume, E. (2000). Metathesis website. http://www.ling.ohio-state.edu/ ehume/metathesis/.

Hume, E. (2001). Metathesis: formal and functional considerations. In *Surface Syllable Structure and Segment Sequencing*, pages 1–25. Leiden, NL, HIL.

Hume, E. and Johnson, K. (2001). *The Role of Speech Perception in Phonology.* San Diego, Academic Press.

Hyman, L. (1970). How concrete is phonology? *Language*, 46:58–76.

Janda, R. D. and Joseph, B. D. (1989). In further defense of a non-phonological account of Sanskrit root-initial aspiration alternations. In *ESCOL 88. Proceedings of the 5th Eastern States Conference on Linguistics*, pages 246–260. The Ohio State University Department of Linguistics.

Jardine, A. (2013). Tone is (computationally) different. ms., University of Delaware.

Jarosz, G. (2006a). *Rich Lexicons and Restrictive Grammars - Maximum Likelihood Learning in Optimality Theory.* PhD thesis, Johns Hopkins University.

Jarosz, G. (2006b). Richness of the Base and probabilistic unsupervised learning in Optimality Theory. In *Proceedings of the 8th Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON).* Association for Computational Linguistics.

Jarosz, G. (2007). Stages of acquisition without ranking biases: the roles of frequency and markedness in phonological learning. In Becker, M., editor, *UMass Occasional Papers in Linguistics.*

Jarosz, G. (2009a). Naive parameter learning for Optimality Theory - the hidden structure problem. In *NELS 40: Proceedings of the 40th Annual Meeting of the North East Linguistic Society*. GLSA, University of Massachusetts Amherst.

Jarosz, G. (2009b). Restrictiveness in phonological grammar and lexicon learning. In *Proceedings of the 43rd Annual Meeting of the Chicago Linguistic Society*.

Jarosz, G. (2010). Implicational markedness and frequency in constraint-based computational models of phonological learning. *Journal of Child Language*, 37(3):565–606.

Jarosz, G. (2011). The roles of phonotactics and frequency in the learning of alternations. In *Proceedings of the 35th Annual Meeting of the Boston University Conference on Language Development*.

Jarosz, G. (2013). Learning with hidden structure in Optimality Theory and Harmonic Grammar: Beyond robust interpretive parsing. *Phonology*, 30(1):27–71.

Jensen, J. T. (1974). A constraint on variables in phonology. *Language*, 50(4):675–686.

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

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

Johnson, R. E. (1975). *The Role of Phonetic Detail in Coeur d'Alene Phonology*. Ann Arbor, UMI.

Joseph, B. D. and Philippaki-Warburton, I. (1987). *Modern Greek*. Wolfeboro, NH, Croom Helm.

Jusczyk, P., Cutler, A., and Redanz, N. (1993a). Infants preference for the predominant stress patterns of English words. *Child Development*, 64:675–687.

Jusczyk, P., Friederici, A. D., Wessels, J. M. I., Svenkerud, V. Y., and Jusczyk, M. (1993b). Infants' sensitivity to the sound patterns of native language words. *Journal of Memory and Language*, 32(3):402–420.

Jusczyk, P., Luce, P., and Charles-Luce, J. (1994). Infants sensitivity to phonotactic patterns in the native language. *Journal of Memory and Language*, 33:630–645.

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

Kahn, M. (1976). How abstract is pharyngology: Evidence from Kurmanji. In Mufwene, S. S., Walker, C. A., and Steever, S. B., editors, *Papers from the 12th Regional Meeting of the Chicago Linguistic Society, April 23-25, 1976*, pages 313–20. Chicago Linguistic Society.

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

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

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

Kaye, J., Lowenstamm, J., and Vergnaud, J.-R. (1985). The internal structure of phonological representations: A theory of charm and government. *Phonology Yearbook*, 2:305–328.

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

Kenstowicz, M. (1979). Chukchee vowel harmony and epenthesis. In Clyne, P., Hanks, W., and Hofbauer, C., editors, *Proceedings of CLS 15: The Elements: Parasession on Linguistic Units and Level*, pages 402–412. Chicago, Chicago Linguistic Society.

Kenstowicz, M. (1994). *Phonology in Generative Grammar*. Blackwell.

Kenstowicz, M. and Kisseberth, C. (1977). *Topics in Phonological Theory*. Academic Press.

Kerkhoff, A. (2004). Acquisition of voicing alternations. In Baauw, S. and van Kampen, J., editors, *Proceedings of GALA 2003*, volume 2, pages 269–280. LOT.

Kiparsky, P. (1968). *How abstract is phonology?* Bloomington: Indiana University Linguistics Club.

Kiparsky, P. (1976). Sonorant clusters in Greek. *Language*, 43(3):619–635.

Kiparsky, P. (1981). Vowel harmony. ms., MIT.

Klein, T. B. (2005). Infixation and segmental constraint effects: *um* and *in* in Tagalog, Chamorro, and Toba Batak. *Lingua*, 115(7):959–995.

Kobele, G. (2006). *Generating Copies: An Investigation into Structural Identity in Language and Grammar*. PhD thesis, UCLA.

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

Krause, S. (1980). *Topics in Chukchee phonology and morphology*. PhD thesis, University of Illinois, Urbana-Champaign.

Krishnamurti, B. (1955). The history of vowel-length in Telugu verbal bases. *Journal of the American Oriental Society*, 75:237–52.

Krishnamurti, B. (1978). Areal and lexical diffusion of sound change: Evidence from Dravidian. *Language*, 54:1–20.

Kuipers, A. H. (1974). *The Shuswap Language*. The Hague, Mouton.

Lai, R. (2012). *Domain Specificity in Phonology*. PhD thesis, University of Delaware.

Lai, R. (2014). Learnable vs. unlearnable harmony patterns. *Linguistic Inquiry, to appear*.

Langdon, M. (1976). Metathesis in Yuman languages. *Language*, 52:866–83.

Legendre, G., Miyata, Y., and Smolensky, P. (1990). Harmonic grammar: A formal multi-level connectionist theory of linguistic well-formedness: Theoretical foundations. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, pages 388–395. Lawrence Erlbaum Associates.

Lejeune, M. (1955). *Traité de phonétique Grecque*. Paris, Klincksieck, 2nd edition.

Lejeune, M. (1972). *Phonétique historique de Mycénien et du Grec Ancien*. Paris, Klincksieck.

Liberman, M. (1975). *The Intonational System of English*. PhD thesis, MIT.

Liberman, M. and Prince, A. (1977). On stress and linguistic rhythm. *Linguistic Inquiry*, 8:249–336.

Lipski, J. M. (1990). Metathesis as template matching: A case study from Spanish. *Folia Linguistica Historica*, 11:89–104.

Luo, H. (2013). Long-distance consonant harmony and subsequentiality. ms., University of Delaware.

Magri, G. (2010). Complexity of the acquisition of phonotactics in Optimality Theory. In Heinz, J., Cahill, L., and Wicentowski, R., editors, *Proceedings of the 11th meeting of the ACL Special Interest Group in Computational Morphology and Phonology (SIGMORPHON)*, pages 19–27. Association for Computational Linguistics.

Magri, G. (2011). An online model of the acquisition of phonotactics within Optimality Theory. In L.Carlson, C.Hölscher, and Shipley, T., editors, *Proceedings of the 33rd annual conference of the Cognitive Science Society*. Cognitive Science Society.

Magri, G. (2012). An approximation approach to the problem of the acquisition of phonotactics in Optimality Theory. In Cahill, L. and Albright, A., editors, *Proceedings of the 12th meeting of the ACL Special Interest Group in Computational Morphology and Phonology (SIGMORPHON)*, pages 52–61. Association for Computational Linguistics.

Malone, J. (1971). Systematic metathesis in Mandaic. *Language*, 47:394–415.

Malone, J. L. (1985). Classical mandaic radical metathesis, radical assimilation, and the devil's advocate. *General Linguistics*, 25(2):92–122.

Marantz, A. (1982). Re reduplication. *Linguistic Inquiry*, 13(3):435–482.

Mascaró, J. (1976). *Catalan Phonology and the Phonological Cycle*. PhD thesis, MIT.

Matras, Y. (2002). *Romani: A linguistic introduction*. Cambridge, Cambridge University Press.

Mattina, A. (1979). Pharyngeal movement in Colville and related phenomena in the Interior Salish languages. *International Journal of American Linguistics*, 45:17–24.

McCarthy, J. J. (1979). *Formal Problems in Semitic Phonology and Morphology*. PhD thesis, MIT.

McCarthy, J. J. (1989). Linear order in phonological representations. *Linguistic Inquiry*, 20:71–100.

McCarthy, J. J. (2000a). 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. (2000b). The prosody of phase in Rotuman. *Natural Language & Linguistic Theory*, 18(1):147–197.

McCarthy, J. J. (2008). *Doing Optimality Theory*. Malden, MA: Wiley-Blackwell.

McCarthy, J. J. and Prince, A. (1995). Faithfulness and reduplicative identity. In Beckman, J., Dickey, L. W., and Urbanczyk, S., editors, *University of Massachusetts Occasional Papers in Linguistics [UMOP] 18: Papers in Optimality Theory*, pages 249–384. GLSA University of Massachusetts Amherst.

McCarthy, J. J. and Prince, A. (1996). Prosodic morphology 1986. Technical Report 32, Rutgers University Center for Cognitive Science.

McCarthy, J. J. and Prince, A. (1999). Faithfulness and identity in prosodic morphology. In Kager, R., van der Hulst, H., and Zonneveld, W., editors, *The Prosody Morphology Interface*, pages 218–309. Cambridge, Cambridge University Press.

McNaughton, R. and Papert, S. (1971). *Counter-Free Automata*. Cambridge, MA, MIT Press.

Meinhof, C. (1932). *Introduction to the phonology of the Bantu languages*. Berlin: Dietrich Reimer/Ernst Vohsen.

Merchant, N. and Tesar, B. (2008). Learning underlying forms by searching restricted lexical subspaces. In *Proceedings of CLS 41*.

Mielke, J. (2004). *P-Base 1.95*. http://137.122.133.199/ Jeff/pbase/.

Mielke, J. and Hume, E. (2001). Consequences of word recognition for metathesis. In Hume, E., Smith, N., and van de Weijer, J., editors, *Surface Syllable Structure and Segment Sequencing*, pages 135–158. Leiden: HIL.

Mills, R. F. and Grima, J. (1980). Historical developments in Lettinese. In Naylor, P. B., editor, *Austronesian studies: Papers from the 2nd Eastern Conference on Austronesian Languages*, pages 273–83. Ann Arbor, The University of Michigan Center for South and Southeast Asian Studies.

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

Nelson, N. (2005). Wrong side reduplication is epiphenomenal: evidence from Yoruba. In Hurch, B., editor, *Studies on Reduplication*, pages 135–160. Berlin, Mouton de Gruyter.

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

Newman, S. (1944). *Yokuts language of California*. New York, Viking Fund Publications in Anthropology.

Ní Chiosáin, M. and Padgett, J. (1997). Markedness, segment realization, and locality in spreading. Technical report. Report no. LRC-97-01, Linguistics Research Center, UC Santa Cruz, CA.

Ní Chiosáin, M. and Padgett, J. (2001). Markedness, segment realization, and locality in spreading. In Lombardi, L., editor, *Segmental phonology in Optimality Theory*, pages 118–156. Cambridge, Cambridge University Press.

Noonan, M. (1997). Inverted roots in Salish. *International Journal of American Linguistics*, 63:475–515.

Odden, D. (1994). Adjacency parameters in phonology. *Language*, 70(2):289–330.

Odden, D. (2007). The unnatural tonology of Zina Kotoko. In Riad, T. and Gussenhoven, C., editors, *Tones and Tunes, Vol. 1: Typological studies in word and sentence prosody*, pages 63–89. Berlin, Mouton de Gruyter.

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

Orgun, C. O. and Sprouse, R. L. (1999). From mparse to control: Deriving ungrammaticality. *Phonology*, 16:191–224.

Oswalt, R. L. (1961). *A Kashaya Grammar (Southwestern Pomo)*. PhD thesis, UC Berkeley.

Oswalt, R. L. (1976). Comparative verb morphology of Pomo. In Langdon, M. and Silver, S., editors, *Hokan Studies*, pages 13–28. Mouton.

Paradis, C. and Prunet, J.-F. (1989). On coronal transparency. *Phonology*, 6:317–348.

Paradis, C. and Prunet, J.-F. (1991). *The special status of coronals: Internal and external evidence*. San Diego, Academic Press.

Pater, J. (2004). Exceptions and Optimality Theory: Typology and learnability. In *Conference on Redefining Elicitation: Novel Data in Phonological Theory*. New York University.

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.

Pater, J. and Tessier, A. M. (2003). Phonotactic knowledge and the acquisition of alternations. In *Proceedings of the 15th International Congress on Phonetic Sciences*, pages 1777–1180.

Payne, A. (2013). Dissimilation as a subsequential process. ms., University of Delaware.

Penny, R. (1991). *A History of the Spanish Language*. Cambridge, Cambridge University Press.

Peperkamp, S., Skoruppa, K., and Dupoux, E. (2006). The role of phonetic naturalness in phonological rule acquisition. In Bamman, D., Magnitskaia, T., and Zaller, C., editors, *Proceedings of the 30th Annual Boston University Conference on Language Development*, pages 464–475. Somerville, MA: Cascadilla.

Piggott, G. L. (1996). Implications of consonant nasalization for a theory of harmony. *Canadian Journal of Linguistics*, 41:141–74.

Prince, A. and Smolensky, P. (1991). Notes on connectionism and Harmony Theory in linguistics. Technical Report CU-CS-533-91, Department of Computer Science, University of Colorado, Boulder.

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

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

Prince, A. and Tesar, B. (1999). Learning phonotactic distributions. Technical Report RuCCS-TR-54, Rutgers Center for Cognitive Science, Rutgers University.

Prince, A. and Tesar, B. (2004). Learning phonotactic distributions. In Kager, R., Pater, J., and Zonneveld, W., editors, *Fixing Priorities: Constraints in Phonological Acquisition*, pages 245–291. Cambridge, Cambridge University Press.

Pulleyblank, D. and Turkel, W. J. (1995). The acquisition of variable constraint rankings by a genetic algorithm. Paper presented at the 2nd Workshop on Processing Consequences of Contrasting Language Phonologies, Scuola Internazionale Superiore di Studi Avanzati, Trieste.

Pulleyblank, D. and Turkel, W. J. (1996). Optimality Theory and learning algorithms: the representation of recurrent featural asymmetries. In Durand, J. and Laks, B., editors, *Current Trends in Phonology: Models and Methods*, volume 2, pages 653–684. University of Salford Publications.

Pulleyblank, D. and Turkel, W. J. (1998). The logical problem of language acquisition in Optimality Theory. In Barbosa, P., Fox, D., Hagstrom, P., McGinnis, M., and Pesetsky, D., editors, *Is the Best Good Enough?: Optimality and Competition in Syntax*, pages 399–420. Cambridge, MA, MIT Press.

Pulleyblank, D. and Turkel, W. J. (2000). Learning phonology: Genetic algorithms and Yoruba tongue-root harmony. In Dekkers, J., van der Leeuw, F., and van de Weijer, J., editors, *Optimality Theory: Phonology, Syntax, and Acquisition*, pages 554–591. Oxford, Oxford University Press.

Pycha, A., Nowak, P., Shin, E., and Shosted, R. (2003). Phonological rule-learning and its implications for a theory of vowel harmony. In Garding, G. and Tsujimura, M., editors, *WCCFL 22: Proceedings of the 22nd West Coast Conference on Formal Linguistics*, pages 423–435. Somerville, MA, Cascadilla.

Raimy, E. (1999). *Representing Reduplication*. PhD thesis, University of Delaware.

Raimy, E. (2000). *The Phonology and Morphology of Reduplication*. Mouton de Gruyter.

Ray, P. S. (1966). *Bengali Language Handbook*. Washington, D.C., Center for Applied Linguistics.

Riggle, J. (2003). Non-local reduplication. In *NELS 34: Proceedings of the 34th Annual Meeting of the North East Linguistic Society*. GLSA, University of Massachusetts Amherst.

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

Riggle, J. (2006a). Infixing reduplication in Pima and its theoretical consequences. *Natural Language & Linguistic Theory*, 24:857–891.

Riggle, J. (2006b). Using entropy to learn OT grammars from surface forms alone. In Baumer, D., Montero, D., and Scanlon, M., editors, *WCCFL 25: Proceedings of the 25th West Coast Conference on Formal Linguistics*. Somerville, MA, Cascadilla.

Riggle, J. (2010). Sampling rankings. ms., University of Chicago.

Roark, B. and Sproat, R. (2007). *Computational Approaches to Morphology and Syntax*. Oxford, Oxford University Press.

Roche, E. and Schabes, Y. (1997). *Finite-State Language Processing*. Cambridge, MA, MIT Press.

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

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

Rohlfs, G. (1950). *Historische grammatik der unteritalienischen Gräzität*. München, Verlag der Bayerischen Akademie der Wissenchaften.

Rose, S. and Walker, R. (2004). A typology of consonant agreement as correspondence. *Language*, 80:475–531.

Sapir, E. and Hoijer, H. (1967). *The phonology and morphology of the Navaho language*. Berkeley, University of California Press.

Schane, S. A., Tranel, B., and Lane, H. (1974). On the psychological reality of a natural rule of syllable structure. *Cognition*, 3(4):351–358.

Schieber, S. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

Schwyzer, E. (1953). *Griechische Grammatik I.* München, Beck, 2nd edition.

Seidl, A. and Buckley, E. (2005). On the learning of arbitrary phonological rules. *Language Learning and Development*, 1:289–316.

Seo, M. and Hume, E. (2001). A comparative OT account of metathesis in Faroese and Lithuanian. In Hume, E., Smith, N., and van de Weijer, J., editors, *Surface syllable structure and segment sequencing*. Leiden, NL, HIL.

Shaw, P. (1991). Consonant harmony systems: The special status of coronal harmony. In Paradis, C. and Prunet, J.-F., editors, *The Special Status of Coronals*, pages 125–157. New York, Academic Press.

Sneddon, J. (1996). *Indonesian: a comprehensive grammar*. Routledge.

Steinhauer, H. (1996). Synchronic metathesis and apocope in three austronesian languages of the Timor area. In *The 4th International Symposium on Language and Linguistics*, pages 471–493. Institute of Language and Culture for Rural Development, Mahidol University.

Steriade, D. (1982). *Greek Prosodies and the Nature of Syllabification*. PhD thesis, MIT.

Steriade, D. (1987). Locality conditions and feature geometry. In *NELS 17: Proceedings of the 17th Annual Meeting of the North East Linguistic Society*, pages 595–617. GLSA, University of Massachusetts Amherst.

Subrahmanyam, P. S. (1983). *Dravidian Comparative Phonology*. Annamalainagar, Annamalai University.

Suzuki, K. (1998). *A Typological Investigation of Dissimilation*. PhD thesis, University of Arizona.

Tesar, B. (1995). *Computational Optimality Theory*. PhD thesis, University of Colorado, Boulder.

Tesar, B. (1998a). Error-driven learning in Optimality Theory via the efficient computation of optimal forms. In Barbosa, P., Fox, D., Hagstrom, P., McGinnis, M., and Pesetsky, D., editors, *Is the Best Good Enough?: Optimality and Competition in Syntax*, pages 421–435. Cambridge, MA, MIT Press.

Tesar, B. (1998b). An iterative strategy for language learning. *Lingua*, 104:131–145.

Tesar, B. (2002). Enforcing grammatical restrictiveness can help resolve structural ambiguity. In Mikkelsen, L. and Potts, C., editors, *WCCFL 21: Proceedings of the 21st West Coast Conference on Formal Linguistics*, pages 443–456. Somerville, MA, Cascadilla.

Tesar, B. (2003). Learnability. In Frawley, W., editor, *The Oxford International Encyclopedia of Linguistics*. Oxford, Oxford University Press, 2nd edition.

Tesar, B. (2004). Using inconsistency detection to overcome structural ambiguity. *Linguistic Inquiry*, 35(2):219–253.

Tesar, B. (2008). Output-driven maps. ms., Rutgers University.

Tesar, B. (2014). *Output-Driven Phonology: Theory and Learning*. Cambridge, Cambridge University Press.

Tesar, B., Alderete, J., Horwood, G., Merchant, N., Nishitani, K., and Prince, A. (2003). Surgery in language learning. In Garding, G. and Tsujimura, M., editors, *WCCFL 22: Proceedings of the 22nd West Coast Conference on Formal Linguistics*, pages 477–490. Somerville, MA, Cascadilla.

Tesar, B. and Prince, A. (2003). Using phonotactics to learn phonological alternations. In *Proceedings of CLS 39, Vol. II: The Panels*.

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

Tesar, B. and Smolensky, P. (1996). Learnability in Optimality Theory (short version). Technical Report JHU-CogSci-96-2, Cognitive Science Department, The Johns Hopkins University.

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

Tesar, B. and Smolensky, P. (2000). *Learnability in Optimality Theory*. Cambridge, MA, MIT Press.

Touretzky, D. S., III, G. E., and Wheeler, D. W. (1990). Phonological rule induction: An architectural solution. Technical Report AIP-118, Carnegie Mellon University.

Turner, R. L. (1969). *A Comparative Dictionary of the Indo-Aryan Languages*. London, Oxford University Press.

Ultan, R. (1978). A typological view of metathesis. In Greenberg, J. H., editor, *Universals of human language, Vol. 2, Phonology*, pages 367–402. Stanford, CA, Stanford University Press.

Valiant, L. (1984). A theory of the learnable. *Communications of the ACM*, 27.

van der Hulst, H. and van Engelenhoven, A. (1995). Metathesis effects in Tutukeian-Letinese. In van der Hulst, H. and van de Weijer, J., editors, *Leiden in Last. HIL phonology papers I. HIL Publications 1*, pages 243–267. The Hague, Holland Academic Graphics.

Vennemann, g. N. T. (1988). *Preference Laws for Syllable Structure and the Explanation of Sound Change*. Berlin, Mouton de Gruyter.

Vennemann, g. N. T. (1996). The development of reduplicating verbs in Germanic. In Rauch, I. and Carr, G. F., editors, *Insights in Germanic linguistics II: Classic and contemporary*, pages 297–336. Berlin, Mouton de Gruyter.

Walker, R. (1998). *Nasalization, Neutral Segments, and Opacity Effects*. PhD thesis, University of California, Santa Cruz.

Wanner, D. (1989). On metathesis in diachrony. *Papers from the 25th Annual Regional Meeting of the Chicago Linguistic Society*, pages 434–450.

Warner, N., Jongman, A., Cutler, A., and Mücke, D. (2001). The phonological status of Dutch epenthetic schwa. *Phonology*, 18:387–420.

Webb, N. (1965). Phonology and noun morphology of the Kindibu dialect of Kikongo. Master's thesis, UCLA.

Wheeler, M. (1979). *Phonology of Catalan*. Oxford, Basil Blackwell.

Whitney, W. D. (1896). *A Sanskrit Grammar*. Leipzig and Boston, Breitkopf and Härtel and Ginn and Company, 3rd edition.

Wichmann, S. (1995). *The Relationship Among the Mixe-Zoquean Languages of Mexico*. Salt Lake City: University of Utah Press.

Williamson, K. (1965). *A Grammar of the Kolokuma Dialect of Ijo*. Cambridge, Cambridge University Press.

Wilson, C. (2003). Experimental investigation of phonological naturalness. In Garding, G. and Tsujimura, M., editors, *WCCFL 22: Proceedings of the 22nd West Coast Conference for Formal Linguistics*, pages 533–546. Somerville, MA, Cascadilla.

Wilson, C. (2006). Learning phonology with substantive bias: An experimental and computational study of velar palatalization. *Cognitive Science*, 30:945–982.

Winfield, W. W. (1928). *A Grammar of the Kui Language*. Calcutta, The Asiatic Society of Bengal.

Wonderly, W. L. (1951). Zoque II: Phonemes and morphophonemes. *International Journal of American Linguistics*, 17(2):105–123.

Zorc, D. P. (1977). *The Bisayan dialects of the Philippines: Subgrouping and reconstruction. Pacific Linguistics, Series C, 44*. Canberra, Department of Linguistics, Research School of Pacific Studies, The Australian National University.

Zuraw, K. (2002). Aggressive reduplication. *Phonology*, 19(3):395–439.

Appendix

FORMAL CHARACTERIZATIONS AND PROPERTIES OF STRICTLY LOCAL FUNCTIONS

This appendix formalizes the construction of SL FSTs and presents a few additional properties of SL functions. §A repeats the relevant mathematical notations from the end of Chapter 2. §§A and (1) prove that the language-theoretic and automata-theoretic characterizations of ISL and OSL functions, respectively, are equivalent. These sections also present the formal construction of $\mathcal{T}_{ISL}(f)$ and $\mathcal{T}_{OSL}(f)$, respectively, for a given phonological rule. §(1) addresses the special case of word-boundary processes, and §(1) provides the minimization procedure for SL FSTs. Lastly, §(2) presents the properties of SL functions that are known so far.[1]

Preliminaries

An *alphabet*, Σ, is a finite set of symbols. The set of all possible strings of symbols in Σ is Σ^*, and the set of all possible strings of length n and of length up to and including n is Σ^n and $\Sigma^{\leq n}$, respectively. The empty string containing zero symbols is represented with λ. The length of a string w is the number of symbols it contains and is designated $|w|$ (so $|\lambda| = 0$).

If $w = ps$ for some $p, s \in \Sigma^*$, then p is a *prefix* of w and s is a *suffix* of w. The set of prefixes of w, $\texttt{Pref}(w) = \{p \in \Sigma^* \mid \exists s \in \Sigma^* \ [w = ps]\}$, and the set of suffixes of $w, \texttt{Suff}(w) = \{s \in \Sigma^* \mid \exists p \in \Sigma^* \ [w = ps]\}$. The unique suffix of w of length n is $\texttt{Suff}^n(w)$. If $|w| < n, \texttt{Suff}^n(w) = w$. If $w = ps$, then $ws^{-1} = p$ and $p^{-1}w = s$. The longest string in a set S, $\texttt{max}(S)$, is $s \in S$ such that $\forall s' \in S, |s'| < |s|$. The longest common prefix of a set of strings S, $\texttt{lcp}(S)$, is then $\texttt{max}(\cap_{w \in S}\texttt{Pref}(w))$.

[1] The proofs in this appendix were developed in collaboration with Jeffrey Heinz and Rémi Eyraud.

A *language* $\mathcal{L}$ is a subset of Σ^* ($\mathcal{L} \subseteq \Sigma^*$). A relation $\mathcal{R}$ is a subset of the cross-product of two languages: $\mathcal{R} \subseteq \mathcal{L}_1 \times \mathcal{L}_2 = \{(w_1, w_2) \mid w_1 \in \mathcal{L}_1 \text{ and } w_2 \in \mathcal{L}_2\}$. The *left projection* of a relation $\mathcal{R}$ is $\psi_1 = \{w_1 \mid \exists w_2 \, [(w_1, w_2) \in \mathcal{R}]\}$, and the *right projection* of $\mathcal{R}$ is $\psi_2 = \{w_2 \mid \exists w_1 \, [(w_1, w_2) \in \mathcal{R}]\}$. A relation f is a *function* if $\forall w_1 \in \psi_1$ there is exactly one $w_2 \in \psi_2$.

Definition 8 (FST)**.** *(Oncina et al., 1993) A* finite state transducer (FST) *is a six-tuple* $\tau = \{Q, \Sigma, \Gamma, q_0, F, E\}$*, where* Q *is a set of states,* Σ *and* Γ *are the input and output alphabets, respectively,* q_0 *is the initial state,* $F \subseteq Q$ *are final states, and* $E \subseteq (Q \times \Sigma \times \Gamma^* \times Q)$ *is a set of* edges*.*

The set of edges E of a FST is associated with a transition function δ: $\forall (q, a, o, q') \in E$, $\delta(q, a) = (o, q')$. Two auxiliary functions that specify only the output string and destination state of a given transition will also be useful. They are defined as follows: $\forall (q, a, o, q') \in E$, $\delta_1(q, a) = o$, and $\delta_2(q, a) = q'$.

Definition 9 (SFST)**.** *A* subsequential finite state transducer (SFST) *is a six-tuple* $\tau = \{Q, \Sigma, \Gamma, q_0, E, \sigma\}$*, where* Q *is again a finite set of states, all of which are final, and* q_0, Σ, Γ*, and* E *are defined as for FSTs. The* final output function $\sigma : Q \mapsto \Gamma^*$ *maps each state to a (possibly empty) string that is appended to the output of any input string that ends in that state.*

SFSTs are deterministic on the input, meaning $(q, a, u, r), (q, a, v, s) \in E \Rightarrow (u = v \wedge r = s)$. A *path* in a SFST is a sequence of edges in E, $\pi = (q_0, a_1, o_1, q_1)...$ (q_{n-1}, a_n, o_n, q_n), $q_i \in Q, a_i \in \Sigma, o_i \in \Gamma^*, 1 \leq i \leq n$, which can also be condensed to $(q_0, a_1...a_n, o_1...o_n, q_n)$. Let Π_τ be the set of all possible paths in τ. The function f realized by τ is then $f(w_1) = w_2\sigma(q)$ such that $(q_0, w_1, w_2, q) \in \Pi_\tau$ (Oncina et al., 1993).

In the canonical form of a SFST, the states correspond to the distinct sets of $\texttt{tails}_f(x)$ for $x \in \Sigma^*$, with $\texttt{tails}_f(x) = \{(y, v) \mid f(xy) = uv \wedge u = \texttt{lcp}(f(x\Sigma^*))\}$ (see Chapter 3 for a more detailed explanation of tails). This set is empty if $x \notin$

$\cap_{w \in \psi_1} \texttt{Pref}(w)$. If two strings $x_1, x_2 \in \Sigma^*$ have the same set of tails with respect to a function f, we write $x_1 \sim_f x_2$.

ISL Functions

Definition 10 (Input Strictly Local Function)**.** *A function f is Input Strictly Local iff $\exists k \in \mathbb{N}$ such that for all $u_1, u_2 \in \Sigma^*$, it is the case that if $\texttt{Suff}^{k-1}(u_1) = \texttt{Suff}^{k-1}(u_2)$ then $\texttt{tails}_f(u_1) = \texttt{tails}_f(u_2)$.*

Theorem 5. *A function f is Input Strictly Local iff $\exists k \in \mathbb{N}$ such that f can be described with a SFST $\mathcal{T}_{ISL}(f)$ for which*

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

Proof. ($\Rightarrow$) Consider any f such that there is a k such that for all $u_1, u_2 \in \Sigma^*$ it is the case that if $\texttt{Suff}^{k-1}(u_1) = \texttt{Suff}^{k-1}(u_2)$ then $\texttt{tails}_f(u_1) = \texttt{tails}_f(u_2)$. We show that $\mathcal{T}_{ISL}(f)$ exists. Since $Q = \Sigma^{\leq k-1}$ is a finite set, the equivalence relation $\sim_f$ partitions Σ^* into at most $|Q|$ blocks. Hence f is subsequential and a canonical subsequential transducer $\mathcal{T}(f) = \{Q_\mathcal{T}, \Sigma, \Gamma, q_0, E_\mathcal{T}, \sigma_\mathcal{T}\}$ exists.

We construct $\mathcal{T}'$ recognizing f from $\mathcal{T}(f)$ using a mapping h from the states of $\mathcal{T}'$ to $\mathcal{T}(f)$: $\forall w \in \Sigma^{k-1}$, $h(w) = \texttt{tails}_f(w)$. Construct $\mathcal{T}' = \{Q, \Sigma, \Gamma, q_0, E, \sigma\}$ as follows.

- $Q = \Sigma^{\leq k-1}$ and $q_0 = \lambda$
- $\forall q \in Q, \sigma(q) = \sigma_\mathcal{T}(h(q))$
- $\forall q \in Q, \forall a \in \Sigma, \forall u \in \Gamma^*$, $[(q, a, u, \texttt{Suff}^{k-1}(qa)) \in E$ iff $(h(q), a, u, h(qa)) \in E_\mathcal{T}]$

We show that $\mathcal{T}' = \mathcal{T}_{ISL}(f)$. Clearly, by its construction, the states and transitions meet the requirements of (1) and (2) in Theorem 5. The mapping h ensures that $\mathcal{T}'$ computes the same function as $\mathcal{T}(f)$ because f is k-ISL: for all $x, x' \in \Sigma^*$, if $\texttt{Suff}^{k-1}(x) = \texttt{Suff}^{k-1}(x')$ then $\texttt{tails}_f(x) = \texttt{tails}_f(x')$. As proof, we show the paths are in one-to-one correspondence between the two machines. Consider any $x = x_1 \cdots x_n$, with $\forall i$ $x_i \in \Sigma$. It follows that $f(x) = yz$ such that

$(\texttt{tails}_f(\lambda), x, y, \texttt{tails}_f(x)) \in \Pi_{\mathcal{T}}$ and $\sigma_{\mathcal{T}}(\texttt{tails}_f(x)) = z$. Furthermore, there is a path in $\mathcal{T}(f)$ equal to the following sequence of edges:

$$(\texttt{tails}_f(\lambda), x_1, y_1, \texttt{tails}_f(x_1)),$$
$$(\texttt{tails}_f(x_1), x_2, y_2, \texttt{tails}_f(x_1x_2)), \cdots$$
$$(\texttt{tails}_f(xx_n^{-1}), x_n, y_n, \texttt{tails}_f(x))$$

where $y_1 \cdots y_n = y$ (with each $y_i \in \Gamma^*$). By the definition of E of $\mathcal{T}'$, this corresponds to the sequence of edges,

$$(\lambda, x_1, y_1, \texttt{Suff}^{k-1}(x_1)),$$
$$(\texttt{Suff}^{k-1}(x_1), x_2, y_2, \texttt{Suff}^{k-1}(x_1x_2)), \cdots$$
$$(\texttt{Suff}^{k-1}(xx_n^{-1}), x_n, y_n, \texttt{Suff}^{k-1}(x)).$$

Also, by definition of σ, $\sigma(\texttt{Suff}^{k-1}(x)) = \sigma_{\mathcal{T}}(\texttt{tails}_f(x)) = z$. Thus $\mathcal{T}'(x) = yz$. A similar argument shows that for any x if $\mathcal{T}'(x) = yz$ (where $(\lambda, x, y, \texttt{Suff}^{k-1}(x)) \in \Pi_{\mathcal{T}'}$ and $\sigma(\texttt{Suff}^{k-1}(x)) = z$, then $f(x) = yz$.

($\Leftarrow$) Consider $\mathcal{T} = \{Q, \Sigma, \Gamma, q_0, E, \sigma\}$ for some function f and some $k \in \mathbb{N}$ such that $Q = \Sigma^{\leq k-1}$, $q_0 = \lambda$, and $(\forall q \in Q, \forall a \in \Sigma, \forall u \in \Gamma^*)[(q, a, u, q') \in E \Rightarrow q' = \texttt{Suff}^{k-1}(qa)]$. Now consider any $u_1, u_2 \in \Sigma^*$ such that $\texttt{Suff}^{k-1}(u_1) = \texttt{Suff}^{k-1}(u_2)$. By construction of $\mathcal{T}$, both u_1 and u_2 lead to the same state and therefore $\texttt{tails}_f(u_1) = \texttt{tails}_f(u_2)$. Thus f is k-ISL. □

Constructing $\mathcal{T}_{ISL}(f)$ for a phonological rule

Given a rule of the form in (1),

(1) $x_i \rightarrow y_i \ / \ U \_\_ V$

where $U \subseteq \{\#, \lambda\} \cdot \Sigma^*$, $V \subseteq \Sigma^* \cdot \{\#, \lambda\}$, and $(\mathrm{x}_i, \mathrm{y}_i) \in X \times Y$ for $X, Y \subseteq \Sigma^*$, provided U, X, and V are finite we can construct $\mathcal{T}_{ISL}(f)$ as follows. Let $UXV = \{uxv \mid$

$u \in U, x \in X, v \in V\}$ and let $k = |\texttt{max}(UXV)|$. In other words, the k-value of the function has to cover the longest possible uxv string.

Let H be the set of all possible strings formed by concatenating a string from X (a target) with a string from V (a right context). So $H = \{xv \mid x \in X \text{ and } v \in V\}$. These are the strings that the FST will 'hold' all prefixes of (if they are preceded by some $u \in U$) until it verifies that an entire xv has been read, at which point it will output the corresponding yv. The following auxiliary function π will be used to determine, for any given state q, the (possibly empty) string being held in q.

$$\pi(q) = \begin{cases} z & q = suz \; for \; s \in \Sigma^* \; and \; z \in \texttt{Pref}(h) \; for \; some \; h \in H \\ \lambda & otherwise \end{cases}$$

$\mathcal{T}_{ISL}(f) = \{Q, \Sigma, \Gamma, q_0, \delta, \sigma\}$, where,

- Σ is the input alphabet and Γ is the output alphabet, with $\Sigma \subseteq \Gamma$
- $Q = \Sigma^{\leq k-1}$
- $q_0 = \lambda$
- $\forall q \in Q$ and $a \in \Sigma$,
  1. If $q = suh_0...h_i$ for any $s \in \Sigma^*$, $u \in U$, and $h \in H$ and $a = h_{i+1}$, then $\delta(q, a) = (\lambda, \texttt{Suff}^{k-1}(qa))$.
  2. Else if $q = suh_0...h_i$ for any $s \in \Sigma^*$, $u \in U$, and $h \in H$ but $a \neq h_{i+1}$, let $o = \pi(q)a$ be the output, OR if $qa = ux_iv$ for some $u \in U, x_i \in X$, and $v \in V$, let $o = y_iv$ be the output. Now let $t = \pi(\texttt{Suff}^{k-1}(qa))$ be the string being held in the destination state. The portion (if any) of o that overlaps with t is $r = \texttt{max}(\texttt{Suff}(o) \cap \texttt{Pref}(t))$. Then $\delta(q, a) = (or^{-1}, \texttt{Suff}^{k-1}(qa))$.
  3. If neither of the above conditions holds, then $\delta(q, a) = (a, \texttt{Suff}^{k-1}(qa))$.
- $\forall q \in Q$,

$$\sigma(q) = \begin{cases} y_iv & if \; q\# = ux_iv \; for \; some \; u \in U, x_i \in X, v \in V \\ \pi(q) & otherwise \end{cases}$$

OSL Functions

Definition 11 (Output Strictly Local Functions)**.** *A function f is Output Strictly Local iff $\exists k \in \mathbb{N}$ such that for all $u_1, u_2 \in \Sigma^*$, it is the case that if $\texttt{Suff}^{k-1}(f(u_1)) = \texttt{Suff}^{k-1}(f(u_2))$ then $\texttt{tails}_f(u_1) = \texttt{tails}_f(u_2)$.*

Theorem 6. *A function f is Output Strictly Local iff $\exists k \in \mathbb{N}$ such that f can be described with a SFST $\mathcal{T}_{OSL}(f)$ for which*

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

Proof. ($\Rightarrow$) Consider any f such that there is a k such that for all $u_1, u_2 \in \Sigma^*$ it is the case that if $\texttt{Suff}^{k-1}(f(u_1)) = \texttt{Suff}^{k-1}(f(u_2))$ then $\texttt{tails}_f(u_1) = \texttt{tails}_f(u_2)$. We show that $\mathcal{T}_{OSL}(f)$ exists. Since $Q = \Gamma^{\leq k-1}$ is a finite set, the equivalence relation $\sim_f$ partitions Σ^* into at most $|Q|$ blocks. Hence f is subsequential and a canonical subsequential transducer $\mathcal{T}(f) = \{Q_\mathcal{T}, \Sigma, \Gamma, q_0, E_\mathcal{T}, \sigma_\mathcal{T}\}$ exists.

We construct $\mathcal{T}'$ recognizing f from $\mathcal{T}(f)$, again utilizing a mapping h from the states of $\mathcal{T}'$ onto $\mathcal{T}(f)$. h is defined recursively as follows:

1. $\lambda \mapsto_h \texttt{tails}_f(\lambda)$
2. $(\forall u \in \Sigma^*, a \in \Sigma), [\texttt{Suff}^{k-1}(f(ua)) \mapsto_h \texttt{tails}_f(ua)]$

Since $\mathcal{T}(f)$ exists, h can be determined by considering only $u \in \Sigma^{k-2}$. Now construct $\mathcal{T}' = \{Q, \Sigma, \Gamma, q_0, E, \sigma\}$ as follows.

- $Q = \Gamma^{\leq k-1}$ and $q_0 = \lambda$
- $\forall q \in Q, \sigma(q) = \sigma_\mathcal{T}(h(q))$
- $\forall q \in Q, \forall a \in \Sigma, \forall u \in \Gamma^*$, $[(q, a, u, \texttt{Suff}^{k-1}(qu)) \in E$ iff $(h(q), a, u, h(\texttt{Suff}^{k-1}(qu)))$ $\in E_\mathcal{T}]$

We show that $\mathcal{T}' = \mathcal{T}_{OSL}(f)$. Clearly, by its construction, the states and transitions meet the requirements of (1) and (2) of Theorem 6. As before, the mapping h ensures that $\mathcal{T}'$ computes the same function as $\mathcal{T}(f)$ because f is k-OSL: for all $x, x' \in \Sigma^*$ if $\texttt{Suff}^{k-1}(f(x)) = \texttt{Suff}^{k-1}(f(x'))$ then $\texttt{tails}_f(x) = \texttt{tails}_f(x')$.

($\Leftarrow$) Consider $\mathcal{T} = \{Q, \Sigma, \Gamma, q_0, E, \sigma\}$ for some function f and some $k \in \mathbb{N}$ such that $Q = \Gamma^{\leq k-1}$, $q_0 = \lambda$, and $(\forall q \in Q, \forall a \in \Sigma, \forall u \in \Gamma^*)[(q, a, u, q') \in E \Rightarrow q' = \texttt{Suff}^{k-1}(qu)]$. Now consider any $u_1, u_2 \in \Sigma^*$ such that $\texttt{Suff}^{k-1}(f(u_1)) = \texttt{Suff}^{k-1}(f(u_2))$. By construction of $\mathcal{T}$, both u_1 and u_2 lead to the same state and therefore $\texttt{tails}_f(u_1) = \texttt{tails}_f(u_2)$. Thus, f is k-OSL. □

Constructing $\mathcal{T}_{OSL}(f)$

$\mathcal{T}_{OSL}(f) = \{Q, \Sigma, \Gamma, q_0, \delta, \sigma\}$, where,

- Σ, Γ, q_0, and σ are defined as for $\mathcal{T}_{ISL}(f)$
- $Q = \Gamma^{\leq k-1}$
- Conditions 1 and 3 of δ are defined as for $\mathcal{T}_{ISL}(f)$. Condition 2 is defined as follows.
  - If $q = suh_0...h_i$ for any $s \in \Gamma^*$, $u \in U$, and $h \in H$, and $a \neq h_{i+1}$, let $o = \pi(q)a$ be the output, OR if $qa = ux_iv$ for some $u \in U$, $x_i \in X$, and $v \in V$, let $o = y_iv$ be the output. Now let $t = \pi(\texttt{Suff}^{k-1}(qo))$ be the string being held in the destination state. The overlap between o and t is $r = \texttt{max}(\texttt{Suff}(o) \cap \texttt{Pref}(t))$. Then $\delta(q, a) = (or^{-1}, \texttt{Suff}^{k-1}(qo))$.

Word-initial processes

If and only if $\exists u \in U$ such that $u = \#u'$ for some $u' \in \Sigma^*$, the above constructions should be modified as follows:

- $Q = \{\#\}\cdot\Sigma^{\leq k-2} \cup \Sigma^{k-1} \cup \{\#\}$ (for $\mathcal{T}_{ISL}(f)$)
- $Q = \{\#\}\cdot\Gamma^{\leq k-2} \cup \Gamma^{k-1} \cup \{\#\}$ (for $\mathcal{T}_{OSL}(f)$)
- $q_0 = \#$

Minimization

The above constructions for $\mathcal{T}_{ISL}(f)$ and $\mathcal{T}_{OSL}(f)$ do not produce minimal FSTs. The minimal version of both FSTs can be constructed by instead using the state set $Q = \bigcup_{w \in UXV}[\texttt{Pref}(w) - w]$. Since $k = |\texttt{max}(UXV)|$, this state set will include only the factors up to $k-1$ of all possible uxv strings. The FSTs can then be constructed

just as before, with one modification. For each transition in δ, if the destination state $\texttt{Suff}^{k-1}(qa)$ (for $\mathcal{T}_{ISL}(f)$) or $\texttt{Suff}^{k-1}(qo)$ (for $\mathcal{T}_{ISL}(f)$) is not in Q, then the destination state of that transition should be q_0.

To see an example of this minimized construction, recall the example rule in (2). The minimized $\mathcal{T}_{ISL}(f)$ and $\mathcal{T}_{OSL}(f)$ are shown in Figures A.1 and A.2, respectively.

(2) a → b / a __a

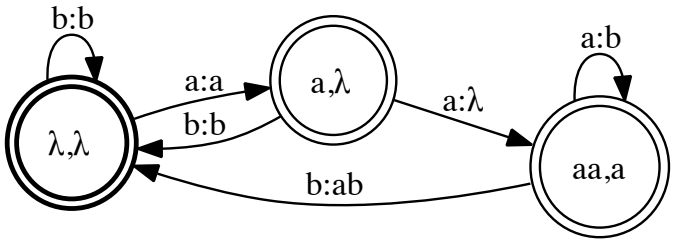


Figure A.1: Minimized $\mathcal{T}_{ISL}(f)$ for (2), $k = 3$, $\Sigma = \{\text{a, b}\}$

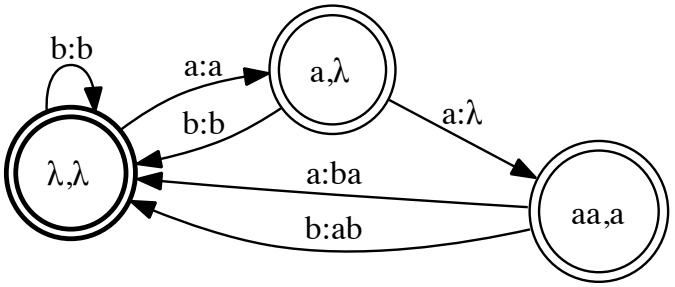


Figure A.2: Minimized $\mathcal{T}_{OSL}(f)$ for (2), $k = 3$, $\Sigma = \{\text{a, b}\}$

These minimized SL transducers are in fact the canonical subsequential transducers for f.

Properties of SL Functions

First, it can be shown that both the ISL and the OSL functions are proper subclasses of the subsequential functions. They are subclasses by definition, since every ISL or OSL function is a subsequential function. But there exist subsequential functions that are neither ISL nor OSL, as stated in the following theorem.

Theorem 7. *ISL and OSL functions are proper subclasses of subsequential functions.*

Proof. Consider the subsequential function f_1 represented by the SFST in Figure A.3. This function is neither ISL nor OSL for any $k \in \mathbb{N}$. Choose any such k. Consider two inputs bc^k and ac^k, which share a common suffix of length $k-1$. But $f_1(bc^kb) = bc^kb$ and $f_1(ac^kb) = ac^ka$, which means $\texttt{tails}_{f_1}(bc^k) \neq \texttt{tails}_{f_1}(ac^k)$. Likewise, the outputs bc^k and ac^k also share a common suffix of length $k-1$, but again the tails of their respective inputs are distinct. Hence by Definitions 10 and 11, there is no k for which f_1 is ISL or OSL. □

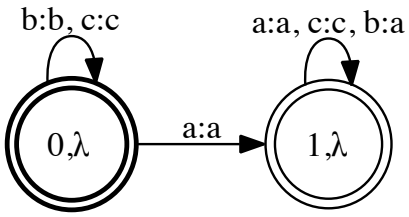


Figure A.3: A subsequential function f_1 which is neither ISL nor OSL, $\Sigma = \{\text{a, b, c}\}$

Next, it can be shown that the classes of ISL and OSL functions are not identical.

Theorem 8. *The class of ISL functions is incomparable to the class of OSL functions.*

Proof. Consider the function f_2 exemplified in Figure A.4. This function is ISL by Theorem 5. However, it is not OSL. Consider any $k \in \mathbb{N}$. Observe that $f_2(aa^k) = f_2(ab^k) = ab^k$, and so the outputs share the same $k-1$-length suffix. However, $\texttt{tails}_{f_2}(aa^k) \neq \texttt{tails}_{f_2}(ab^k)$, since $(a, b) \in \texttt{tails}_{f_2}(aa^k)$ but $(a, a) \in \texttt{tails}_{f_2}(ab^k)$.

Similarly, consider the function f_3 exemplified in Figure A.5. This function is OSL by Theorem 6. However, it is not ISL. Consider any $k \in \mathbb{N}$. The inputs cb^k and ab^k share the same $k-1$-length suffix. However, $\texttt{tails}_{f_3}(cb^k) \neq \texttt{tails}_{f_3}(ab^k)$, since $(b, b) \in \texttt{tails}_{f_3}(cb^k)$ but $(b, a) \in \texttt{tails}_{f_3}(ab^k)$.

Finally, the two classes are not disjoint: the identity function, for example, clearly belongs to both ISL and OSL. □

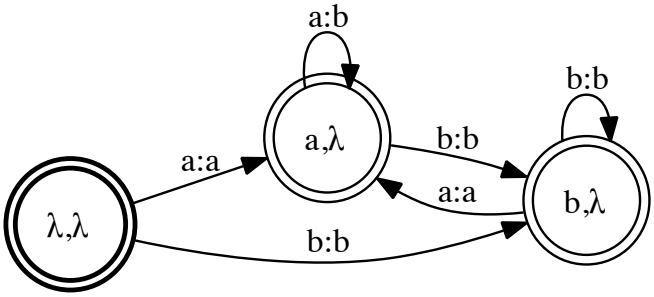


Figure A.4: A subsequential function f_2 which is ISL but not OSL, $k = 2$, $\Sigma = \{$a, b$\}$

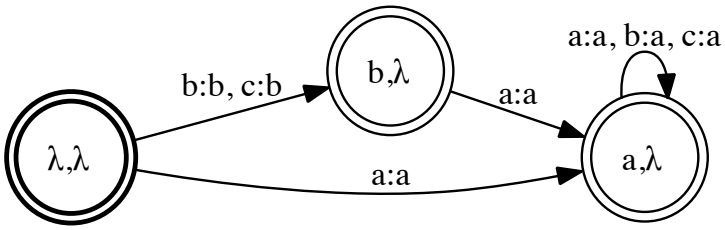


Figure A.5: A subsequential function f_3 which is OSL but not ISL, $k = 2$, $\Sigma = \{$a, b$\}$

The next theorem establishes that the OSL functions are not closed under composition.

Theorem 9. *The OSL functions are not closed under composition.*

Proof. It suffices to provide a counterexample of two functions f and g that are OSL but their composition $g \circ f$ is not. Let f be the 2-OSL function represented in Figure A.6. The input alphabet is $\Sigma = \{\text{V, d, n}\}$ and the output alphabet is $\Gamma = \{\text{V, d, n, } \tilde{\text{V}}\}$.

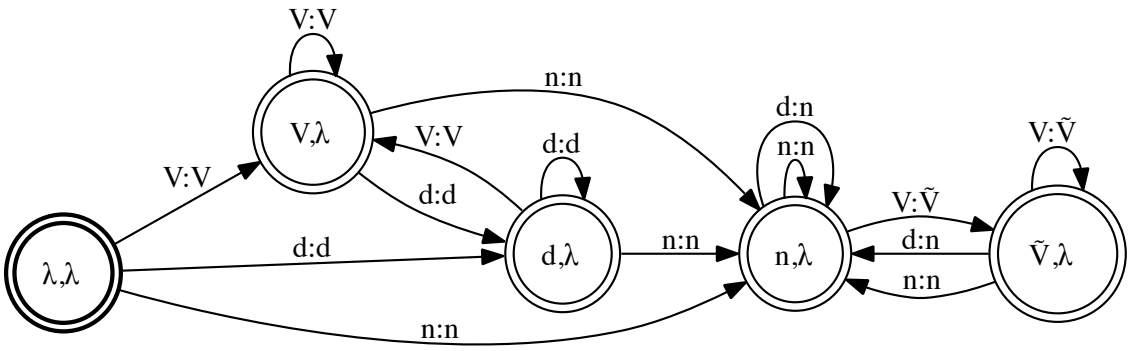


Figure A.6: OSL function f

And let g be the 1-OSL function represented in Figure A.7. The input alphabet is $\Sigma = \{\text{V, d, n, } \tilde{\text{V}}\}$ and the output alphabet is $\Gamma = \{\text{V, d, n}\}$.

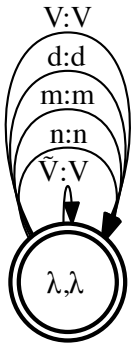


Figure A.7: OSL function g

The mapping represented by the composition $g \circ f$ is not OSL for any k. Consider the two inputs V^k and nV^k for some $k \in \mathbb{N}$. The outputs of $g \circ f$ for these inputs have the same suffix of length $k-1$: $g(f(V^k)) = V^k$ and $g(f(nV^k)) = nV^k$. But $\texttt{tails}_{g \circ f}(V^k) \neq$

$\texttt{tails}_{g\circ f}(nV^k)$, since $(d,d) \in \texttt{tails}_{g\circ f}(V^k)$ but $(d,n) \in \texttt{tails}_{g\circ f}(nV^k)$. Since k is arbitrary, it follows that $g \circ f$ is not OSL for any k. □

Lastly, it is shown that the right projection of SL functions is not guaranteed to be a SL language.

Theorem 10. *ψ_2 of ISL and OSL functions is not necessarily a Strictly Local language.*

Proof. Consider function f_4 depicted in Figure A.8, which, by Theorems 5 and 6, is both ISL and OSL. It can be shown that ψ_2 of f_4 is not SL-k for any k. Choose some $k \in \mathbb{N}$.

Case 1: k is even. Choose $w, w' \in \psi_2$ such that $w = a^k b$ and $w' = ba^k$. These strings must be in ψ_2 because for $n = k/2$, $f(a^n b) = a^k b$ and $f(ba^n) = ba^k$. If ψ_2 is SL-k, then by Theorem 1 (Chapter 2), since $w = a{\cdot}\, a^{k-1}{\cdot}\, b$ and $w' = b{\cdot}\, a^{k-1}{\cdot}\, a$ are in ψ_2, then $a{\cdot}\, a^{k-1}{\cdot}\, a = a^{k+1}$ must also be in ψ_2. But this is false, since there is no string $x \in \Sigma^*$ such that $f(x) = a^{k+1}$ (because $k+1$ is odd).

Case 2: k is odd. Choose $w, w' \in \psi_2$ such that $w = a^{k+1}b$ and $w' = ba^{k+1}$. These strings must be in ψ_2 because for $n = (k+1)/2$, $f(a^n b) = a^{k+1}b$ and $f(ba^n) = ba^{k+1}$. If ψ_2 is SL-k, then by Theorem 1, since $w = aa{\cdot}\, a^{k-1}{\cdot}\, b$ and $w' = ba{\cdot}\, a^{k-1}{\cdot}\, a$ are in ψ_2, then so must be $aa{\cdot}\, a^{k-1}{\cdot}\, a = a^{k+2}$. But this is false, since there is no string $x \in \Sigma^*$ such that $f(x) = a^{k+2}$ (because $k+2$ is odd).

Since k was arbitrary, it is shown that ψ_2 is not SL for any k.

□

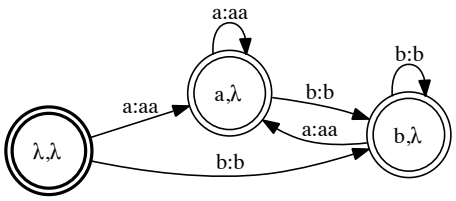


Figure A.8: A subsequential function f_4 which is both ISL and OSL but for which ψ_2 is not SL, $k = 2$, $\Sigma = \{\text{a, b}\}$