Non-local phonological processes as Multi-Tiered Strictly Local maps

Phillip Burness

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy in Linguistics

Department of Linguistics
Faculty of Arts
University of Ottawa

Dedication

In loving memory of Austin Samuel Lapointe (December 19, 1995 – April 5, 2018)
And I have not forgot, but a silence crept over me – Joanna Newsom

Abstract

Phonological processes can be characterized as functions from input strings to output strings, and treating them as mathematical objects like this reveals properties that hold regardless of how we implement them (i.e., with rules, constraints, or other tools). For example, Chandlee (2014) found that a vast majority of phonological processes can be modelled as Strictly Local (SL) functions, which are sensitive to a window of finite size. Long-distance processes like vowel and consonant harmony are exceptions to this generalization, although a key observation is that they look local once irrelevant information is ignored. This thesis shows how such selective attention can be modelled by augmenting SL functions with autosegmental tiers (e.g., Goldsmith, 1976). A single tier is sufficient to capture individual long-distance processes, and having multiple tiers available allows us to model multiple long-distance processes simultaneously as well as interactions between local and non-local patterns. Furthermore, probabilistic variants of these tier-based functions allow for a cognitively plausible model of what Zymet (2015) calls distance-based decay. Unrestricted use of multiple tiers is, however, quite powerful and so I additionally argue that tiersets should be defined from the perspective of individual input elements (i.e., potential process targets). Each input element designates a superset-subset hierarchy of tiers and pays attention to them alone; the tiers specified by another input element are either redundant or irrelevant. Restricting tiersets in this manner has beneficial consequences for learnability as it imparts a structure onto the learner's hypothesis space that can be exploited to great effect. Furthermore, tier-based functions meeting this restriction fail to generate a number of pathological behaviours that can be characterized as subsequential functions, a type of function that has previously been offered as a model of non-local phonological processes (Heinz & Lai, 2013; Luo, 2017; Payne, 2017). In light of their empirical coverage, their comparative lack of pathological predictions, and their efficient learnability, I conclude that tier-based functions act as a more accurate characterization of long-distance phonology.

Acknowledgements

First and foremost, this thesis would never have happened if my supervisor, Kevin McMullin, hadn't convinced me to come back to school for a PhD over a round of drinks. His enthusiasm for my endless questions and weird ideas kept me motivated through a global pandemic, which is no small feat. I am also indebted to Tania Zamuner, who supervised my first comprehensive project. I had never done any experimental work before, and my perhaps overly ambitious combination of eye-tracking and artificial grammar learning methodologies would have never worked without her careful guidance. Thanks is also due to all the professors whose courses I took during my program: Marc Brunelle, Laura Sabourin, Suzy Ahn, Dennis Ott, Andres Salanova, and Eric Matthieu. I never had a class I didn't enjoy, which really speaks to the quality of their teaching. Additionally, thank you to the University of Ottawa and the Social Sciences and Humanities Research Council of Canada for the generous funding, without which I could have never completed my doctoral studies.

Of course, academic research doesn't (or at least shouldn't) take place within the confines of a single institution, and I have many fellow linguists from other universities to thank. In particular, it is no exaggeration to say that Jane Chandlee's work on Strictly Local functions is the foundation for my entire thesis, and I was very fortunate to meet her and speak with her in my first year. Furthermore, considering the math-heavy content of this thesis, it might come to some as a shock that I did zero computational / mathematical work prior to taking on this PhD, and I could not have made the transition were it not for the encouragement of others in the computational / mathematical linguistics community. I especially owe thanks to Jeff Heinz, Adam Jardine, Jim Rogers, Thomas Graf, Aniello De Santo, Dakotah Labert, Alëna Aksënova, Jon Rawski, and Hossep Dolatian, who all provided precious feedback on my work at both official and unofficial conferences.

Moving on to the more personal, my boyfriend Jamie deserves plenty of credit for constantly reminding me (with maybe a little too much zeal) that I should be writing my thesis. He will be disappointed to find that I did not follow his suggestion to include an appendix of cute dog pictures, and for that I formally apologize. Also on the personal side, I would be remiss if I didn't acknowledge the plenteous support from my family, so thank you to my mom Louise, my dad David, my brother Wesley, my sister Andrea, my uncle John, and my aunt Linda. No less important are the friends that each in their own way alleviated the stress of grad school: thank you Carol for

the violin/piano duets; thank you Maddie for teaching me to knit; thank you Matt for sending me awful Tumblr memes; thank you Patrick, Dan, Alex, Sean, and Vida for being great roommates; thank you Drew and Tony for the evenings of Chinese take-out and Fire Emblem; thank you Darren for the nights of wine and 90s music; thank you Jenny for the long Skype conversations about life and RuPaul's Drag Race; thank you Rachael for the gossip on our walks along the canal; thank you Scotty and Brónagh for nerding out about Harry Potter with me; thank you Jordan and Christian for the anime convention weekends; and thank you OriginalRaisins (aka John) and LordPangTong (aka Paul) for your great Twitch channels and Discord communities.

Finally, I would like to express immense gratitude to Catherine, Paul, and Zachary Lapointe. Despite the much-too-brief time that I shared with their son / brother Austin, he made an incredibly positive impact on my life, and I am forever grateful for being treated like a family member during the many painful and uncertain days we all spent together at the hospital. I could not have worked past my grief and could not have carried on with my doctoral studies were it not for the memory of how proud Austin was of my initial efforts. This thesis is for him.

Table of Contents

List of Figures

List of Tables

List of Definitions

Notation glossary

$X \subseteq Y$	X is a subset of Y
$X \subset Y$	X is a proper subset of Y
$X \times Y$	the cross-product of sets X and Y
$X \cup Y$	the union of sets X and Y
$X \cap Y$	the intersection of sets X and Y
$\forall x$	for all x
$\exists x$	there exists x
$x \in Y$	x is a member of Y
$x \notin Y$	x is not a member of Y
$\{x \mid \phi\}$	the set of all x that satisfy ϕ
$\phi \wedge \psi$	ϕ and ψ
$\phi \vee \psi$	ϕ or ψ
$\phi \Rightarrow \psi$	ϕ implies ψ
$\phi \Leftrightarrow \psi$	ϕ if and only if ψ
Σ^*	all strings of any length that can be made using elements from Σ
$\Sigma^{\leq n}$	all strings up to length n that can be made using elements from Σ
$\lvert w \rvert$	the length of string w
$\texttt{fac}_k(w)$	the k-factors of string w
$\texttt{erase}_T(w)$	the string w with all non-members of T removed
$x \cdot y$	the concatenation of string x and string y
$x^{-1} \cdot y$	the removal of string x from the beginning of string y
$\texttt{suff}^n(w)$	the n-long suffix of string w
$\texttt{suff}^n_T(w)$	$\texttt{suff}^n(\texttt{erase}_T(w))$
$\texttt{pref}^n(w)$	the n-long prefix of string w
$\texttt{lcp}(W)$	the longest common prefix of stringset W
$f(w)$	the function f applied to input w
$\texttt{tails}_f(w)$	the tails of input w with respect to function f
$\texttt{cont}_f(\sigma, w)$	the contribution of σ given w with respect to function f

Chapter 1

Introduction

1.1 Overview

Sound patterns in human language are overwhelmingly *local* in that speech sounds tend to interact with other speech sounds in their direct vicinity. Take for instance the plural suffix in English. This suffix is normally pronounced as the voiced alveolar fricative [z] (like in 'dogs', 'cows', and 'bins'), but when it attaches to a noun that ends in a voiceless stop consonant such as [p], [t], or [k] it is pronounced as the voiceless alveolar fricative [s] (like in 'mops', 'cats', and 'tricks'). Importantly, while the pronunciation of the English plural suffix is variable, that variation is predictable and depends only upon the identity of the noun-final segment. A typical phonological analysis of these facts would postulate that the English suffix is mentally stored as voiced /z/, but that the phonological system will sometimes convert this to voiceless [s] in order to avoid a consonant cluster with mixed voicing.[1] The actual conversion can be accomplished in a number of ways, but no matter how we decide to implement it, the fact remains that the underlying /z/ is transformed to [s] in response to the immediately preceding segment, information that is decidedly local.

Some sound patterns, however, go against the grain and are sensitive to information that can be arbitrarily far away. Consider the perfective suffix in Aari, an Omotic language of Ethiopa. It is normally pronounced as the alveolar sibilant [s] as shown in (1a), but surfaces instead as the lamino-postalveolar sibilant [ʃ] when preceded (at any distance) by a lamino-postalveolar sibilant, as shown in (1b-f). Sibilant consonants are displayed in bold for clarity, and the distance between the alternating suffix and the alternation's trigger ranges from 0 to 4 segments, suggesting that this distance does not matter. *Long-distance* or *non-local* processes like Aari sibilant harmony are the focus of this thesis.

[1] I follow convention by writing underlying forms between slashes and surface forms between square brackets.

(1) Progressive unbounded sibilant harmony in Aari (Hayward, 1990)

a.	/baʔ-s-e/	[baʔ**s**e]	‘bring-PERF-3SG’
b.	/ʔuʃ-s-it/	[ʔu**ʃʃ**it]	‘cook-PERF-1SG’
c.	/ʧ’a̤ːq-s-it/	[**ʧ’**a̤ːq**ʃ**it]	‘swear-PERF-1SG’
d.	/ʒaʔ-s-it/	[**ʒ**aʔ**ʃ**it]	‘arrive-PERF-1SG’
e.	/ʃed-er-s-it/	[**ʃ**eder**ʃ**it]	‘see-PASS-PERF-1SG’
f.	/ʒa̤ːg-er-s-e/	[**ʒ**a̤ːger**ʃ**e]	‘sew-PASS-PERF-3SG’

A deceptively simple question that I aim to answer in this thesis is: how do local processes differ from non-local processes? Giving a satisfactory answer to this question first requires an exact notion of what it means to be local. To help pin down such a definition, compare the process of voicing assimilation affecting the English plural to the process of local palatalization affecting Japanese /s/. The sequence [si] is not allowed in Japanese, and so it is replaced with [ɕi] wherever it would occur, like when the verb root /hos-/ ‘dry’ is followed by the past tense suffix /-ita/, giving us [hoɕita] ‘dried’ rather than *[hosita]. At a glance, voicing assimilation and palatalization look completely unrelated, and indeed, most linguists would agree that they have different articulatory/perceptual underpinnings. At a higher level, however, they share a very important property in common. Namely, both processes can be expressed as one sound (the process target) changing in response to leftward or rightward material (the process trigger) *up to a fixed maximum distance away*[2].

Let us therefore informally define a local process as one where a target is always within a fixed maximum distance from the trigger it is reacting to. Aari sibilant harmony does not fulfill this requirement since it (in principle) applies no matter how many other sounds intervene between the target and trigger. It is thus tempting to say that Aari sibilant harmony has nothing in common with English voicing assimilation or Japanese palatalization. This would, however, be ignoring the important observation that Aari sibilant harmony is only non-local when we are looking at the raw string of all sounds. If we erase everything but the sibilants (i.e., get rid of all the “irrelevant” intervening material), the trigger and target become adjacent. Though the process of sibilant harmony is not local in a literal sense, it is nonetheless local if we relativize our attention to just the sibilants. Aari sibilant harmony is not unique in this way; most non-local phonological processes can be reinterpreted as local when only a subset of the sound inventory is considered.

Adopting a computational perspective, this thesis explores the above idea that local and long-distance phonological processes differ only in how they assess locality. Where local processes consider the raw string of all elements, long-distance processes have a more selective attention and consider only elements of a particular type. Letting a pattern’s *tier* be its inventory of relevant

[2]At least when we are working with strings (as I do for most of this thesis) and not a more complex data structure like a graph.

speech sounds, local patterns are then those for which the tier is equal to the language's entire sound inventory, and long-distance behaviour emerges when the tier is a proper subset of the inventory. While ideas along these lines have existed in phonological theory since at least the advent of *autosegmental phonology* (Goldsmith, 1976), it is only recently that they have been approached from the computer science point of view. Computational work making use of tiers has mainly been concerned with *phonotactic* patterns, static co-occurence restrictions that divide well-formed pronunciations from ill-formed pronunciations (Heinz et al., 2011; Jardine & Heinz, 2016; McMullin, 2016; McMullin & Hansson, 2016; Jardine & McMullin, 2017; Aksënova & Deskmukh, 2018; McMullin et al., 2019; Lambert & Rogers, 2020). The work collected in this thesis represents the first in-depth computational exploration of the utility of tiers for modelling morpho-phonological transformations. The main innovation I introduce concerns the reconciliation of simultaneous and conflicting influences from different tiers, a question which previously had been considered only in the context of phonotactics (Aksënova & Deskmukh, 2018; McMullin et al., 2019). I show that intuitive analyses of such challenging phenomena are possible when we frame the formal definitions specifically from the target's perspective.

What do we gain from pursuing such a computational line of inquiry? Theoretical analyses of phonological processes employ a wide variety of mechanisms, the most common of which are optimization over sets of candidates or sequential application of rewrite rules. That being said, whether we adopt a constraint-based approach or a rule-based approach (or another approach entirely), the mappings from underlying representations to surface pronunciations that we intend to model can be characterized as string-to-string functions, mathematical objects that pair each input string with a corresponding output string. An advantage of such mathematical abstraction is that we can discover and describe properties of phonological processes that hold regardless of how we go on to implement those processes. On this front, my thesis provides concrete but maximally theory-neutral formalizations of (i) the distinction between a local and long-distance process and (ii) the resolution of conflicting influences from distinct sources. Another related reason to investigate phonology through the lens of mathematics/computation is that we can make strong hypotheses about the division between logically possible and actually possible patterns, usually via arguments of (non-)learnability. A secondary contribution of this thesis is to show that properties of tier-based processes make them efficiently learnable from positive data, letting us use the set of tier-based processes as a candidate for the set of possible long-distance processes.

One can rightfully point out here that a purely computational approach neglects the substantive aspects of long-distance phonological patterns, which have long been a focus of the literature, especially within constraint-based frameworks. Many vowel harmonies are hypothesized to evolve through the exaggeration of co-articulation (Boyce, 1988; Ohala, 1994; Manuel, 1999; Beddor et al., 2001, 2002; Kaun, 2004; Linebaugh, 2007) and some harmonies are hypothesized to serve

as reinforcement for perceptually weak contrasts (Suomi, 1983; Kaun, 1995, 2004; Walker, 2005; Kimper, 2017). For their part, consonant harmonies are hypothesized to derive from speech errors involving similar sounds (Hansson, 2010). Such considerations of substance are important, of course, and many phonological analyses incorporate them directly; for example, a constraint-based analysis of rounding harmony might include a constraint that forces an instance of rounding in a perceptually weak environment to extend its temporal domain and therefore achieve a stronger likelihood of being perceived correctly (see for example Kaun, 1995). Though my thesis largely ignores substantive aspects of patterns, it does not argue against substantive approaches, rather it argues that long-distance phonological processes share a core set of computational characteristics despite their varied diachronic causes and despite the diverse articulatory and perceptual factors that shape them synchronically.

1.2 Structure of the thesis

I begin in Chapter 2 by laying the conceptual groundwork for the chapters that follow. There I discuss how local phonotactics have been described using the class of Strictly Local (SL) formal languages (McNaughton & Papert, 1971; Rogers & Pullum, 2011; Rogers et al., 2013) and how certain properties of these SL languages were extended to processes by Chandlee (2014) and Chandlee et al. (2014, 2015), yielding the SL *functions*. Then, after illustrating how the SL functions can be represented using finite-state transducers (FSTs), I demonstrate their broad empirical coverage with real examples while also highlighting how they fail to describe non-local processes like Aari sibilant harmony from above. I close the chapter by discussing a more powerful class of functions known as the subsequential functions, which are capable of modelling non-local processes like vowel harmony (Heinz & Lai, 2013), consonant harmony (Luo, 2017), and dissimilation (Payne, 2017). The various classes of tier-based functions that I define in subsequent chapters are intermediate in power between the SL and subsequential functions.

Chapter 3 starts with a return to phonotactics and formal languages. The key intuition I expand upon there is that non-local patterns appear local once irrelevant material is ignored or once relevant material is projected onto a separate level of representation. Such suppression/projection has traditionally been accomplished using autosegmental *tiers*. After providing an overview of how tiers have been used in traditional generative phonology, I recapitulate work showing that we can augment an SL language with a tier to yield a Tier-based Strictly Local (TSL) language. These TSL languages readily model non-local phonotactics while maintaining many desirable properties of SL functions (Heinz et al., 2011; McMullin, 2016; McMullin & Hansson, 2016; Jardine & Heinz, 2016; Jardine & McMullin, 2017; Lambert & Rogers, 2020). By augmenting SL functions in a parallel manner, we can equally describe long-distance processes while maintaining desir-

able properties of SL functions. I go on to show that such TSL functions (Burness & McMullin, 2019; Hao & Andersson, 2019; Hao & Bowers, 2019) can capture common behaviours exhibited by long-distance processes such as segmental transparency, segmental blocking, parasitism, target iciness, and iterativity. The TSL functions are not all powerful however, and I close the chapter by discussing some shortcomings that are addressed in Chapters 4 and 5.

The main shortcoming of TSL functions addressed by Chapter 4 is that they are mostly limited to describing one process at a time, whereas real language patterns often involve multiple processes applying in tandem. Chandlee et al. (2018) demonstrated that many interactions between local processes can be modelled using a single SL function, circumventing the need to model the interaction as a sequential application of multiple different SL functions. Multiple tiers are necessary if we are to accomplish a similar melding of non-local processes, since information that is irrelevant to one process can be crucial for another and vice versa. I accordingly define the class of Multi-Tiered Strictly Local (MTSL) functions, which are exactly like the TSL functions except that they can track more than one tier at a time. We run a very real risk of overgenerating if we do not place any limits on the number of available tiers and the relationships between those tiers, so I also define a restriction on MTSL functions which I call *target specification*. I go on to show that target-specified MTSL functions can model many complex interactions such as the overriding of a local process by a non-local one (or vice versa), split harmony, and trigger preference. I additionally show that the target-specified MTSL functions can in some instances overcome another limitation of TSL functions, namely the inability to model non-local processes with two-sided contexts. The chapter closes with a comparison between the subsequential and MTSL functions, showing that the former class predicts pathological behaviours that the latter class cannot model, suggesting that tier-based functions are a more accurate characterization of non-local processes.

All of the patterns analyzed prior to Chapter 5 are unidirectional, with any non-local triggers either always appearing to the left or always appearing to the right of their targets. It is not uncommon, however, for a process to apply bidirectionally, usually starting from a root and affecting both prefixes and suffixes. Bidirectional application is, however, beyond the capabilities of an individual TSL or MTSL function. To accommodate such processes, I take inspiration from Heinz & Lai (2013) who showed that bidirectional vowel harmony can be modelled as a progressive subsequential function applying to the output of a regressive subsequential function or vice versa (i.e., by composing two subsequential functions with opposite directionality). I extend their result by showing that a composition of two tier-based functions is sufficient. I also discuss a phenomenon known as *unbounded circumambience* (Jardine, 2016) where the ultimate fate of an input segment depends on unbounded information in both directions simultaneously. To varying degrees, these patterns can be intuitively modelled as the composition of two opposite-direction tier-based functions without the need for problematic strategies such as abstract intermediate markup.

Chapters 2 through 5 tacitly assume that processes apply deterministically, which is a convenient abstraction but not one that can be globally maintained. Many processes apply only *optionally*, and Chapter 6 explores how the structure of TSL and MTSL functions can be modified to account for this. Of particular importance to this chapter is the phenomenon of *distance-based decay* (Zymet, 2015), whereby a process applies with exponentially decreasing probability as the trigger and target become more distant. I show how weighted finite-state transducers built according to a TSL or MTSL template model this decay in a cognitively plausible manner, and that the approach can accurately reproduce the rates of application in two real-language data sets exhibiting distance-based decay. The first of these case studies is of Malagasy rounding dissimilation, and the second is of Hungarian backness harmony.

Chapter 7 turns to considerations of learnability. Here I start with a summary of relevant prior results, the most notable being that SL functions can be efficiently learned from positive evidence (Chandlee et al., 2014, 2015) and that TSL languages can also be learned from positive evidence, even without prior knowledge of the tier (Jardine & Heinz, 2016; Jardine & McMullin, 2017). Following that, I improve upon the results in Burness & McMullin (2019). They showed that any TSL function can be learned from positive evidence when the tier is known in advance, and that the tier itself can be learned provided that the function only ever cares about the most recent single tier element. I present an alternative implementation of their tier learning method that is much more efficient; the original version was slower than an algorithm for learning the superclass of subsequential functions, but the present version is faster, running at speeds comparable to those established for SL function learning by Chandlee et al. (2014, 2015). I then extend these results, showing that target specification (defined in Chapter 4) imparts a structure onto the space of possible hypotheses which a learner can exploit to great effect. Tier learning is guaranteed when each input element only tracks a single tier, and several test cases suggest that efficient tier learning is also possible when an element tracks more than one tier, so long as the tiers are in a superset-subset relationship.

Finally, Chapter 8 concludes and is followed by two appendices. Appendix A contains analyses of additional unbounded circumambient patterns that were excluded from Chapter 5 for reasons of space only. Appendix B contains formal statements and proofs of the learning results that were explained more informally in Chapter 7.

Chapter 2

Background

The overarching goal of this dissertation is to argue that long-distance phonological processes operate according to a relativized notion of locality. Specifically, they can be modeled using function classes that require the output fate of an input segment to depend solely on 'local' information, where locality is assessed with reference to one or more subsets of the available alphabets known as *tiers*. These tier-based functions are direct extensions of the Strictly Local (SL) functions initially developed by Chandlee (2014) and Chandlee et al. (2014, 2015). Before I can talk effectively about tier-based functions, then, it is necessary to illustrate the SL functions and how they work. I start by building up to a formal definition and characterization of the SL functions, introducing along the way notational conventions that I will use throughout the remaining chapters. Following that, I demonstrate the capabilities of the SL functions with real language examples as well as demonstrate how they are incapable of modelling long-distance processes. Finally, I introduce the subsequential functions which are a model of long-distance processes to which I will compare my own proposals.

2.1 Computational preliminaries

2.1.1 Formal languages and strictly locality

To better understand the limits of possible language patterns, early work in computational linguistics classified patterns according to the machinery necessary to produce them, giving rise to what is known as the Chomsky hierarchy of formal languages (Chomsky, 1956). This language-theoretic perspective abstracts away from linguistic meaning, defining a language as a collection of word forms made from a specified alphabet. By convention, the alphabet in question is denoted by capital sigma (Σ), and in the context of phonology represents an inventory of phones. A word or *string* is some finite sequence of symbols from Σ with a strict order. Arbitrary strings are often referred

to using lower case letters from the end of the English alphabet (like w), although λ is used for the unique *empty string*. Given the alphabet Σ, the set denoted by Σ^* is the set containing λ and all strings of any length that can be made from elements in Σ. A formal language L is then either equal to Σ^* or is some subset of Σ^*, written as $L \subseteq \Sigma^*$. The Chomsky hierarchy then classifies these formal languages according to the criteria by which they include and exclude strings.

At the very bottom of the Chomsky hierarchy are the *finite languages*, which can be modelled as a finite list of words. Anything on the list is considered a good string and included in the language, whereas anything not on the list is considered a bad string and excluded from the language. The original Chomsky hierarchy distinguishes four classes of languages above the finite languages, ordered according to the minimum computational power necessary to decide whether any arbitrary string belongs to the language. From least to most powerful, these are the regular languages, the context-free languages, the context-sensitive languages, and the recursively enumerable languages.[1] A visual representation of the hierarchy is shown in Figure 2.1.1. Of these classes, the regular languages are the most relevant to this dissertation, since phonology is argued to be at most regular in terms of complexity (Johnson, 1972; Kaplan & Kay, 1994). In most cases, though, phonology does not seem to require the full capabilities of the regular languages. Accordingly, much work has divided the regular region into a hierarchy of *subregular* formal language classes (McNaughton & Papert, 1971; Rogers & Pullum, 2011; Rogers et al., 2013), including the Strictly Local languages.

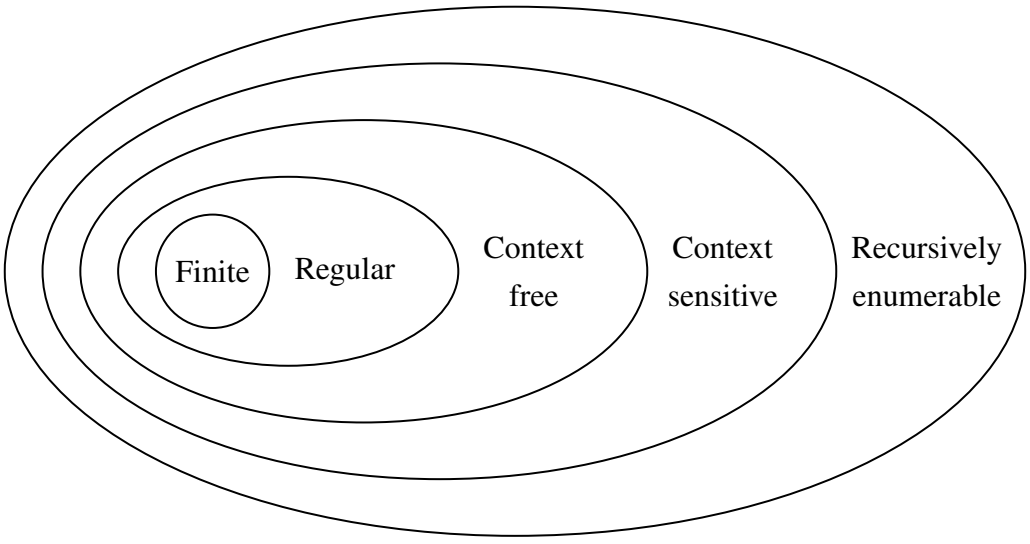


Figure 2.1: The Chomsky hierarchy of formal languages

[1]Such classes are often defined in terms of the machines that can model them: finite-state automata model regular languages, pushdown automata model context-free languages, linear bounded automata model context-sensitive languages, and turing machines model recursively enumerable languages.

The Strictly Local languages are those that ban particular contiguous sequences of up to a fixed length, excluding any string that contains one or more illegal contiguous sequences.[2] In order to formalize the functioning of a Strictly Local language, the following notation will be necessary. First, the length of a string w is written as $|w|$, for example $|abcd| = 4$. Note that in the special case of the empty string λ, $|\lambda| = 0$. Next, a k-factor of a string w is any contiguous substring of w such that $|w| = k$, though in the special case that $|w| \leq k$, the one and only k-factor of w is w itself. In what follows, $\texttt{fac}_k(w)$ denotes all the k-factors contained in a string w. For example, $\texttt{fac}_3(aababb) = \{aab, aba, bab, abb\}$ and $\texttt{fac}_{47}(aababb) = \{aababb\}$. Finally, $\Sigma^{\leq n}$ is the set of all possible strings with length n or shorter made from symbols in the alphabet Σ.

With this notation in mind, we can create a Strictly k-Local (SL_k) language by specifying a grammar G that is a subset of $\Sigma^{\leq k}$. The grammar can act as a list of legal k-factors, in which case we get a positively defined language that includes all and only those strings containing just legal k-factors (i.e., those that do not contain a k-factor not in G). The grammar can also act as a list of illegal k-factors, in which case we get a negatively defined language that excludes all and only those strings that contain an illegal k-factor (i.e., includes all and only those strings that do not contain an illegal k-factor). Any SL_k language can be defined positively or negatively, although negative definitions may be more intuitive to phonologists working in constraint-based frameworks like Optimality Theory (OT: Prince & Smolensky, 2004) since they function like inviolable constraints.

Definition 2.1 Strictly k-Local language (positive)**.**
A language L is Strictly k-Local if and only if there is a grammar $G \subseteq \Sigma^{\leq k}$ such that:

$$L = \{w \in \Sigma^* \mid \texttt{fac}_k(w) \subseteq G\}$$

Definition 2.2 Strictly k-Local language (negative)**.**
A language L is Strictly k-Local if and only if there is a grammar $G \subseteq \Sigma^{\leq k}$ such that:

$$L = \{w \in \Sigma^* \mid \texttt{fac}_k(w) \cap G = \emptyset\}$$

A wide variety of attested phonotactic patterns can be described in this manner. For example, in Japanese the alveolar fricative [s] cannot occur immediately before the high front vowel [i]. This fact is witnessed by the complete absence of the sequence *[si] in the language's lexicon but is also observed by way of alternations that occur to avoid said sequence. One such alternation can be seen when a verb root ending in /s/ is put into its polite form. Compare the mappings /hos-u/ → [hosu] 'dry-NPST' and /hos-imas-u/ → [hoɕimasu] 'dry-POL-NPST'. Assuming that Σ is the inventory of all surface phones in the Japanese language, the palatalization of [s] before [i] results in an SL_2 language. The phonotactic grammar G of Japanese can be said to penalize the 2-factor *[si], which will prevent /hos-imas-u/ 'dry-POL-NPST' from surfacing as *[hosimasu]. The

[2] An *n-gram model*, which may be more familiar to some readers, is effectively a probabilistic SL language.

palatalized form [hoɕimasu] is chosen instead since it does not include any illegal 2-factors and so is an acceptable output form. Note that such a language model does not actually describe the apparent *process* of palatalization, just its *result*.

An important property of the SL_k languages that directly follows from their abstract characterization is presented in Theorem 2.1 and is known as Suffix Substitution Closure (SSC; Rogers & Pullum, 2011; Rogers et al., 2013). A precise definition of SSC requires a means of denoting string concatenation. Following convention, I write the concatenation of two strings u and v as $u \cdot v$. For $u = aba$ and $v = bba$, then, $u \cdot v = ababba$. Informally, SSC states that when two grammatical strings have a matching middle section with a length of at least $k-1$, we can *substitute the suffixes* that come after the shared sequence without causing ungrammaticality.

Theorem 2.1 Suffix Substitution Closure (adapted from Rogers & Pullum 2011)**.**
A stringset L is Strictly k-Local if and only if for any two strings $u_1 \cdot x \cdot v_1$ and $u_2 \cdot x \cdot v_2$ such that $|x| \geq k-1$, it is the case that:

$$[u_1 \cdot x \cdot v_1 \in L \ \wedge \ u_2 \cdot x \cdot v_2 \in L] \implies u_1 \cdot x \cdot v_2 \in L$$

Suffix Substitution Closure is very useful for demonstrating that a language is not SL_k for a specific value of k. Take for instance the pattern of liquid harmony in Bukusu, where underlying /l/ becomes output [r] if the nearest leftward surface liquid is [r] (Odden, 1994; Hansson, 2010). The pattern of liquid harmony affects the applicative suffix /-ila/, exemplified by the data in (2). The suffix's underlying /l/ surfaces faithfully when the root contains no liquids as in (2a) or when the only liquids in the root are all instances of /l/ as in (2b). When the base contains an /r/, though, the liquid in the applicative suffix alternates to obey harmony. This happens across a single vowel as in (2c) and at further distances as in (2d).

(2) Bukusu liquid harmony (Odden, 1994)
   a. xam-il-a ‘milk-APPL’
   b. lim-il-a ‘cultivate-APPL’
   c. kar-ir-a ‘twist-APPL’
   d. rum-ir-a ‘send-APPL’

The pair of forms [xamila] ‘milk-APPL’ and [rumira] ‘send-APPL’ show us that the result of liquid harmony is not SL_3. We can divide [rumira] into [ru·mi·ra] and divide [xamila] into [xa·mi·la]. The shared internal sequence [mi] is of length 2, so if the result of liquid harmony were SL_3 we could legally swap the suffix [ra] of [rumira] with the suffix [la] of [xamila]. However, swapping the suffixes gives us *[rumila] which violates liquid harmony.[3] We can furthermore show that, in

[3]Although it should be said that liquid harmony in Bukusu becomes optional at distances greater than one intervening segment (Hansson, 2010, p. 96). Optionality will be discussed in Chapter 6.

theory, liquid harmony is not SL_k for any value of k. Pick an arbitrary value for k and suppose that we have two words [rumi·x·ra] and [xami·x·la], where x is made entirely of non-liquid segments and is of length $k-1$. Swapping the suffixes [ra] and [la] gives us *[rumi·x·la] which violates liquid harmony.

Suffix Substitution Closure is also useful for one of its corollaries. As expressed in Corollary 2.1, any two grammatical strings from an SL_k language that end in the same $k-1$ (or more) symbols can be legally continued by the exact same set of strings. The statement of the corollary requires a way to designate string suffixes. A suffix of some string w is any string s such that $w = x \cdot s$ and $x, s \in \Sigma^*$. Note that any string is a suffix of itself, and that the empty string λ is a suffix of every string. When $|w| \geq n$, $\texttt{suff}^n(w)$ denotes the unique suffix of w with a length of n; when $|w| < n$, $\texttt{suff}^n(w)$ simply denotes w itself.

Corollary 2.1 Suffix-defined residuals (Chandlee et al., 2015)**.**
A stringset L is Strictly k-Local if and only if for all pairs $w_1, w_2 \in \Sigma^*$:

$$\texttt{suff}^{k-1}(w_1) = \texttt{suff}^{k-1}(w_2) \;\Rightarrow\; \{v \mid w_1 \cdot v \in L\} = \{v \mid w_2 \cdot v \in L\}$$

Consider again the Japanese requirement of palatalization before [i]. We've determined the result of this rule to be SL_2, which means that any two strings that end in the same one symbol (or more) will have the same legal continuations. Take the two verb stems /hanas-/ 'talk' and /tomos-/ 'light (a candle)' which both end in /s/. The strings [hanas] and [tomos] both can be continued by a string starting with [a] as in [hanas-anai] 'talk-NEG' and [tomos-anai] 'light-NEG', by a string starting with [e] as in [hanas-e-ru] 'talk-POT-NPST'[4] and [tomos-e-ru] 'light-POT-NPST', etc. Neither string, however, can be continued by a string starting with [i] due to the requirement of palatalization. Indeed, any string that can be added to the end of the string [hanas] can be added to the end of the string [tomos] and vice versa precisely because they share a suffix with length $k - 1 = 1$. It is this notion of Suffix-defined Residuals that Chandlee (2014) and Chandlee et al. (2014, 2015) use to define and characterize the SL functions. How they do so is described in the next section.

2.1.2 String-to-string mappings and strict locality

Up until now, we have been discussing phonological patterns only as restrictions on output forms. While such purely phonotactic investigations have plenty of merit, the focus of this dissertation is instead on how and why specific underlying forms get transformed in the way they do. Returning once again to the Japanese prohibitiion on *[si], for example, we want to know how and why

[4]The choice between /-ru/ and /-u/ for the non-past suffix (among other such choices) is made based off of verb class membership, and the potential suffix puts all verbs into the class demanding /-ru/.

underlying sequences of /si/ are corrected through palatalization instead of some other process. To switch perspectives from phonotactics to processes, we will now go from viewing patterns as formal languages to viewing patterns as string-to-string mappings. Given the input and output alphabets Σ and Δ, a mapping pairs strings in Σ^* with strings in Δ^*. Most of the mappings I will consider in my dissertation will be *functions*, meaning that each $w \in \Sigma^*$ is paired with one $y \in \Delta^*$.

Before making the jump to discussing functions, some additional notation is necessary. A prefix of some string w is any string p such that $w = p \cdot x$ and $p, x \in \Sigma^*$. Note that, as for suffixes, any string is a prefix of itself, and that λ is a prefix of every string. When $|w| \geq n$, $\texttt{pref}^n(w)$ denotes the unique prefix of w with a length of n; when $|w| < n$, it simply denotes w itself. Next, given a set of strings S, I write $\texttt{lcp}(S)$ to denote the *longest common prefix* of S, which is the string u such that u is a prefix of every $w \in S$, and there exists no other string v such that v is also a prefix of every $w \in S$ and $|v| \geq |u|$. For example, $\texttt{lcp}(\{aaa, aab, aac\}) = aa$ since every string begins with aa although each then continues in a different way. Note that the longest common prefix of a set can (and often will be) the empty string λ like in the case of $\texttt{lcp}(\{aaa, baa, caa\}) = \lambda$. Finally, given a function f and a set of input strings $I \subseteq \Sigma^*$, $f(I) = \bigcup_{i \in I}\{f(i)\}$ is the set of all outputs associated to at least one of those inputs.

A first question one might ask about string-to-string functions is whether we can translate one or more of the properties ascribed to the SL languages above into analogous properties that hold over string pairs. Above I talked about the legal continuations of a string w with respect to a language L. It is possible to define a similar notion relative to a function, which are typically called the *tails* of an input string w with respect to a function f and are denoted with $\texttt{tails}_f(w)$. In words, $\texttt{tails}_f(w)$ pairs every possible string $y \in \Sigma^*$ with the portion of the output $f(wy)$ that is *directly attributable* to the input sequence y. The tails of w are, in a sense, the effect that w will have on the output corresponding to a subsequent string of input symbols.

Definition 2.3 Tails (Oncina & Garcia, 1991)**.**
Given a function f and an input $w \in \Sigma^*$:

$$\mathit{tails}_f(w) = \{(y, v) \mid f(wy) = uv \ \wedge \ u = \mathit{lcp}(f(w\Sigma^*))\}$$

To illustrate, take the process of palatalization in Japanese and the five mappings in (3) for the verb root /hos/ 'dry'. Comparing the outputs of this small sample, the longest common prefix shared by the outputs is [ho]. This gives us the reasonable hypothesis that for the function f of Japanese phonology, $\texttt{lcp}(f(\text{hos}\Sigma^*)) = $ [ho]. Now if we subtract this string from the front of each output, we can determine the singular *tail* of /hos/ for each of the suffixes with respect to f. Doing so tells us that the pairs (anai, sanai), (eru, seru), (u, su), (oo, soo), and (imasu, ɕimasu) are all in $\texttt{tails}_f$(hos). This is, of course, just a small cross-section of $\texttt{tails}_f$(hos). To get the entirety of $\texttt{tails}_f$(hos) we would need to, for each possible input starting with /hos/, subtract [ho] from

the front of its corresponding output.

(3) Mappings with the Japanese verb root /hos/ 'dry'

a.	+ /anai/	'NEG'	→	[hosanai]	'does not dry'
b.	+ /e-ru/	'POT-NPST'	→	[hoseru]	'can dry'
c.	+ /u/	'NPST'	→	[hosu]	'dries'
d.	+ /oo/	'VOL'	→	[hosoo]	'let us dry'
d.	+ /imas-u/	'POL-NPST'	→	[hoɕimasu]	'dries (polite)'

The tails of just the input /hos/ do not tell us much about the phonological function of Japanese, but an important generalization becomes apparent when we compare $\texttt{tails}_f(\text{hos})$ with the tails of other input strings ending in /s/ such as the other verb roots /hanas/ 'talk' and /tomos/ 'light (a candle)'. Relative to any verb root ending in /s/, we will have the tail (anai, sanai) for the negative suffix, the tail (eru, seru) for the potential suffix, the tail (u, su) for the non-past suffix, the tail (oo, soo) for the volitive suffix, and the tail (imasu, ɕimasu) for the polite suffix. We have $\texttt{tails}_f(\text{hos}) = \texttt{tails}_f(\text{hanas}) = \texttt{tails}_f(\text{tomos})$ and so on, meaning that all strings ending in /s/ are *tail equivalent* with respect to the Japanese phonology function f. Being tail equivalent with respect to a function is analogous to having the same set of residuals with respect to a language. Using this parallel, Chandlee (2014) and Chandlee et al. (2014) defined the Input Strictly Local (ISL) functions according to tail equivalency as follows.

Definition 2.4 Input Strictly k-Local function (Chandlee, 2014)**.**
A function $f : \Sigma^* \rightarrow \Delta^*$ is ISL$_k$ if and only if for all pairs $w_1, w_2 \in \Sigma^*$ it is the case that:

$$\texttt{suff}^{k-1}(w_1) = \texttt{suff}^{k-1}(w_2) \;\Rightarrow\; \texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$$

Recall from Corollary 2.1 that a language L is SL$_k$ if the residual-equivalence classes of L correspond to suffixes with length $k-1$ (in other words, the residuals are defined by suffixes of length $k-1$). The definitions of the ISL functions are directly parallel to the statement of Corollary 2.1, swapping the notion of residual equivalency with the notion of tail equivalency. Informally, a function f is ISL$_k$ if the tail-equivalence classes of f correspond to $k-1$ suffixes of the input string w. The portion of Japanese phonology seen above is ISL$_2$ since the tails are determined by the most recently read input element, which was /s/ in the above examples.

The definition of the Output strictly Local (OSL) functions is very similar, although requires one additional concept. To define the OSL functions, we need to be able to pick out the portion of the output that corresponds to actual input material. Viewed from another perspective, we need to be able to ignore the portion of the output that would correspond to a word-end symbol. To make this distinction, Chandlee et al. (2015) defined the prefix function f^p associated with a function f as in Definition 2.5. The OSL functions can then be defined as in Definition 2.6.

Definition 2.5 Prefix function (Chandlee et al., 2015)**.**
Given a function f, its associated prefix function f^p is such that:

$$f^p(w) = \mathtt{lcp}(f(w\Sigma^*))$$

Definition 2.6 Output Strictly k-Local function (Chandlee et al., 2015)**.**
A function $f : \Sigma^* \to \Delta^*$ is OSL$_k$ if and only if for all pairs $w_1, w_2 \in \Sigma^*$ it is the case that:

$$\mathtt{suff}^{k-1}(f^p(w_1)) = \mathtt{suff}^{k-1}(f^p(w_2)) \Rightarrow \mathtt{tails}_f(w_1) = \mathtt{tails}_f(w_2)$$

The definition of the OSL functions also adapts the statement of Corollary 2.1 by replacing residual equivalency with tail equivalency. The OSL functions differ from the ISL functions only in that they associate the tail-equivalence classes with suffixes of the prefix function output, instead of with suffixes of the input string. Thus, a function f is OSL$_k$ if the tail-equivalence classes of f correspond to $k-1$ suffixes of f^p. Interestingly, it turns out that the above fragment of Japanese phonology is not OSL$_2$, at least if we are reading input strings from left to right. The final element of f^p(hos) = [ho] is [o] which is also the final element of f^p(hirow) = [hiro], as can be seen from the data in (4) for the verb root /hirow/ 'pick up'. Despite sharing a suffix on the output end, these two verbs have different tails for all of the suffixes. That being said, the function can be OSL$_2$ if we read input strings from right to left. More will be said on directionality in Section 2.2.2.

(4) Mappings with the Japanese verb root /hirow/ 'pick up'

a.	+ /anai/	'NEG'	→	[hirowanai]	'does not pick up'
b.	+ /e-ru/	'POT-NPST'	→	[hiroeru]	'can pick up'
c.	+ /u/	'NPST'	→	[hirou]	'picks up'
d.	+ /oo/	'VOL'	→	[hirooo]	'let us pick up'
d.	+ /imas-u/	'POL-NPST'	→	[hiroimasu]	'picks up (polite)'

The above discussion of the ISL and OSL functions is very abstract, and we would like a way to more concretely visualize their inner workings. Such a view is possible if we take the abstract definitions in terms of function tails and translate them into types of finite-state transducers. The next Subsection introduces the aspects of automata theory necessary for such a characterization.

2.1.3 Finite-state transducers and strict locality

A finite-state transducer (FST) is a machine that produces an output string incrementally by reading an input string one element at a time in a single direction. Such a machine consists of a finite set of states, which can be thought of as a primitive sort of memory, and a finite set of transitions between these states, which are the machine's instructions. The machine begins in one of potentially

many designated initial states, and traverses a path through the state set by following transitions in response to the input that it reads. At any given point of this transduction, the machine will be in some state, and upon reading the next input element, the machine follows a transition out of its current state corresponding to that element. The transition tells the machine what to append to its output, and also the state to which it must move. Transitions are of the format (q, x, y, r) where q is the transition's origin state, $x \in \Sigma$ is the input symbol being read, $y \in \Delta^*$ is the output string to append, and r is the transition's landing state. Additionally, a subset of the machine's states are designated as being final states (also called accepting states), with the remaining states being rejecting states. Each final state is associated to some string $z \in \Delta^*$ that gets appended to the output if the machine is in that state after reading the entire input string. These *final strings* can, of course, be the empty string λ. If the machine is in a final state after reading the full input, the transduction succeeds after appending the state's designated final string, but if the machine ends in a rejecting state, the transduction fails. All of the above is stated formally in Definition 2.7.

Definition 2.7 Finite-state transducer**.**
A finite-state transducer is a 7-tuple $\langle Q, I, F, \Sigma, \Delta, \delta, \phi \rangle$ *where:*

- Q *is a finite set of states*
- $I \subseteq Q$ *is the set of initial states*
- $F \subseteq Q$ *is the set of final states*
- Σ *is the input alphabet*
- Δ *is the output alphabet*
- $\delta \subseteq Q \times \Sigma \times \Delta^* \times Q$ *is the transition relation*
- $\phi \subseteq F \times \Delta^*$ *is the final relation.*

A useful way to visualize FSTs is to use circle diagrams as in Figure 2.2, which computes the Japanese rule of palatalization affecting /s/ before /i/ (how it does so is explained in the next paragraph). Single circles represent rejecting states and double circles represent accepting states, though for the transducers throughout this dissertation, it will generally be the case that every state is a final state. Initial states are marked with an incoming arrow that has no label. By convention, states are labelled with numbers in order to differentiate them, and the final string of an accepting state is added after the label, separated from it by a colon. The labelled arrows represent the transitions and can be interpreted as follows. A transition labelled 'a:b' that starts in state n and ends in state m tells the machine that if it is in state n and reads 'a', it must append 'b' to the output

and move to state m. Note that a transition can start and end in the same state, in which case they are sometimes called loops. For more on FSTs in general, see Mohri (1997).

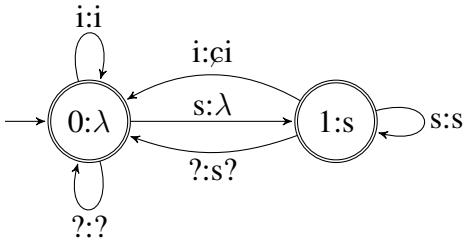


Figure 2.2: A finite-state transducer that computes the palatalization of /s/ before /i/.

How the transducer in Figure 2.2 computes palatalization is as follows, where '?' represents a segment other than the consonant 's' or the vowel 'i'. Being in state 0 corresponds to having read nothing so far (since it is the initial state), or else corresponds to having just read an input element that was not an /s/. If we read an input /?/ or /i/ while in state 0 we output the same thing as we read and remain in state 0. If, however, we read an input /s/ while in state 0, we output nothing and move to state 1 since we do not yet know whether this /s/ will need to palatalize. Being in state 1 corresponds to having just read an input /s/. If we read another /s/ while in state 1, we can output one [s] since we know the previously read /s/ does not immediately precede /i/ and need not palatalize, but we now need to wait and see if this new /s/ will need to palatalize. If the transduction finishes in state 1, we take the input /s/ that we did nothing with on the previous step and output it faithfully as [s] since it turned out to be word-final, and thus not immediately followed by /i/. Doing so is accomplished by appending the state's final string to the output. Reading /?/ while in state 1 will lead us back to state 0, and we will output the delayed /s/ faithfully along with the other symbol we read. Reading /i/ while in state 1 also leads us back to state 0, but we output [ɕi] since the previously-read /s/ ended up needing to be palatalized.[5]

If we do not place any restrictions on the structure of FSTs, we are capable of generating any member of the *regular relations*.[6] These are relational (i.e., input-output mapping) analogues to the regular languages of the Chomsky hierarchy. Similarly to formal language models of phonotactics, it seems that most cases do not require the full capabilities of the regular region when we are modelling phonological processes as maps. More will be said about the potentially excessive power afforded by the full regular region in Section 2.3, but for now let us assume we want to

[5]To avoid over-complicating the discussion, I am abstracting away from the fact that (1) Japanese words cannot end in [s], (2) Japanese does not permit clusters of 3+ consonants, and (3) a sequence of /Vssi/ would become [VɕɕCi].

[6]Mappings that I (and the North American literature) call *regular* are called *rational* in the French literature, who use the term *regular* for an even more powerful class computable by 2-way FSTs (Engelfriet & Hoogeboom, 2001; Filiot & Reynier, 2016). I will follow North American terminological conventions throughout.

restrict ourselves to producing only Strictly Local mappings (i.e., ISL or OSL functions). To do so, we must impose three requirements on FSTs. First, they must be deterministic, meaning that (i) there is just one initial state, (ii) there is only one transition per input element for each state, and (iii) there is only one final string per accepting state. These three sub-requirements guarantee that for a given input string, the machine will always follow the same path through a Strictly Local FST, whereas multiple paths may be possible through a regular FST. The second requirement is that the transducer be *onward*, meaning that the output is never unnecessarily delayed.[7] Put another way, the machine writes as much as it can on each step of a transduction. Formal definitions of determinism and onwardness are provided here for reference.

Definition 2.8 Determinism**.**
An FST is deterministic *if and only if the following hold:*

- *The set of initial states I has a cardinality of one*
- $[(q, \sigma, u, r) \in \delta \wedge (q, \sigma, v, s) \in \delta] \Longrightarrow [r = s \wedge u = v]$
- $[(q, u) \in \phi \wedge (q, v) \in \phi] \Longrightarrow u = v$

Definition 2.9 Transition function closure**.**
Given δ from an FST, its closure δ^* is the smallest set that contains δ and satisfies the following:

- $(q, \lambda, \lambda, q) \in \delta^*$
- $[(q, w, u, q') \in \delta^* \wedge (q', \sigma, v, q'') \in \delta] \Longrightarrow (q, w \cdot \sigma, u \cdot v, q'') \in \delta^*$

Definition 2.10 Onwardness**.**
Let q_0 be the unique initial state of an FST. That FST is onward *if and only if:*

$$(q_0, w, u, q) \in \delta^* \Longleftrightarrow u = \mathtt{lcp}(f(w\Sigma^*))$$

Finally, the third limitation dictates the configurations that states can and must represent. In order to ensure that an FST computes an ISL_k function, the states must correspond to the possible input suffixes with length up to $k-1$, and all transitions must land in the state corresponding to the $k-1$ suffix of the input string seen so far. In order to ensure that an FST computes an OSL_k function, the states must correspond to the possible suffixes of f^p with length up to $k-1$, and all transitions must land in the state corresponding to the $k-1$ suffix of the output string produced so far. Note that according to these latter requirements, the initial state of an SL transducer will correspond to an empty suffix λ.

[7] There will be cases below where the writing is *necessarily* delayed for a finite number of steps.

Definition 2.11 Input Strictly k-Local transducer**.**
A deterministic and onward FST is ISL$_k$ if the following hold:

- $Q \subseteq \Sigma^{\leq k-1}$
- $I = \{\lambda\}$
- $(q, \sigma, u, q') \in \delta \Longrightarrow q' = \mathtt{suff}^{k-1}(q \cdot \sigma)$

Definition 2.12 Output Strictly k-Local transducer**.**
A deterministic and onward FST is OSL$_k$ if the following hold:

- $Q \subseteq \Delta^{\leq k-1}$
- $I = \{\lambda\}$
- $(q, \sigma, u, q') \in \delta \Longrightarrow q' = \mathtt{suff}^{k-1}(q \cdot u)$

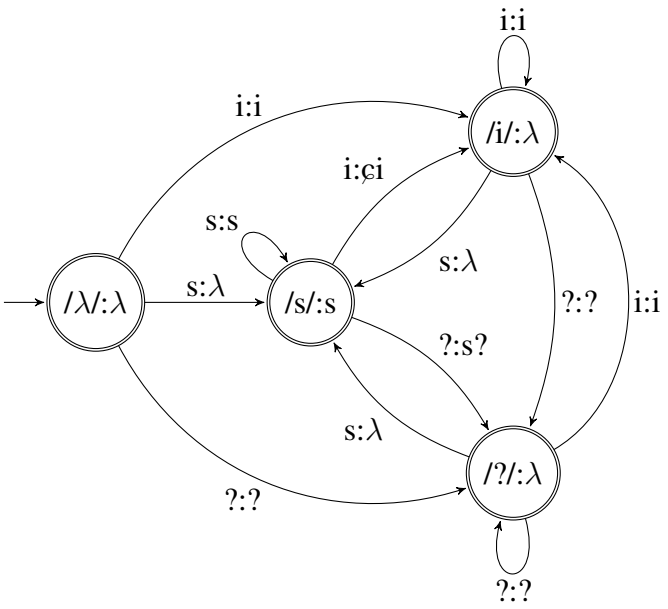


Figure 2.3: An ISL$_2$ transducer that computes the palatalization of /s/ before /i/.

Figure 2.3 shows an ISL$_2$ transducer that produces the same function as the unrestricted transducer in Figure 2.2. For ease of interpretation, I will henceforth label the states of an ISL transducer with the input suffix they correspond to placed between two slashes (e.g., /w/), and label the states of an OSL transducer with the output suffix they correspond to placed between square brackets (e.g., [w]). For transducers that are not SL, I will continue to label their states with numbers, since the states of these machines can represent arbitrary configurations, precluding a transparent label

in most cases. For all transducers, I continue to write the final string (i.e., what gets outputted if the derivation ends in that state) after the state label, separated by a colon. Chandlee (2014) and Chandlee et al. (2014) showed that any ISL_k function can be represented by an ISL_k transducer, and that the function computed by any ISL_k transducer will always be an ISL_k function. In a similar vein, Chandlee et al. (2015) showed that any OSL_k function can be represented by an OSL_k transducer, and that the function computed by any OSL_k transducer will always be an OSL_k function[8]. Accordingly, the next section explores the capabilities of the SL functions by exploring the capabilities of SL transducers.

2.2 Empirical coverage of the Strictly Local functions

The Strictly Local functions are quite versatile and this section will provide an overview of the processes they can handle. While their capabilities overlap to a certain degree, I discuss the Input Strictly Local (ISL) functions separately from the the Output Strictly Local (OSL) functions, since each class can handle patterns that the other cannot. I then close the section by demonstrating that neither the ISL functions nor the OSL functions are capable of modelling long-distance phonological processes.

2.2.1 Input Strictly Local functions

In the sections above, we have seen that ISL functions are capable of modelling local substitutions whose triggering context is to the right (e.g., palatalization of /s/ immediately before /i/). It will come as no surprise, then, that the ISL functions are also capable of modelling local substitutions whose triggering context is to the left. Take for instance the rule of post-nasal voicing in the Puyu Pungo dialect of Quechua (Orr, 1962; Rice, 1993). Data for the pattern are provided in (5). Comparing the pair of words in each row, we can see that the initial obstruents of the genitive suffix /-pa/, the locative suffix /-pi/, and the object suffix /-ta/ become voiced immediately after nasal consonants.

(5) Post-nasal voicing in Puyu Pungo Quechua (Quechua; Orr, 1962; Rice, 1993)

a.	[sinik-pa]	'porcupine-GEN'	[kam-ba]	'you-GEN'
b.	[sa$\widehat{\text{tʃ}}$a-pi]	'jungle-LOC'	[hatum-bi]	'big.one-LOC'
c.	[wasi-ta]	'house-OBJ'	[wakin-da]	'others-OBJ'

[8]Chandlee et al. (2015) prove this using a slightly different type of transducer known as a *delimited* transducer. I will mainly use non-delimited transducers throughout for simplicity, but will introduce delimited transducers in Chapter 7 when they become necessary for machine learning.

A transducer computing this rule is shown in Figure 2.4. To save space and avoid visual clutter, I consider only an input inventory of vowels (represented as 'V'), nasal consonants (represented as 'N') and voiceless obstruents (represented as 'T'). The crucial state in this transducer is the one labelled /N/, which corresponds to having read a nasal consonant on the previous step. If we read an input voiceless obstruent while in this state, we must transform it into the voiced obstruent of the same place (represented as 'T:D' on the transition). All other transitions faithfully reproduce the input symbol being read, and all of the accepting states have the empty string as their final string since nothing special happens at the word edges.

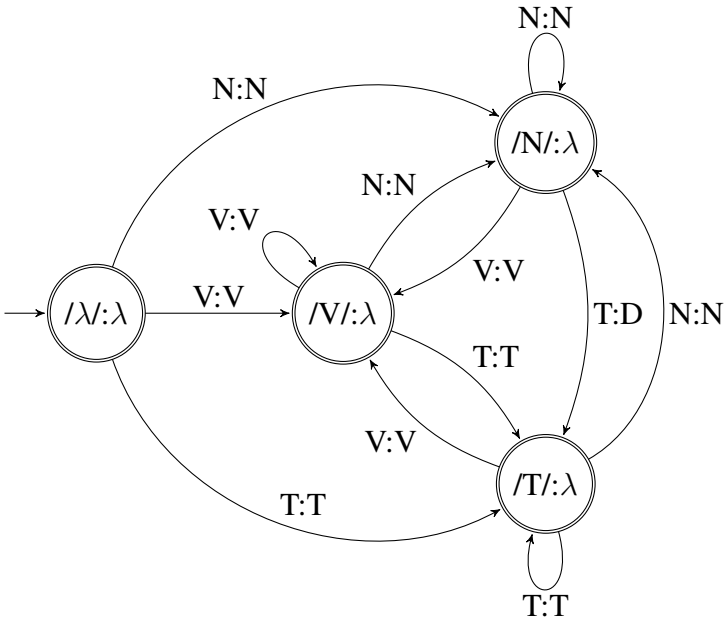


Figure 2.4: An ISL_2 transducer that computes post-nasal voicing.

Notice how whenever this transducer reads an input /T/, we already know whether it was immediately preceded by a nasal consonant, and can transform that /T/ as needed right away. This was not true for the Japanese rule of palatalization, where the transducer had to output nothing upon reading input /s/ in order to wait and see whether a triggering /i/ immediately followed. This type of 'postponing' behaviour is possible in an ISL function because the states correspond to the identity and order of the $k-1$ most recent input symbols. While not literally true, a useful analogy would be to say that the machine can delay the output of an input element by storing it in a finite window of memory.

This ability to delay outputs allows the ISL functions to model rules with two-sided contexts, so long as each side is bounded in length. One such rule comes from Northern Corsican, where voiceless stops become voiced when they appear between two vowels (Dinnsen & Eckman, 1977; Gurevich, 2011). This intervocalic voicing can be seen by the pairs in (6). To know whether

an underlyingly voiceless obstruent should become voiced, we need to know the identity of the single element to its right and the single element to its left. This can be accomplished by an ISL_3 transducer taking advantage of the possibility to postpone outputs.

(6) Intervocalic voicing in Northern Corsican

a.	[peðe]	'foot'	[u beðe]	'the foot'
b.	[tengu]	'I have'	[u dengu]	'I have it'
c.	[kaza]	'house'	[a gaza]	'the house'

An ISL_3 transducer computing this rule is shown in Figure 2.5. Similar to what I did for the post-nasal voicing in Figure 2.4, I consider only an input inventory of vowels (represented as 'V') and voiceless obstruents (represented as 'T') for maximum legibility. Suppose that the machine has just read /V/ for which it has outputted [V]. Depending on what was read beforehand, the machine is either in the state labelled /V/, /VV/, or /TV/. Now suppose that it reads /T/. The machine does not yet know whether this should remain [T] or become its homorganic voiced counterpart [D]. The machine therefore postpones the decision by writing nothing and moves to the state labelled /VT/. Such 'waiting' transitions are marked with dashed lines. If the transducer then reads another /V/ it will output [DV] since the /T/ turned out to be straddled by vowels, but if it reads another /T/ it will output [TT] since the delayed /T/ was preceded but not followed by a vowel and the current /T/ has a non-vowel to its left. If the transduction ends in the state /VT/, the state's final string ensures that the delayed /T/ gets outputted faithfully.

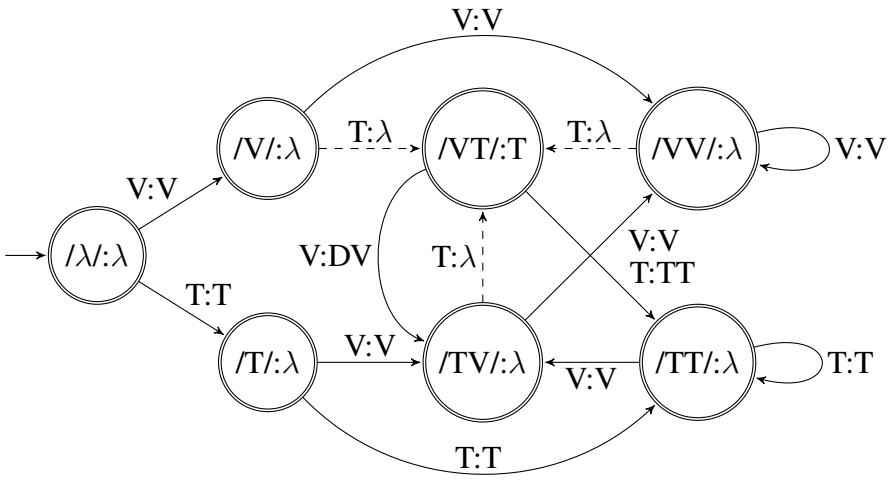


Figure 2.5: An ISL_3 transducer that produces intervocalic voicing.

It is possible to suspend the output fate of an input element until sufficient information has been accrued because transitions can output any string, including the empty string. The availability of such empty transitions also makes it easy to model deletion phenomena. An input segment might

get deleted immediately if the context for deletion is to the left, or it might get deleted because the input element was 'stored' and then never subsequently discharged. An example of the former type comes from Karok, where vowel hiatus is resolved by deleting the second vowel (Bright, 1957; Harris, 2011), demonstrated by the data in (7). An example of the latter type comes from French, where vowel hiatus is resolved by deleting the first vowel (Harris, 2011), demonstrated by the data in (8).

(7) Resolution of vowel hiatus in Karok (second vowel deletes)
   a. /ni-axjar/ → [ni-xjar] 'I fill'
   b. /ni-uksp/ → [ni-kʃup] 'I point'

(8) Resolution of vowel hiatus in French (first vowel deletes)
   a. /lə-avjɔ̃/ → [l-avjɔ̃] 'the airplane'
   b. /lə-pjano/ → [lə-pjano] 'the piano'

An ISL_2 transducer computing the Karok rule is presented in Figure 2.6 and an ISL_2 transducer computing the French rule is presented in Figure 2.7. Each transducer uses 'C' to represent an arbitrary consonant and 'V' to represent an arbitrary vowel. The Karok transducer outputs [V] upon reading /V/ while in the /C/ state, and if it then reads a new vowel while in the /V/ state it will output nothing. No subsequent transitions or final strings in the Karok transducer can recover the suppressed vowel, leaving it forever deleted. The French transducer outputs nothing upon reading /V/ while in the /C/ state, and if it then reads a new vowel while in the /V/ state it will also output nothing. The French transducer will continue to suppress input vowels while in the state /V/ until it either reads a consonant, upon which it produces [VC], or else reaches the end of the transduction, upon which it produces [V]. Only the last vowel in a sequence of adjacent input vowels, then, appears in the output. The preceding vowels of such a vocalic sequence are effectively stored and then forgotten by the French transducer.

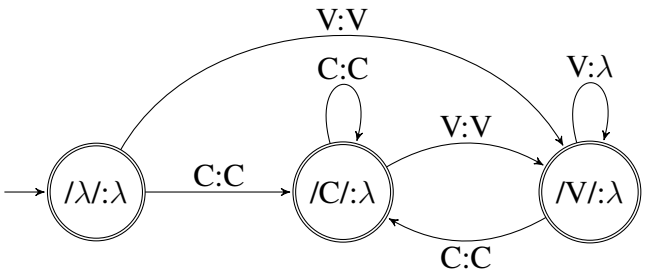


Figure 2.6: An ISL_2 transducer that resolves vowel hiatus by deleting all but the first vowel.

Epenthesis phenomena are equally easy to model using ISL transducers, again because of the freedom afforded to the output edges of transitions. Recall that in order to model substitutions with

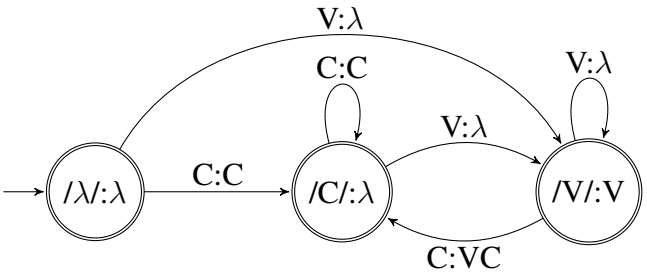


Figure 2.7: An ISL_2 transducer that resolves vowel hiatus by deleting all but the last vowel.

rightward contexts, it was necessary to output nothing on one step and then output two things on the next. Epenthesis occurs when a transition outputs a string longer than a single element and there was no suppression beforehand. We are increasing the length of the input string here, rather than maintaining it as in the case of substitution or decreasing it as in the case of deletion. A familiar instance of epenthesis comes from the plural suffix /-z/ of English. When the suffix attaches to a noun ending in a sibilant consonant, a schwa is inserted to avoid having a cluster of two sibilants as in [haʊs-əz] 'houses' which is never pronounced as *[haʊs-z]. An ISL_2 transducer computing this rule is presented in Figure 2.8, where 'S' represents a sibilant consonant and '?' represents any other sound. While in the /S/ state, the transducer outputs [əS] when it reads a sibilant in order to prevent a sibilant cluster. The schwa on the output edge of this transition is purely epenthetic, since it does not serve as the deferred output of an earlier input element.

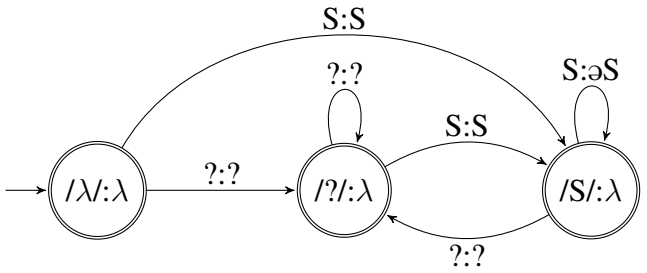


Figure 2.8: An ISL_2 transducer that epenthesizes schwa between two sibilants.

For all of the above examples, the final strings of accepting states have, when not empty, served to faithfully reproduce an element stored on a previous step. This need not be the case, though, since the freedom afforded to the output edges of transitions also applies to final strings. Accordingly, the ISL functions are well-suited to describing substitution, deletion, and epenthesis phenomena that occur specifically at the end of a word. Consider the well-known process of final devoicing in Dutch obstruents, exemplified by the data in (9).

(9) Dutch final devoicing of obstruents

a. /bɛd/ → [bɛt] ‘bed’
b. /bɛd-ən/ → [bɛd-ən] ‘beds’

This rule can be modeled by an ISL_2 transducer shown in Figure 2.9, where ‘V’ stands for a vowel, ‘D’ stands for a voiced obstruent, and ‘T’ stands for a voiceless obstruent.[9] The transducer works very much like the one that models Japanese palatalization in Fig 2.3. Similar to how the Japanese transducer postpones the output of /s/, the Dutch transducer postpones the output of /D/ each time it is read. That being said, the specifically word-final nature of the substitution arises because the Dutch transducer differs from the Japanese transducer in two important ways. First, the transitions out of the /D/ state in the Dutch transducer reproduce the stored /D/ faithfully, whereas one of the transitions out of the /s/ state of the Japanese transducer maps the stored /s/ unfaithfully to its palatal counterpart [ɕ]. Second, the final string of the /D/ state relinquishes the stored /D/ but does so unfaithfully by transforming it to [T]; compare this to the /s/ state of the Japanese transducer whose final string faithfully reproduces the stored /s/ as [s]. Because the transformation occurs only via an accepting state’s final string (i.e., when the transduction ends), the devoicing happens only in word-final position. Similar transducers can be built for cases of word-final epenthesis and word-final deletion. The key suppressions (if any) occur on the way to some state whose designated string fails to do anything with the stored element(s) in the deletion case or whose designated string outputs more than what was read in the epenthesis case. For reasons of space I will not provide explicit examples.

Rounding off the demonstration of ISL functions, a final type of process that they can model is bounded methathesis, although this requires some additional assumptions. Chandlee (2014, chapter 4) showed that bounded metathesis is ISL if we analyze it in two steps. Specifically, a copy of an element is epenthesized to a nearby location by a first function, and the output of that function acts as the input to a second function that deletes the original instance of the copied element, creating the illusion of displacement. Since the two sub-processes of metathesis (epenthesis and deletion) have been covered already in this section, I do not illustrate metathesis here. More detailed arguments for the insert-then-delete analysis of metathesis and an illustration thereof can be found in Chandlee (2014, chapter 4).

2.2.2 Output Strictly Local functions

Earlier in Section 2.1.3, I mentioned that the Japanese process of palatalization before /i/ is not OSL_2 when we are reading the input from left-to-right. This holds for any rule with a right-hand context and for any value of k, since the states of an OSL transducer have no direct relationship

[9] Similar to the Japanese FST above, this FST abstracts away from the fact that there is a maximum number of consonants that can appear in a cluster, and that word-internal clusters would have consistent voicing.

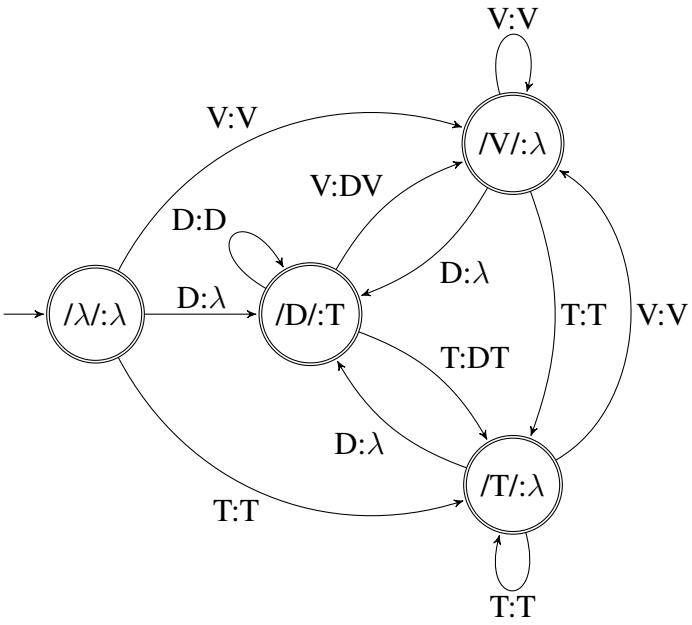


Figure 2.9: An ISL_2 transducer that computes word final devoicing of obstruents.

to the input string. Without a direct link between states and information about the identity/order of the most recent input elements, we cannot take advantage of the states to pursue the 'wait and see' strategy that was possible in ISL transducers. Repeating the analogy of memory storage, a machine computing an OSL function does not 'remember' anything about the input string since it is tracking information about the output string, and so can never delay its decisions.

Processes that have purely left-hand contexts can, however, be computed by OSL transducers reading from left to right. For example, the OSL_2 transducer in Figure 2.10 computes the Puyu Pungo Quechua rule of post-nasal voicing from (5) in Section 2.2.1. Like the ISL_2 transducer for this rule in Figure 2.4 on page 20, the input alphabet consists of Vowels ('V'), nasal consonants ('N'), and voiceless obstruents ('T'). Also like the ISL_2 transducer, the OSL_2 transducer follows a transition labelled 'T:D' when it reads an input /T/ after having produced [N] by reading /N/. The differences are that the OSL transition originates from the state labelled [N] and lands in the state labelled [D], whereas the ISL transition originated from the state labelled /N/ and lands in the state labelled /T/. The differences arise because the ISL transducer needs to be in the state corresponding to the most recently *read* element and the OSL transducer needs to be in the state corresponding to the most recently *produced* element. Despite these differences, though, the two transducers compute the exact same function.

This is not to say that it is fully impossible for OSL functions to model rules with purely right-hand contexts. To model such processes, we can specify that a given OSL transducer reads input strings backwards. The output string will then also be backwards, and reversing the output

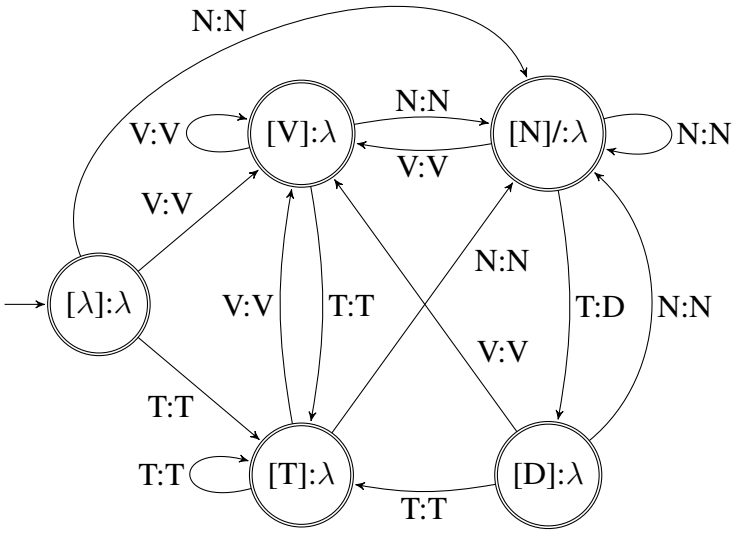


Figure 2.10: A left-to-right OSL_2 transducer that computes post-nasal voicing.

string after completing the transduction gives us the intended result. To illustrate, the transducer in Figure 2.11 models the Japanese palatalization of /s/ before /i/ if we assume that it reads input strings from right to left. The crucial transition is the one labelled 's:ɕ' out of the state labelled [i]. The palatalization looks like it is applying *progressively* in this transducer (i.e., /s/ seems to palatalize to [ɕ] when it *follows* [i]), and the intended *regressive* nature of the palatalization can be obtained by reversing the completed output string.

The fact that processes can apply either progressively or regressively is typically captured in the computational literature by allowing FSTs to read from left-to-right or from right-to-left (Heinz & Lai, 2013; Chandlee et al., 2015). The OSL functions that read left to right (Left-OSL functions) and those that read from right to left (Right-OSL functions) are distinct but overlapping classes: some patterns are both Left-OSL and Right-OSL while other patterns are Left-OSL but not Right-OSL or vice versa. Case in point, the Quechua process is a Left-OSL function but not a Right-OSL function, and the Japanese process is a Right-OSL function but not a Left-OSL function. I will often refer to the two classes collectively as the OSL functions, but will specify the directionality of individual functions and transducers. Interestingly, reading direction has no consequences for the ISL functions, since the Left-ISL and Right-ISL functions are fully equivalent (Chandlee, 2014). The actual FSTs for the Left-ISL and Right-ISL analysis of a single process may end up looking different, but they will be extensionally equivalent.

Both the ISL and OSL functions can model processes that have purely left-hand and purely right-hand contexts. Nonetheless, it remains that only the ISL functions can model processes

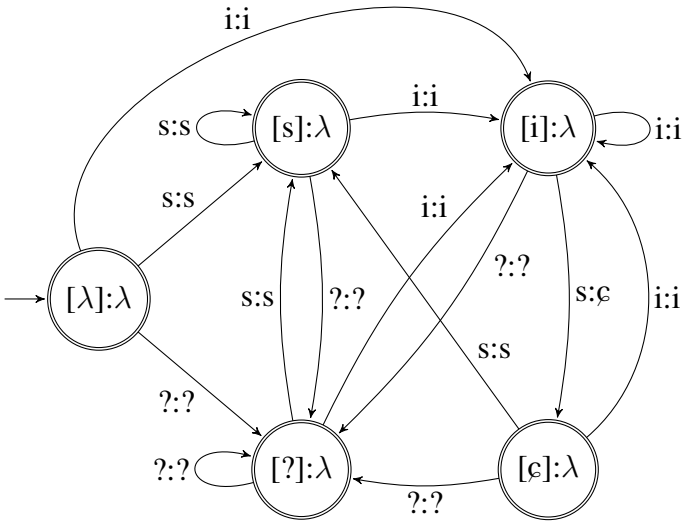


Figure 2.11: A right-to-left OSL_2 transducer that computes palatalization of /s/ before /i/.

with two-sided contexts. This might seem to shed doubt on the usefulness of the OSL functions, but there is one well-attested behaviour that is OSL but not ISL, namely iterative spreading. The decision to continue spreading a phonological feature throughout a line of segments depends on the identity of the most recent output element *regardless of its input specification*. A prime example comes from Johore Malay, which iteratively applies a rule of nasal assimilation where sonorant segments become nasal if they are immediately preceded by a nasal segment (Onn, 1980). The rule keeps applying until the spreading nasality reaches the end of the word or encounters an oral stop, with the output of one iteration acting as the input to the next iteration. Importantly, a nasal vowel will trigger the rule whether or not it was originally nasal. Relevant data from Onn (1980, p. 45) are presented in (10). For simplicity's sake, I will assume that nasal vowels do not exist in underlying forms and will ignore the blocking of spreading by oral consonants. Neither of these decisions significantly affect the analyses below.

(10) Johore Malay progressive nasal spreading

a.	/minum-an/	→	[mĩnũmãn]	'drink (refreshment)'
b.	/makan-an/	→	[mãkanãn]	'food'
c.	/pəŋ-awas-an/	→	[pəŋãw̃ãsan]	'supervision'
d.	/baŋun-an/	→	[baŋũnãn]	'building'

Figure 2.12 shows an ISL_2 transducer that attempts but fails to model the Johore Malay pattern, where 'V' stands for an oral vowel, 'Ṽ' stands for a nasal vowel, and 'N' stands for a nasal consonant. The ISL_2 transducer will correctly apply the first instance in a chain of desired nasal

spreads, but will fail to spread the nasality any further. Reading /V/ while in the state labelled /N/ brings us to the state labelled /V/, meaning that we are erroneously treating the vowel as oral for the purposes of the rule. Of course, we could achieve iterative spreading by stipulating that the transducer may re-read its own outputs as new input strings until no further changes are made, but this will encounter some problems. Unlike the regular relations, the ISL functions do not have a property known as *closure under composition*, meaning that given two ISL functions f and g (which may or may not be distinct), the function obtained by applying g to the output of f is not guaranteed to also be ISL (Chandlee et al., 2018). Allowing the ISL transducer to apply iteratively, then, could inflate the generative capacity of the resulting pattern, and so a solution that applies an SL function only once (i.e., uses a single pass through an SL transducer) is more desirable.

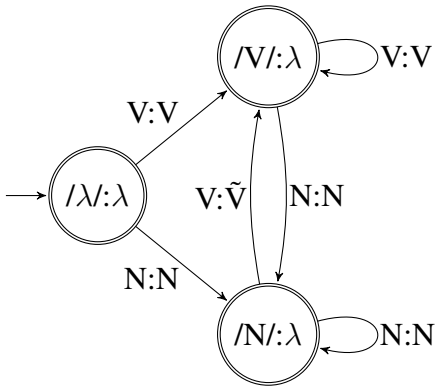


Figure 2.12: An ISL_2 transducer that produces non-iterative nasal assimilation.

As it turns out, a single pass through the OSL_2 transducer in Figure 2.13 is sufficient to model the iterative nature of the nasal spreading. States in an OSL_2 transducer correspond to the most recent output element, so the transducer will correctly treat a nasalized vowel as nasal for the purposes of the rule. A noteworthy observation is that an 'opposite' pattern exists in Capanahua, where there is iterative and regressive nasal spreading blocked by oral consonants (Loos, 1969; Rose & Walker, 2011; Walker, 1998). Data for this pattern are provided in (11), where sources of nasality are underlined. The regressive version of this same pattern can easily be achieved by having the transducer in Figure 2.13 read from right to left.

(11) Regressive nasal spreading in Capanahua (Rose & Walker, 2011)

a.	ʔõn̲ãm̲pãn̲	'I will learn'
b.	põj̃ãn̲	'arm'
c.	kajatãn̲aiʔ	'I went and jumped'
d.	tʃipõn̲ki	'downriver'

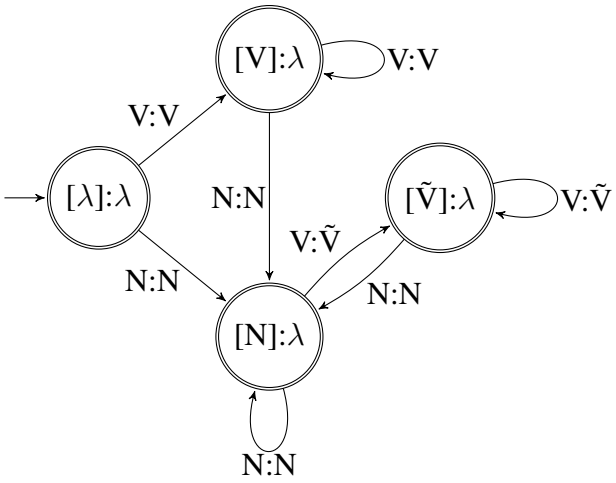


Figure 2.13: An OSL_2 transducer that produces iterative nasal assimilation.

2.2.3 Inability to model long-distance behaviour

Their differences aside, the ISL and OSL functions together capture an impressive range of attested phonological patterns in a rather intuitive and computationally simple manner. Specifically, the output fate of an input element depends only on a very limited type of information: the identity and order of elements in an adjacent window with a fixed maximum size. The ISL functions can model substitution, epenthesis, and deletion provided that the context of the transformation fits into a bounded window, which may or may not straddle the affected segment. They can additionally model bounded metathesis if we split the one process into two, adopting a 'copy then delete' strategy (Blevins & Garrett, 1998, 2004; Chandlee & Heinz, 2012; Chandlee, 2014). The OSL functions can also model local substitution, epenthesis, and deletion provided that the triggering context is bounded in length and lies entirely to one side of the affected segment. More importantly, the OSL functions can model local iterative spreading, which falls outside of the ISL class.

The crucial limitation of the ISL and OSL functions which this thesis aims to address is their inability to model *long-distance* processes such as consonant harmony and dissimilation, in which the trigger can be arbitrarily far away from its target. To show that something cannot be modelled by the SL functions, we can adopt a strategy like the one used in Section 2.1.1 to show that the result of Bukusu liquid harmony was not an SL stringset. There, we compared strategic pairs of words to establish that the set of permissible surface forms did not exhibit Suffix Substitution Closure. In the function case, we can compare strategic pairs of mappings to establish that a function's tail-equivalency classes do not satisfy the definition of an ISL function or OSL function.

Consider the two mappings /xam-ila/ $\rightarrow$ [xamila] 'milk-APPL' and /rum-ila/ $\rightarrow$ [rumira] 'send-APPL'. It is reasonable to assume that $\texttt{lcp}(f(\text{xami}\Sigma^*)) = f^p(\text{xami}) = [\text{xami}]$ and that $\texttt{lcp}(f(\text{rumi}\Sigma^*)) = f^p(\text{rumi}) = [\text{rumi}]$. From these facts we can determine that the pair (la, la) is

in $\texttt{tails}_f$(xami) and that the pair (la, ra) is in $\texttt{tails}_f$(rumi). Consequently, the Bukusu function cannot be ISL_3 since the inputs /xami/ and /rumi/ share an input suffix of length 2, and yet have different sets of function tails. Furthermore, the Bukusu function cannot be OSL_3 since f^p(xami) and f^p(rumi) share a suffix of length 2, and yet have different sets of function tails. Additionally, we can insert an arbitrarily long string z of non-liquid consonants into the inputs to yield a pair /xa·z·mi/ and /ru·z·mi/ which also have different tails despite sharing an input suffix and a suffix of f^p each with an arbitrary length. The Bukusu rule of liquid harmony is therefore not ISL_k or OSL_k for *any* value of k.

Where the ISL and OSL functions go wrong here is that they need harmony and dissimilation triggers to fall into a bounded window in order to apply their effects. The effects of these triggers, however, can in principle apply across any distance. My dissertation will argue that this particular shortcoming of the SL functions can be overcome using different but closely related classes of functions that relativize locality through the use of tiers. A tier is a subset of the relevant alphabet, and only elements belonging to the tier take up space in the function's window. In this way, two tier elements may be separated by an arbitrary number of non-tier elements in a string, but will nevertheless be adjacent on the tier projection of that string. A harmony or dissimilation trigger can thus stay in the function's window and exert its influence across a string of non-tier elements because these never get the chance to push the trigger out of the window. The next chapter will more thoroughly show how tiers can be incorporated into the SL functions. Before that, though, the next section outlines an existing approach to long-distancce harmony and dissimilation which will often act as a benchmark for the suitability of my proposals.

2.3 Subsequential functions

Transducers that are SL (and, therefore, the SL functions) cannot handle long-distance processes because the restrictions imposed upon their structure are too severe. One might suggest, then, that as much as we would like to use strictly local transducers when modeling human phonology, we must abandon strict locality when modeling long-distance patterns. As stated previously in Section 2.1.3, unrestricted FSTs generate the class of regular relations. The hypothesis that the class of regular relations reflects the range of possible phonological patterns is attractive since the regular region excludes a type of process known as *majority rules* harmony which is widely regarded as being pathological. In a majority rules process, determining the outcome requires comparing the number of occurrences of one segment type versus another within the same underlying form. For example, in majority rules backness vowel harmony, a stem would surface containing only front vowels if front vowels are more numerous than back vowels in the underlying form, but would surface containing only back vowels if back vowels are more numerous than front vowels in the

underlying form. Human phonology does not seem to track relative frequency in this way, so the fact that the regular hypothesis excludes majority rules behaviour is a welcome result.

More recently, though, long-distance phonological processes have been argued to instantiate *subsequential functions*, a class of functions which is properly contained by the class of regular relations (every subsequential process is also regular, but not vice-versa). A function is subsequential if and only if it divides Σ^* into a finite number of tail-equivalence classes (Oncina & Garcia, 1991; Oncina et al., 1993). This makes them a strict superset of the SL functions, who also divide Σ^* into a finite number of tail-equivalence classes, but additionally require a specific relationship between those classes and string suffixes. For their part, the tail-equivalence classes of a subsequential function can correspond to arbitrary configurations.

Subsequential functions can also be defined in terms of the finite-state machines that compute them (Mohri, 1997). Recall that for FSTs to generate only SL functions they had to be deterministic and onward, and their states had to correspond to suffixes of up to a specified length. The automata characterization of the subsequential functions keeps the requirement of determinism but discards the requirement of onwardness and places no restrictions on what states may represent. In other words, the subsequential functions are those that can be computed by a deterministic FST. Similarly to the OSL functions, reading direction matters for subsequential functions, meaning that there are actually two overlapping but distinct types of subsequential function. The Left-subsequential functions are those computable by transducers reading from left to right and the Right-subsequential functions are those computable by transducers reading from right to left. Like what I do for the OSL functions, I will mostly refer to the forward-reading and backward-reading variants collectively as simply the subsequential functions.

As support for using subsequential functions as a model of non-local phonological behaviour, several types of long-distance processes have been shown to be subsequential, namely vowel harmony (Gainor et al., 2012; Heinz & Lai, 2013), consonant harmony (Luo, 2017), and consonant dissimilation (Payne, 2017). Subsequential transducers model long-distance harmony and dissimilation by providing a designated state for each harmonic class. This is shown for Bukusu liquid harmony in Figure 2.14, where '?' represents a non-liquid segment. If the first liquid we read is /r/ we go to state 1. This state corresponds to having produced [r] somewhere previously, which compels all subsequent liquids to surface as [r]. Conversely, if the first liquid we read is /l/ we instead go to state 2. This state corresponds to having produced [l] somewhere previously, which compels all subsequent liquids to surface as [l].

A further argument in favour of using subsequential functions parallels the discussion of majority rules harmony above; a pathological process which Wilson (2003, 2006) calls *sour grapes* harmony (adapting a term from Padgett (1995)) is regular but not subsequential (Heinz & Lai, 2013). In a sour grapes process, a harmony-triggering segment will spread its harmonic feature

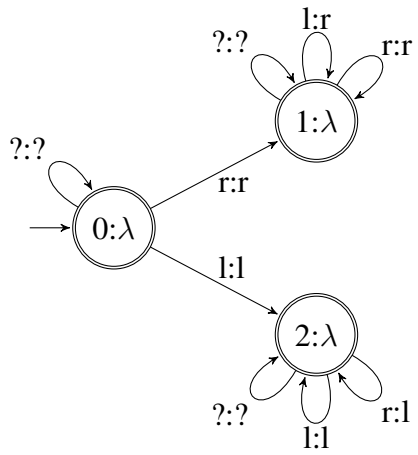


Figure 2.14: A subsequential FST that computes liquid harmony.

to other segments only if it can affect all possible targets; if any one of these targets would resist being changed, the spreading never gets initiated. In other words, the potential harmony trigger first looks ahead to see whether it can affect everything, and preemptively gives up if it sees any antagonistic elements down the line. For example, where '$+$' represents a harmony trigger, where '$-$' represents a harmony target, and where '$\times$' represents a harmony resistor, a sour grapes process would map /$+ - - - -$/ to $[+ + + + +]$ and map /$+ - - \times -$/ to $[+ - - \times -]$. We normally expect the latter input to get mapped to something like $[+ + + \times -]$ with spreading that stops at the resistor or something like $[+ + + \times +]$ with spreading that goes through the resistor. Human phonology does not seem to have the unbounded lookahead capabilities necessary for sour grapes behaviour, and so the fact that the subsequential hypothesis excludes sour grapes behaviour is a welcome result.

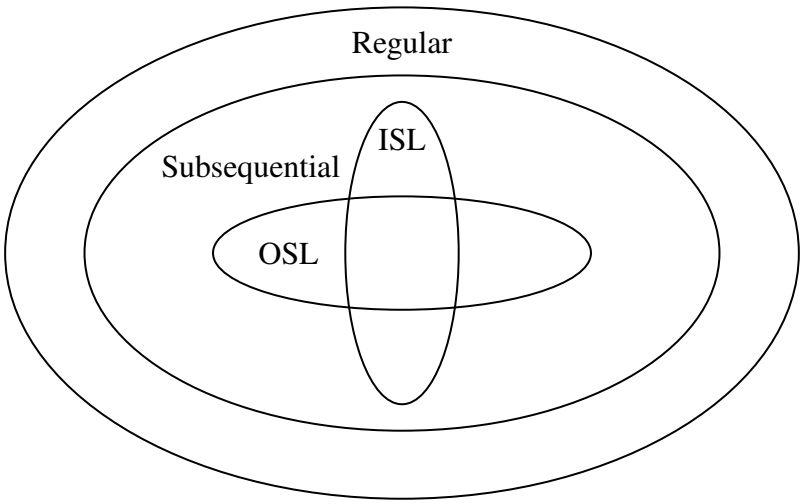


Figure 2.15: The current subregular hierarchy of relations

Subsequential functions act as a compromise between the undergenerating SL functions and the overgenerating regular relations, illustrated by the diagram in Figure 2.15. The regular relations completely contain the subsequential functions, which in turn completely contain the ISL and OSL functions; the ISL and OSL functions overlap but not completely. Majority rules patterns lie outside the regular region, and sour grapes patterns lie in the space between the regular and subsequential borders. As I will discuss in Chapter 4, however, the subsequential functions can still model some pathological behaviours, and I therefore ask whether we can increase our coverage past the SL barrier to encompass long-distance phenomena without needing the full capabilities of the subsequential functions. This dissertation offers just such a compromise: incorporating tiers into the structure of an SL function increases our empirical coverage all while excluding some pathological subsequential patterns.

2.4 Chapter summary

I began this chapter by introducing the SL languages and functions, which form the base of the proposals and discussions in the following chapters. Then, after illustrating how the SL functions can be described by a particular type of FST, I demonstrated how phonological processes of substitution, epenthesis, deletion, metathesis, and iterative spreading fall into the SL class, provided that the triggering contexts fit into a window of fixed and finite length. I furthermore showed that the fixed window size makes it impossible to model long-distance harmony and dissimilation as an SL function. Finally, I outlined how the subsequential functions can be used to generate long-distance harmony and dissimilation. The performance of this last approach will serve as a litmus test for my own proposals in future chapters.

Chapter 3

Tier-based Strictly Local functions

The transformations performed by a long-distance phonological process can be arbitrarily far apart from their triggers, placing such processes outside the capabilities of Strictly Local (SL) functions. This is not to say, however, that long-distance processes operate in a completely different way from local ones. They still display locality effects, but rather than operating according to strict adjacency, these operate according to a looser, relativized notion of adjacency where only relevant segments matter. For example, in sibilant harmony, it is the most recent sibilant that dictates the behaviour of the process at each point in a derivation, rather than the most recent segment. Indeed, if we take an output string that is the product of sibilant harmony and erase all non-sibilant segments, every harmony target will be adjacent to its harmony trigger. The use of *tiers* is a long-standing method of relativizing locality in this way, and this chapter will show how tiers can be incorporated into the SL functions to capture long-distance application. I start by providing some background on the use of tiers in phonology and formal language theory before defining and characterizing the Tier-based Strictly Local (TSL) functions. Following that, I present the various capabilities of TSL functions using real language examples. Finally, I discuss some aspects of long-distance phonological systems that the TSL functions cannot handle. Subsequent chapters will offer extensions of the TSL class that alleviate some of these shortcomings.

3.1 Tiers

3.1.1 Outside of computational phonology

In early rule-based frameworks,[1] a phonological form was a string of feature-value matrices, and long-distance processes were captured by allowing statements with the form "zero or more in-

[1] Such as the one presented in Chomsky & Halle's (1968) foundational book *The Sound Pattern of English.*

stances of x". Consider the Turkish pattern of backness harmony, where suffix vowels taken on the backness value of the rightmost vowel in the base (Clements & Sezer, 1982). Data for the pattern are provided in (12), where the plural suffix alternates between front [-ler] and back [-lar] and the genitive suffix alternates between front [-in] and back [-ɯn]. This harmony rule might be stated as in (13), where a superscript zero indicates that any number of some matrix (including zero) can occupy that position. I assume here that a feature [±syllabic] distinguishes between vowels and consonants, the former being [+syllabic].

(12) Turkish backness vowel harmony

a.	[ip-ler]	'rope-PL'	[ip-ler-in]	'rope-PL-GEN'
b.	[ev-ler]	'house-PL'	[ev-ler-in]	'house-PL-GEN'
c.	[pul-lar]	'stamp-PL'	[pul-lar-ɯn]	'stamp-PL-GEN'
d.	[son-lar]	'end-PL'	[son-lar-ɯn]	'end-PL-GEN'

(13) $$\left[+\text{syllabic} \right] \longrightarrow \left[\alpha \text{ back} \right] \Big/ \left[\begin{array}{l} +\text{syllabic} \\ \alpha \text{ back} \end{array} \right] \left[-\text{syllabic} \right]^0 \_\_\_\_$$

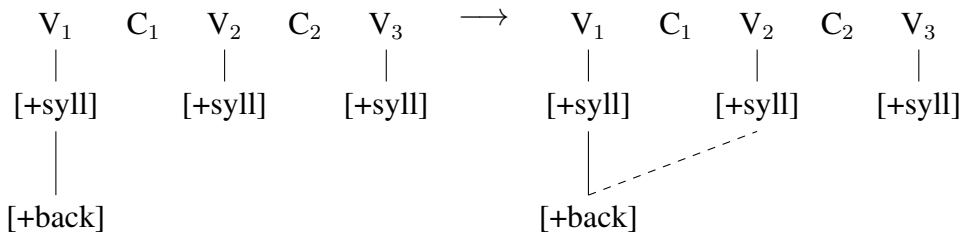
This gets the job done, but doesn't quite reflect the intuition that intervening material is ignored; supposedly irrelevant material needs to be explicitly encoded into the rule's structural description. The subsequent development of autosegmental phonology—originally developed by Goldsmith (1976) for the analysis of tonal phenomena—allowed for a more principled means of excluding irrelevant material, however, and the approach remains influential to this day. As Rose & Walker (2011) put it, the autosegmental approach "directly or indirectly underlies many present-day treatments of particular harmony patterns", and the present dissertation is no exception.

Autosegmental analyses achieve long-distance application by taking the one-dimensional matrix strings of earlier rule-based frameworks and giving them a more sophisticated structure. All instances of some feature exist as an ordered string on a plane or *tier* dedicated to that feature alone, and the various tiers are connected by association lines between feature-value pairs. The tiers are typically organized into a hierarchy such that feature-value pairs or *nodes* on one tier can be associated only to nodes on the immediately dominating tier. Several such hierarchies or *feature geometries* have been proposed (e.g., Clements, 1985; Sagey, 1986; McCarthy, 1988; Clements & Hume, 1995) though it is outside the scope of this dissertation to present any of them in full detail. All tiers ultimately depend on the *root* tier, whose nodes represent segmental timing slots; the pronunciation of a root node is then the phone that corresponds to the collection of feature-value pairs that can be linked back to that root node. A final key component of autosegmental analyses is that segments can be unspecified for a feature in the underlying form. Segments are assumed to be fully specified in the output, so any missing specifications are filled in by the application

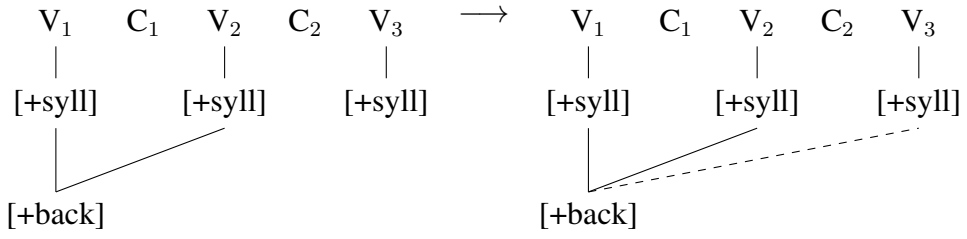
of a phonological process, or else by a rule that inserts some default value at the very end of a derivation.

An autosegmental analysis of backness vowel harmony is depicted in (14) and (15). For expository purposes, I assume that the feature [±syllabic] dominates the feature [±back] and that consonants are unspecified for [±syllabic], receiving [−syllabic] by a later default insertion rule. In the underlying form, only the first vowel is specified for the feature [±back]. The backness specification of this first vowel then spreads iteratively to the remaining nodes on the dominating [±syllabic] tier. It first notices that the node adjacent to its dominating node has nothing depending on it, and so associates itself to that empty node. This step is shown in (14). Upon doing so, it then notices that a node adjacent to one of its dominating nodes is also empty, and so associates itself to that other empty node. This step is shown in (15). As a result of the spreading, all non-initial vowels will match the initial vowel's backness in the output form. Importantly, autosegmental spreading only targets adjacent nodes. While the vowels may not be adjacent on the root tier, they *are* adjacent on the tier dominating the feature [±back]. Consonants are unaffected because they are not linked to any node on the relevant tier, and so are invisible to the spreading process when it applies.

(14)



(15)



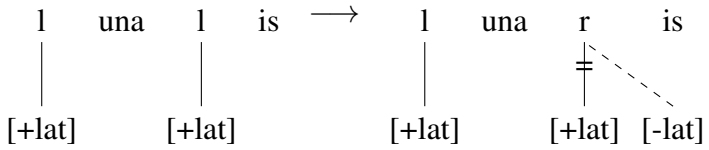
Autosegmental analyses are equally capable of describing long-distance dissimilation, although the mechanisms employed are somewhat different from those used in harmony. Consider the pattern of liquid dissimilation in Latin, where the suffix /-alis/ becomes [-aris] if it would be preceded at any distance by an [l]. Data for the pattern are provided in (16). This and other dissimilatory phenomena arise due to a ban on adjacent and identical instances of some feature. For historical reasons, the banning of adjacent and identical feature-value pairs is known as the Obligatory Contour Principle (OCP). The Latin data suggest that the language bans contiguous sequences of [+lateral][+lateral]. From the autosegmental perspective, all instances of [±lateral] will form a

contiguous string on the [±lateral] tier, and an underlying form like /lun-alis/ will thus contain the illicit sequence. The OCP demands that this configuration be corrected, and so we replace the second instance of [+lateral] with an instance of [−lateral], resulting in a change from /l/ to [r]. The replacement is shown pictorially in (17) by striking through the original association line and adding a dashed association line to a new feature.

(16) Latin liquid dissimilation

a.	/nav-alis/	→	[navalis]	'naval'
b.	/popul-alis/	→	[popularis]	'popular'
c.	/lun-alis/	→	[lunaris]	'lunar'

(17)



The specifics of an autosegmental analysis can vary, since such analyses are strongly influenced by the chosen theory of phonological features and their geometry. Questions of exact implementation aside, though, the autosegmental approach can be distilled down to the following. First, a phonological form consists of multiple and connected levels of representation. Second, a given phonological element (e.g., a feature or segment) can be present on one level of representation but absent on another. Third, material that is non-local on one level may be local on another level. Finally, only material that is local at the relevant level matters when applying a process. It is this core set of principles that the following two sections will expand upon within the computational context. I first show how the concept of an autosegmental tier has been defined from the perspective of formal language theory. I then take that conceptualization of a tier and use it to define a class of string-to-string functions capable of modelling long-distance phonological processes.

3.1.2 In the context of formal languages

Despite the wide-ranging interest in tiers and their utility, it is only recently that their expressive power has been explored in the context of formal language theory (Heinz et al., 2011; Jardine & Heinz, 2016; McMullin, 2016; McMullin & Hansson, 2016; Jardine & McMullin, 2017; Lambert & Rogers, 2020). From this perspective, a tier T is simply a subset of the alphabet Σ, and we can suppress the non-tier elements of a string w by applying an erasure function $\texttt{erase}_T$ that is defined relative to T. The function call $\texttt{erase}_T(w)$ returns w with all non-tier elements removed. A formal definition of this function is provided for completeness. The result of erasing the non-tier elements from a string w is often called the *projection* of w onto the given tier. More sophisticated

methods of projection that consider surrounding local material in addition to segment identity have been explored (Graf & Mayer, 2018; Mayer & Major, 2018; De Santo & Graf, 2019), although I will limit myself in this dissertation to the maximally simple means of projection where segment identity is the only determining factor.

Definition 3.1 Erasure function**.**
Given an alphabet Σ and a tier $T \subseteq \Sigma$, the erasure function *imposed by T on Σ^* is as follows:*

$$\begin{aligned} \texttt{erase}_T(\lambda) &= \lambda \\ \texttt{erase}_T(w) &= \texttt{erase}_T(u) \cdot \sigma \quad \textit{if} \quad [w = u \cdot \sigma] \wedge [\sigma \in T] \\ \texttt{erase}_T(w) &= \texttt{erase}_T(u) \quad \textit{if} \quad [w = u \cdot \sigma] \wedge [\sigma \notin T] \end{aligned}$$

A Tier-based Strictly k-Local (TSL$_k$) language is then a subset of Σ^* that is SL$_k$ once the erasure function has applied to all strings. In other words, a TSL language will ban a given string if that string's projection onto the specified tier contains any impermissible contiguous sequences. Note that the TSL$_k$ languages are a direct extension of the SL$_k$ languages: any SL$_k$ language is also a TSL$_k$ language where $T = \Sigma$. Furthermore, like the SL$_k$ languages, the TSL$_k$ languages can be defined positively or negatively.

Definition 3.2 Tier-based Strictly k-Local language (positive)**.**
A language L is TSL$_k$ if and only if there is a tier $T \subseteq \Sigma$ and a grammar $G \subseteq T^{\leq k}$ such that:

$$L = \{w \in \Sigma^* \mid \texttt{fac}_k(\texttt{erase}_T(w)) \subseteq G\}$$

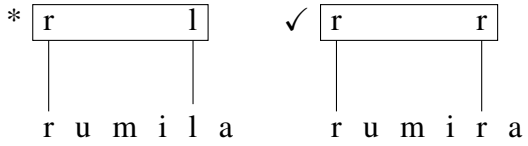
Definition 3.3 Tier-based Strictly k-Local language (negative)**.**
A language L is TSL$_k$ if and only if there is a tier $T \subseteq \Sigma$ and a grammar $G \subseteq T^{\leq k}$ such that:

$$L = \{w \in \Sigma^* \mid \texttt{fac}_k(\texttt{erase}_T(w)) \cap G = \emptyset\}$$

Recall the pattern of liquid harmony in Bukusu, where underlying /l/ becomes output [r] if the nearest leftward surface liquid is [r] (Odden, 1994; Hansson, 2010). The data provided earlier for this pattern are repeated in (18). The result of liquid harmony is TSL$_2$ with the tier $T = \{\text{r}, \text{l}\}$ and a negative grammar $G = \{\text{rl}\}$. The input /rum-ila/ is prevented from surfacing as *[rumila] since the tier 2-factor *[rl] is banned and $\texttt{erase}_T$(rumila) =[rl]. The harmonized form [rumira] is chosen instead since it does not include any illegal tier 2-factors. We can visualize the projections and assessments of banned *[rumila] and acceptable [rumira] as in (19).

(18) Bukusu liquid harmony (Odden, 1994)
   a. xam-ila ‘milk-APPL’
   b. lim-ila ‘cultivateAPPL’
   c. kar-ira ‘twist-APPL’
   d. rum-ira ‘send-APPL’

(19) Visualization of Bukusu liquid harmony as a TSL_2 language



A good question to ask at this point is how an autosegmental analysis of liquid harmony would compare to the TSL analysis just presented. Both make a distinction between a level of representation where the whole phonological string is present and a level of representation where only the liquid consonants of the phonological string are present. Also in both approaches, the prohibition against a contiguous sequence of [−lateral][+lateral] is enforced on the less-inclusive level of representation. Where the approaches differ concerns the way these levels of representation are defined. An autosegmental approach might postulate, for example, that there is a tier containing all and only the instances of the feature [±lateral]. Assuming that only liquid consonants can be specified for the feature [±lateral], the entities present on this [±lateral] tier will then always correspond to an [l] or [r] in the full phonological string. For its part, the TSL approach arbitrarily stipulates that there is a level of representation that includes all and only the instances of [l] and [r].

While inspired by the autosegmental approach, the TSL perspective does not follow it to the letter, allowing us to gain insight into the formal underpinnings of the structures of long-distance phonology with minimal theoretical commitments. Although the tier of a TSL language can in principle be an arbitrary set of segments, TSL analyses are by no means incompatible with approaches that define tiers according to perceptual similarity (as in the Agreement by Correspondence framework; see e.g. Rose & Walker 2004 and Hansson 2010) or according to feature geometry (Clements & Hume, 1995). Many of the tiers used in the computational literature (and in the analyses below) happen to form a natural class relative to typical feature theories, and one can of course decide to limit themselves only to feature-definable tiers, but there is formally no such requirement. However one chooses to restrict the types of tiers that are possible, that choice will not change the fact that TSL languages over those tiers still model the intuition that a non-local pattern can hold locally at an appropriate representational level. Despite their name, the TSL languages do not themselves provide a theory of possible tiers; rather, they provide a model of the effect that a tier (motivated or not) has upon the assessment of locality.

In light of the relatively simple way that the TSL languages generalize the SL languages, it is perhaps unsurprising that a property quite similar to Suffix Substitution Closure (SSC) directly follows from their abstract characterization. The property, which I call Tier-based Suffix Substitution Closure (T-SSC), is presented in Theorem 3.1 which is an adaptation of a similar theorem from Lambert & Rogers (2020). Informally, T-SSC states that non-tier elements can be freely inserted

and deleted without affecting grammaticality and also states that when two grammatical strings have middle sections whose tier strings are the same and have a length of at least $k-1$, we can substitute the suffixes that come after the middle sections without causing ungrammaticality.

Theorem 3.1 Tier-based Suffix Substitution Closure (Lambert & Rogers, 2020)**.**
A stringset L is TSL$_k$ for the tier T if and only if the following hold:

- *It is closed under insertion and deletion of elements not in T: $w \in L \iff \texttt{erase}_T(w) \in L$*
- *For any two strings $u_1 \cdot x_1 \cdot v_1$ and $u_2 \cdot x_2 \cdot v_2$ such that $\texttt{erase}_T(x_1) = \texttt{erase}_T(x_2)$ and $|\texttt{erase}_T(x_1)| = |\texttt{erase}_T(x_2)| \geq k-1$ it is the case that:*

$$[u_1 \cdot x_1 \cdot v_1 \in L \ \wedge \ u_2 \cdot x_2 \cdot v_2 \in L] \ \Rightarrow \ u_1 \cdot x_1 \cdot v_2 \in L$$

The SSC property of the SL$_k$ languages was particularly useful because it meant that we could determine the legal continuations of a legal string by looking only at its last $k-1$ elements. This corollary of SSC served as the base for defining the SL functions. Fortunately, the T-SSC property of the TSL$_k$ languages has a directly parallel corollary that will let us define a class of TSL functions in the next section. As expressed in Corollary 3.1, any two grammatical strings from a TSL$_k$ language whose tier-strings (i.e., the result of applying $\texttt{erase}_T$) end in the same $k-1$ or more symbols can be legally continued by the exact same set of strings. For convenience, I will write $\texttt{suff}^n_T(w)$ to mean $\texttt{suff}^n(\texttt{erase}_T(w))$ in what follows.

Corollary 3.1 Tier-based suffix-defined residuals**.**
A stringset L is TSL$_k$ for the tier T if and only if for all pairs $w_1, w_2 \in \Sigma^*$:

$$\texttt{suff}^{k-1}_T(w_1) = \texttt{suff}^{k-1}_T(w_2) \ \Rightarrow \ \{v \mid w_1 \cdot v \in L\} = \{v \mid w_2 \cdot v \in L\}$$

Consider again the Bukusu requirement of liquid harmony. We've determined the result of this rule to be TSL$_2$, which means that any two strings whose tier strings end in the same one symbol (or more) will have the same legal continuations. Take the two verb stems /kar-/ 'twist' and /rum-/ 'send' whose tier strings are both /r/. The verb stems both take the [ira] allomorph of the applicative suffix /-ila/ and cannot appear with the [ila] allomorph because this would result in the tier string [rl] which is prohibited in Bukusu. Indeed, any string that can be added to the end of the string [kar] can be added to the end of the string [rum] (and vice versa) precisely because they share a tier suffix with length $k-1 = 2-1 = 1$. The next section shows how this notion of Tier-based Suffix-defined Residuals can be used to define and characterize the TSL functions so that they parallel the SL functions in the desired way.

3.1.3 In the context of string-to-string maps

The fact that the SL_k languages were fruitfully extended to the ISL_k and OSL_k functions raises the question of whether the TSL_k languages can be similarly extended to functions in order to model non-local processes. Recall that the definitions of the ISL_k and OSL_k functions rely on the fact that the legal continuations of any string w in an SL_k language can be inferred simply by looking at its $k-1$ suffix. The definitions of the SL_k functions take this corollary of SSC and adapt it for functional tails instead of string continuations. I showed in the previous section that a similar property holds of the TSL_k languages. Namely, the legal continuations of any string w in a TSL_k language can be inferred simply by looking at the $k-1$ suffix of $\texttt{erase}_T(w)$. This corollary of T-SSC can thus be used to define classes of ITSL_k and OTSL_k functions according to how they partition Σ^* into tail-equivalence classes.[2] The definitions here are based off of the definition of the OTSL_k functions in Burness & McMullin (2019), though similar definitions exist in Hao & Bowers (2019) and Hao & Andersson (2019).

Definition 3.4 Input Tier-based Strictly k-Local function**.**
A function $f : \Sigma^* \rightarrow \Delta^*$ is ITSL$_k$ if and only if there is a tier $T \subseteq \Sigma$ such that for all pairs $w_1, w_2 \in \Sigma^*$, it is the case that:

$$\texttt{suff}_T^{k-1}(w_1) = \texttt{suff}_T^{k-1}(w_2) \Rightarrow \texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$$

Definition 3.5 Output Tier-based Strictly k-Local function**.**
A function $f : \Sigma^* \rightarrow \Delta^*$ is OTSL$_k$ if and only if there is a tier $T \subseteq \Delta$ such that for all pairs $w_1, w_2 \in \Sigma^*$, it is the case that:

$$\texttt{suff}_T^{k-1}(f^p(w_1)) = \texttt{suff}_T^{k-1}(f^p(w_2)) \Rightarrow \texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$$

Informally, a function f is ITSL_k if the tail-equivalence classes of f correspond to $k-1$ input *tier* suffixes (where the tier is a subset of the input alphabet Σ), and a function f is OTSL_k if the tail-equivalence classes of f correspond to $k-1$ output *tier* suffixes (where the tier is a subset of the output alphabet Δ). The fact that the definitions of the TSL functions are effectively the definitions of the SL functions modulo the inclusion of a tier permits an equally parallel automata-theoretic characterization using finite-state transducers.[3] For an ITSL_k transducer, the states are labelled with the possible $k-1$ input tier suffixes, and a transition must land in the state corresponding to

[2]These functions are not to be confused with the ITSL and OTSL *languages* of Graf & Mayer (2018), Mayer & Major (2018), and De Santo & Graf (2019). The acronyms ITSL and OTSL consistently refer to the function classes throughout the rest of this dissertation.

[3]A word of caution: the image of a TSL function (i.e., the language formed by collecting all of its possible outputs) is not necessarily a TSL language, just as the image of an SL function is not necessarily an SL language (Chandlee, 2014). The Karajá pattern analyzed in Section 3.2.4 is a TSL function that does not produce a TSL language.

the $k-1$ tier suffix of the input string seen so far. For an OTSL$_k$ transducer, the states are labelled with the possible $k-1$ output tier suffixes, and a transition must land in the state corresponding to the $k-1$ suffix of the output tier string produced so far. Like I did for SL transducers in the previous chapter, I will label the states of an ITSL transducer with the input tier suffix they correspond to placed between two slashes (e.g., /w/), and label the states of an OTSL transducer with the output suffix tier they correspond to placed between square brackets (e.g., $[w]$). As was the case for the exact correspondence between ISL$_k$ functions and ISL$_k$ transducers, any ITSL$_k$ function can be computed by an ITSL$_k$ transducer and any ITSL$_k$ transducer computes an ITSL$_k$ function (*mutatis mutandis* for OTSL$_k$ functions and transducers). See Section B.1 for a formal proof of this fact.

Definition 3.6 Input Tier-based Strictly k-Local transducer**.**
A deterministic and onward FST is ITSL$_k$ for the tier $T \subseteq \Sigma$ if the following hold:

- $Q \subseteq T^{\leq k-1}$
- $I = \{\lambda\}$
- $(q, \sigma, u, q') \in \delta \Longrightarrow q' = \mathtt{suff}_T^{k-1}(q \cdot \sigma)$

Definition 3.7 Output Tier-based Strictly k-Local transducer**.**
A deterministic and onward FST is OTSL$_k$ for the tier $T \subseteq \Delta$ if the following hold:

- $Q \subseteq T^{\leq k-1}$
- $I = \{\lambda\}$
- $(q, \sigma, u, q') \in \delta \Longrightarrow q' = \mathtt{suff}_T^{k-1}(q \cdot u)$

Figure 3.1 shows an OTSL$_2$ transducer that enforces the long-distance liquid harmony pattern in Bukusu. Once again, it is instructive to compare how the TSL and autosegmental approaches compute liquid harmony. The autosegmental approach would postulate something like the following. First, there is a tier containing all and only the instances of the feature [±lateral] and this [±lateral] tier is dominated by the [+liquid] tier (i.e., every instance of [±lateral] depends on some [+liquid] node). Second, there is then a spreading rule where the leftmost instance of [±lateral] checks to see if there is a [+liquid] node immediately to the right of the rightmost [+liquid] node dominating itself. If such a "target" node exists, the instance of [±lateral] will associate itself to the target node, replacing any preexisting specification for [±lateral] that depended on the target. For its part, the OTSL$_2$ transducer is agnostic as to why [l] and [r] form a tier to the exclusion of other output elements, is agnostic as to why the tier segments affect the output correspondent(s) of certain input, and is agnostic as to how this influence is exerted. While the operations being computed by the transducer can be interpreted as the result of a spreading rule, the transducer does

not literally model spreading. It simply represents the map that results from the influence exerted by the tier elements and does not care for the reasons behind tier membership and tier influence.

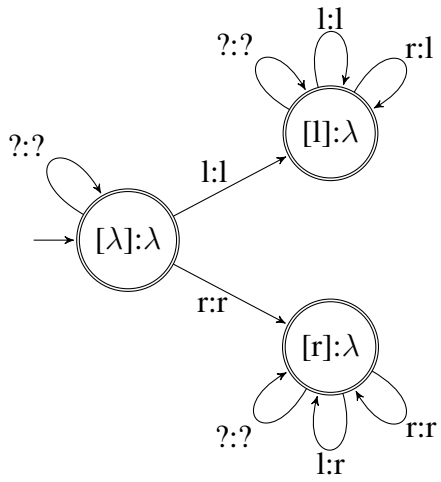


Figure 3.1: An OTSL_2 transducer that computes liquid harmony.

Lack of explanatory power might seem to be a downside for the TSL approach, but one must remember that its goal is to characterize the *computational* properties of the maps, rather than their underlying causes. There are multiple mechanisms that can produce liquid harmony (e.g., autosegmental spreading and agreement-by-correspondence) but the exact mechanism chosen will not change the fact that liquid harmony can be modelled as a TSL functions. The next sections will show how most other long-distance phonological maps also fall within either the class of ITSL functions, the class of OTSL functions, or both.

3.2 Capabilities of TSL functions

By virtue of the way that they are defined, the TSL functions inherit all the capabilities of the SL functions while adding several new possibilities. Investigations of long-distance phonological processes typically focus on the behaviour of individual segments, and so most of this section demonstrates how various segmental behaviours are captured by the TSL functions. In order, I analyze segmental transparency, parasitic harmony, segmental blocking, and icy targets using real language examples. The section then discusses iterativity and symmetry (or lack thereof). Following that, the section closes with some remarks on behaviours that can be modeled by adopting higher values for the k parameter.

3.2.1 Transparency

A well-studied aspect of long-distance phonological processes is the non-participation or *transparency* of material between the trigger and target (see, e.g., Archangeli & Pulleyblank, 2007; Gafos & Dye, 2011; Rose & Walker, 2011; Finley, 2017). Consider once again the Turkish pattern of backness harmony, where suffix vowels take on the backness value of the rightmost vowel in the base (Clements & Sezer, 1982). Consonants do not participate and are seemingly 'skipped over' when determining appropriate suffix allomorphs (though see Section 4.3.3 for interesting counterexamples). Data for the pattern are repreated in (20).

(20) Turkish backness vowel harmony

a.	[ip-ler]	'rope-PL'	[ip-ler-in]	'rope-PL-GEN'
b.	[ev-ler]	'house-PL'	[ev-ler-in]	'house-PL-GEN'
c.	[pul-lar]	'stamp-PL'	[pul-lar-ɯn]	'stamp-PL-GEN'
d.	[son-lar]	'end-PL'	[son-lar-ɯn]	'end-PL-GEN'

TSL functions are well-suited for modelling this inertness, and an OTSL_2 transducer computing Turkish vowel harmony is depicted in Figure 3.2.[4] The tier for this function is all and only the output vowels, though for reasons of space, the transducer collapses all the front vowel states together and collapses all back vowel states together. The transitions are equally collapsed such that a transition labelled '[αbk]:[βbk]' means that an input vowel specified as [αback] gets mapped to its [βback] equivalent with all other features unchanged. Turkish allows disharmonic bases, and some suffixes do not alternate, instead remaining faithful and starting a new domain of harmony. For example, the genitive suffix /-gen/ always appears as [-gen] and causes any subsequent vowels to be front. Accordingly, I assume that harmony can only fill in missing backness values, marking under-specification as [0bk].[5] The transducer models the intuition that a vowel underspecified for [$\pm$back] will take on the [$\pm$back] value of the closest specified vowel to its left. Consonant transparency is captured in this transducer by the fact that reading and producing a consonant never causes a change of state. More generally, an element is transparent to a TSL function if it is in the alphabet but not a member of the tier. Any transition which outputs only non-tier elements can never cause a change of state, since such a transition does not affect the tier suffix of the string.

The transparency of consonantal segments in vowel harmony is a matter of some debate. Some researchers adopt a "local spreading" approach where the harmonic feature spreads onto and through consonants via coarticulation, (e.g., Ní Chiosáin & Padgett, 2001; Jurgec, 2011) rather than skipping them. Transparent segments would then be those for which the harmonizing

[4] The exact same transducer is also ITSL_2.

[5] One can account for such morpheme-specific information in this way since a function's input and output alphabets are not required to be the same (in fact they can in principle be fully disjoint).

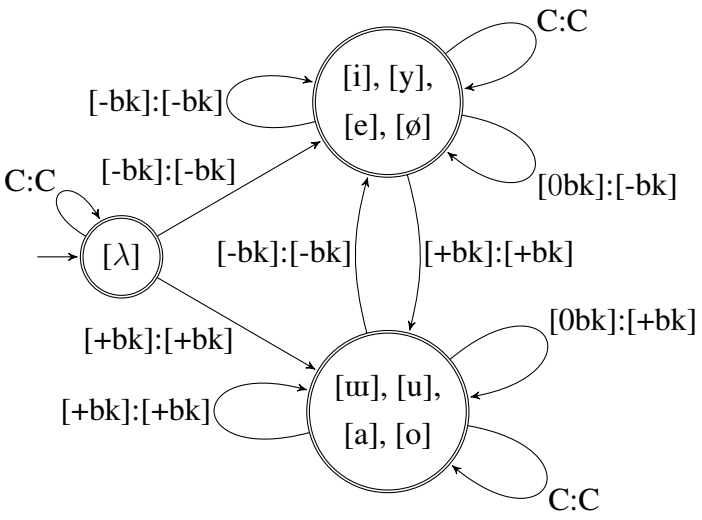


Figure 3.2: An OTSL_2 transducer that produces Turkish backness harmony.

feature does not carry any perceptual or contrastive force. Support for this idea comes from research on the reportedly transparent vowels of Hungarian. Like Turkish, Hungarian has backness harmony, although the non-low front vowels [i] and [e] are unaffected by the harmony and neither trigger nor block it (Gafos & Dye, 2011). Phonetic studies reveal, however, that instances of [i] and [e] between two back vowels are produced slightly back, albeit to a sub-phonemic and likely imperceptible level (Benus et al., 2004; Gafos & Benus, 2006; Benus & Gafos, 2007).

Some other harmony processes, however, are not fully amenable to a local spreading analysis. For example, the usual [d] of the perfective suffix /-idi/ in Yaka becomes nasal if there is a nasal obstruent in the root (Hyman, 1995; Walker, 2000; Rose & Walker, 2004, 2011; Archangeli & Pulleyblank, 2007; Jurgec, 2011). The root in (21a) contains no nasal consonant, so the suffix surfaces faithfully. Compare this to the roots in (21b-c) which *do* have a nasal consonant. No matter whether this nasal consonant is in the adjacent syllable (21b) or several syllables away (21c), it forces the suffix to harmonize and surface as [-ini]. This harmony for the feature [±nasal] occurs across vowels and voiceless obstruents without nasalizing them to any extent (Rose & Walker, 2004). We must instead allow nasality to skip over them, which we can do by excluding them from an appropriate OTSL function's tier.

(21) Yaka nasal consonant harmony (Hyman, 1995)

a.	/tsub-idi/	[tsub-idi]	'roam-PERF'
b.	/tsum-idi/	[tsum-ini]	'sew-PERF'
c.	/nutuk-idi/	[nutuk-ini]	'lean.on-PERF'

Transparency in the Yaka pattern is no more or less difficult to describe using a TSL function

than transparency in Turkish or Hungarian vowel harmony. This is because the TSL class places no limits on which collections of segments can act as a tier. As I mentioned earlier in Sections 3.1.2 and 3.1.3, the TSL languages and functions are agnostic as to *why* the tier takes a particular shape. I believe that this freedom of analysis is advantageous since it is not always possible to derive a segment's tier membership from external factors such as the shape of the phoneme inventory. For example, the low vowel /a/ undergoes tongue root harmony in Kinande (Cole & Kisseberth, 1994; Gick et al., 2006) while it resists and blocks tongue root harmony in Pulaar (Archangeli & Pulleyblank, 1994), even though there are no relevant differences between the languages that can explain this discrepancy (Rose & Walker, 2011).

It is also worth pointing out that while the local spreading approach can be invoked for at least some cases of harmony, it is unclear how local spreading would deal with cases of long-distance dissimilation such as in Latin. Dissimilation is often (but not always) generated using theoretical mechanisms much different from those that generate harmony, though this does not change the fact that both harmony and dissimilation are TSL in terms of complexity. A TSL account of liquid harmony and a TSL account of liquid dissimilation look strikingly similar, the only difference between them boiling down to the content of their respective transducers' transitions. For the liquid harmony pattern, reading an input /l/ or /r/ while in the [l] state will cause one to produce a [l] and loop back to the [l] state, whereas for the dissimilation pattern we would produce [r] and move to the [r] state. In both cases, vowels and non-liquid consonants are transparent by virtue of being excluded from the tier, making it impossible for them to cause a change of state.

3.2.2 Parasitic harmony

Another behaviour that has received plenty of attention in the phonological literature is so-called *parasitic* harmony, where the source and target interact on one dimension only when they agree along another. The most prominent cases of parasitic harmony involve rounding harmony depending on vowel height. Take for instance the Kachin dialect of Khakass. Suffix high vowels in this language become round if the nearest leftward vowel is also high (Korn, 1969; Kaun, 1995). The data in (22) show a [+high] suffix vowel harmonizing with [+high] vowels (a and b) but failing to harmonize with [−high] vowels (c and d). As the additional data in (23) show, a [−high] suffix vowel never harmonizes, even if the preceding vowel is also [−high].

(22) Kachin Khakass rounding harmony in [+high] suffixes (Korn, 1969, 102-103)

a.	/kyn-ni/	[kyn-ny]	'day-ACC'
b.	/kuʃ-tɯn/	[kuʃ-tun]	'of the bird'
c.	/ød-ir/	[ød-ir]	'to kill'
d.	/ok-tɯn/	[ok-tɯn]	'of the arrow'

(23) Kachin Khakass lack of harmony in [−high] suffixes (Korn, 1969, 102-103)

a.	/kyn-ge/	[kyn-ge]	‘to the day’
b.	/kuzuk-ta/	[kuzuk-ta]	‘in the nut’
c.	/tʃør-gen/	[tʃør-gen]	‘who went’
d.	/pol-za/	[pol-za]	‘if he is’

Let us consider how the OTSL_2 transducer in Figure 3.3 models the pattern. The tier includes all vowels, so the machine is always in the state corresponding to the most recently read vowel. For readability, the two [+high, +round] states are collapsed together, as are the states for the six remaining vowels. While in a [+high, +round] state (i.e., the [y] or [u] state), reading /i/ will produce [u] and reading /ɯ/ will produce [u]. The same vowels are produced faithfully while in any other state. By coordinating the work of the tier and the work of the transitions, we can ensure that (i) only high vowels are ever affected by vowel harmony and (ii) they are only so affected when the preceding vowel is also high.

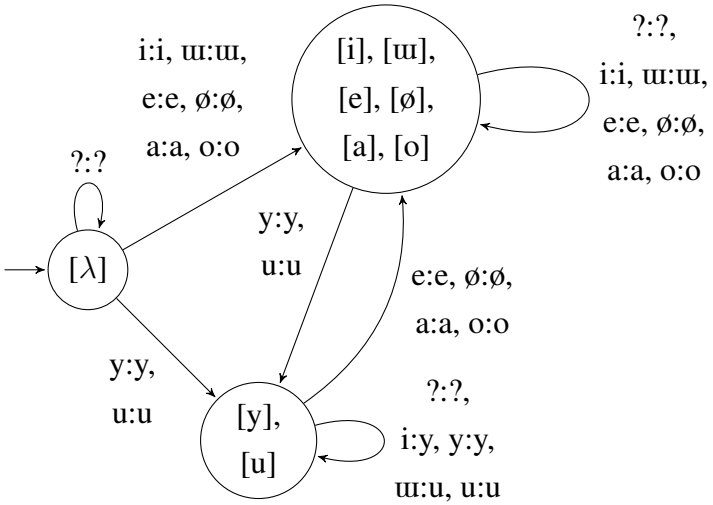


Figure 3.3: An OTSL_2 transducer that produces Kachin Khakass parasitic rounding harmony

Unlike transparency, which is a direct consequence of being excluded from the tier, parasitism has no direct cause in TSL models. Instead the parasitic nature of a harmonic process is more of a coincidence, with unfaithful transitions conveniently arranged such that changes (rounding in the above case) occur only when the target has a certain value for another feature or matches the trigger along some other dimension (height in the above case). In fact, it is just as easy to make a TSL transducer that computes an unattested *anti-parasitic* system where the target must *not* match the trigger along some specified dimension. Similar to how standard formal language theoretic definitions do not prevent hodgepodge collections of elements from forming a tier, ostensibly bizarre patterns are inevitable in any approach that uses FSTs unless we constrain the input and output

alphabets as well as the operations that can be performed by a transition. That being said, I do not believe that this kind of overgeneration is decisively problematic for the hypothesis that phonology is TSL. The TSL hypothesis concerns itself only with the computational properties of phonological patterns, and does not purport to explain the phonetic naturalness of patterns.[6] If one wants to adopt the TSL hypothesis but also account for rule naturalness, they could introduce auxiliary modules such as a theory of proper alphabets and a theory of proper transitions. Alternatively, one could adopt a point of view like that of Blevins's (2004) theory of Evolutionary Phonology where phonetic naturalness has purely diachronic origins and does not influence the form or content of synchronic grammars. Arguing for either one of these possibilities lies well outside the scope of this dissertation, so I will largely ignore the question of phonetic naturalness in what follows.

3.2.3 Blocking

Rounding harmony in Khalkha Mongolian is very similar to rounding harmony in Kachin Khakass but also exhibits what are often called *blocking* effects, the next type of segmental behaviour I would like to discuss from the perspective of the TSL hypothesis. As in Kachin Khakass, rounding harmony is parasitic on height, in this case requiring the target and trigger to both be non-high (Svantesson et al., 2005; Nevins, 2010; Gafos & Dye, 2011). The harmony causes alternations in a variety of suffixes, such as the comitative, data for which are shown in (24). Note that there is a simultaneous process of harmony for the feature [±ATR]. The high front vowel /i/ is transparent to harmony (Nevins, 2010): it does not become rounded when preceded by a round vowel of any height, but also does not prevent rounding from reaching a following [−high] vowel. This is shown in (25) using words with the accusative and reflexive suffixes. The other high vowels /u/ and /ʊ/, however, *do* prevent rounding from reaching a following [−high] vowel, as can be seen from the words in (26) where the causative suffix comes between a root and the perfective suffix.

(24) Khalkha Mongolian rounding harmony, comitative suffix (Nevins, 2010, p. 139)

a. [nar-tai] 'sun-COM'
b. [ner-tei] 'name-COM'
c. [ɔd-tɔi] 'star-COM'
d. [ʊːl-tai] 'mountain-COM'
e. [xun-tei] 'person-COM'

(25) Khalkha Mongolian rounding harmony, transparency of /i/ (Svantesson et al., 2005, p. 50)

[6] Notably, the TSL class excludes sour-grapes-like and majority rules patterns (among others), whose undesirability is purely computational in nature (see Section 4.4 for further discussion).

a. [poːr-ig-o] ‘kidney-ACC-REFL’
b. [xɔːlʒ-ig-ɔ] ‘food-ACC-REFL’
c. [mʊːr-ig-a] ‘cat-ACC-REFL’
d. [suːlʒ-ig-e] ‘tail-ACC-REFL’

(26) Khalkha Mongolian rounding harmony, blocking by /u, ʊ/ (Nevins, 2010, p. 137)
a. [tor-oːd] ‘be.born-PERF’
b. [ɔr-ɔːd] ‘enter-PERF’
c. [tor-uːl-eːd] ‘be.born-CAUS-PERF’
d. [ɔr-ʊːl-aːd] ‘enter-CAUS-PERF’

I will use the word *blocking* to refer to this situation where certain intervening segments do not undergo a process and prevent the process from occurring across them, and use the word *blocker* for the intervening segments that block the process. *Opacity* and *opaque segment* are sometimes used instead of *blocking* and *blocker*, although I will not adopt this nomenclature since it can be confused with *opacity* in the context of rule ordering. Blocking is quite easy to generate within a TSL_2 transducer, only requiring us to include both triggers and blockers on the tier and ensure that the transitions out of a blocker's state are faithful along the appropriate dimension. For example, the Khalkha pattern (abstracting away from the ATR dimension) can be modelled by the OTSL_2 transducer in Figure 3.4 whose tier contains all and only the rounded vowels. Consonants and [i] are off the tier and so always cause a loop. Reading the mid vowel /e/ while in the [o] state will produce [o] and loop back, no matter how many consonants or instances of [i] have been produced since the [o] that put us in the [o] state. If at any point we produce an [u] by reading /u/, we move to the [u] state, out of which all transitions are faithful. No rounding harmony will take place while in the [u] state, even if we were previously in the [o] state. Rounding harmony can only attempt to apply again if a new instance of [o] is produced, taking us out of the [u] state and back to the [o] state.

Blocking effects are not limited to vowel harmony. They are present in at least one case of consonant harmony, namely regressive sibilant harmony in Slovenian. The process in Slovenian is optional, though I will assume for the purposes of this section that it is categorical, postponing a discussion of its optionality to Chapter 6. The Slovenian pattern causes a sibilant to become [−anterior] if it is followed at any distance by another [−anterior] sibilant *unless* a coronal stop intervenes (Jurgec, 2011, pp. 329-333). Examples of successful sibilant harmony are provided in (27a-b) and examples of blocked sibilant harmony are provided in (27c-d) with the second singular prefix. Blocking effects are also sometimes seen in unbounded dissimilation. In Georgian, /r/ dissimilates to [l] when preceded by another /r/ at any distance unless /l/ intervenes (Fallon, 2001; Odden, 1994). Examples (28c-d) show the demonymic suffix /-uri/ appearing as [-uli] due to a

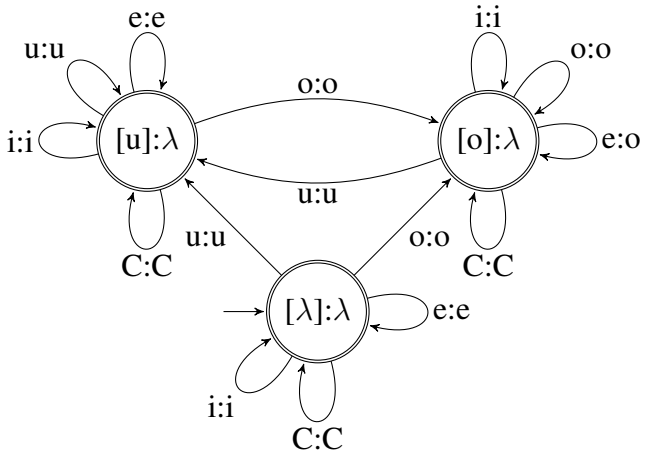


Figure 3.4: An OTSL_2 transducer that computes Khalkha Mongolian rounding harmony

preceding /r/. In example (28e), the suffix is preceded by an /r/, but an /l/ intervenes between the two instances of /r/ and so there is no dissimilation.

(27) Slovenian sibilant harmony, blocking by coronal stops (Jurgec, 2011, pp. 330-331)

a.	/spi-ʃ/	[ʃpi-ʃ]	'sleep-2SG'
b.	/pozabi-ʃ/	[poʒabi-ʃ]	'forget-2SG'
c.	/stoji-ʃ/	[stoji-ʃ]	'stand-2SG'
d.	/zida-ʃ/	[zida-ʃ]	'build-2SG'

(28) Georgian /r/ dissimilation, with blocking by /l/ (Fallon, 2001; Odden, 1994)

a.	/dan-uri/	[dan-uri]	'Danish'
b.	/p'olon-uri/	[p'olon-uri]	'Polish'
c.	/ungr-uri/	[ungr-uli]	'Hungarian'
d.	/aprik-uri/	[aprik-uli]	'African'
e.	/kartl-uri/	[kartl-uri]	'Kartvelian'

Blocking can also be morphologically or lexically driven. I mentioned earlier in Section 3.2.1 that some suffix vowels in Turkish are invariant like the vowel in the nominalizer suffix /-gen/ which is always [−back] (Nevins, 2010, pp. 33-34). The /e/ in /-gen/ acts as a blocker in the sense that it stops the spread of [+back] by remaining [−back], but differs from the cases of blocking above by starting a new harmony domain and causing all following vowels to take on [−back]. As I showed, one way of ensuring this kind of behaviour in a TSL function is to make a distinction between input vowels unspecified for some feature and input vowels pre-specified for the same feature; the former will take on a value for the empty feature based on the most recent tier element (i.e. based on the current state), but the latter will maintain the specification that they already have.

3.2.4 Icy targets

The last type of segment I discuss from the TSL perspective is what Jurgec (2011) calls an *icy target*. These segments act much like a blocker in that they prevent harmony beyond themselves, but differ from blockers in that they undergo the harmony that they block. An instance of iciness comes from the Macro-Jê language Karajá, which has a regressive ATR harmony process that spreads [+ATR] (Ribeiro, 2003; Rose & Walker, 2011; Walker, 2012). Underlyingly [−ATR] high vowels are the icy targets: an underlying [+high, +ATR] vowel will both trigger and propagate [+ATR], while an underlying [+high, −ATR] vowel will undergo harmony but then block its spread (Ribeiro, 2003). In other words, *derived* [+high, +ATR] vowels cause harmony to halt. Compare the examples in (29a-b), in which harmony spreads throughout the word, with those in (29c-d), which contain icy targets.

(29) Karajá ATR harmony (Ribeiro, 2003)

a.	/brɔrɛ-dĩ/	[brore-ni]	‘deer-similar.to’
b.	/bɛdɔ-dĩ/	[bedo-ni]	‘filhote-similar.to’
c.	/krɔbɪ-dĩ/	[krɔbi-ni]	‘monkey-similar.to’
d.	/kɑdʊ-dĩ/	[kɑdu-ni]	‘turtle-similar.to’
e.	/kɔlʊkɔ-dĩ/	[kɔluko-ni]	‘cajá (tree species)-similar.to’

My analysis abstracts away from the local nasal displacement that derives [-ni] from /-dĩ/; it is challenging to implement multiple processes with a single TSL function, but see Chapter 4 for extensions to the TSL class designed with this issue in mind. The pattern is sensitive to the [±ATR] value that each vowel has in the input string and so an ITSL_2 function is required. To keep the transducer in Figure 3.5 legible, I use only the following subset of the Karaja vowel inventory: /i/, /ɪ/, /u/, /ʊ/, /e/, /ɛ/, /o/, and /ɔ/. Limiting the inventory does not significantly affect the analysis. Because the process is regressive, the input string is read from right to left. The FST's tier includes underlyingly [+ATR] vowels alongside /ɪ/, /ʊ/. Crucially, /ɛ/ and /ɔ/ must be excluded from the tier so that reading them while in the [+ATR] state does not move us to the [−ATR] state. While it might feel odd to let non-tier elements be mapped unfaithfully, doing so does not violate the definition of an ITSL function in any way.[7] Compare this to the transitions originating in the [+ATR] state for the inputs /ɪ/ and /ʊ/ which *do* belong on the tier and thus *do* lead to the [−ATR] state. As a result, [+ATR] harmony can affect and pass over/through an underlyingly [−ATR] mid vowel, whereas it can affect but not pass over/through an underlyingly [−ATR] high vowel.

[7]The definition of an ITSL function requires only (i) that the tier be a subset of the input alphabet and (ii) that the corresponding transducer's current state represent the most recent tier elements. This definition places no requirements on the output edges of transitions, and indeed, the output alphabet could in principle be entirely disjoint from the input alphabet, making all transitions “unfaithful”.

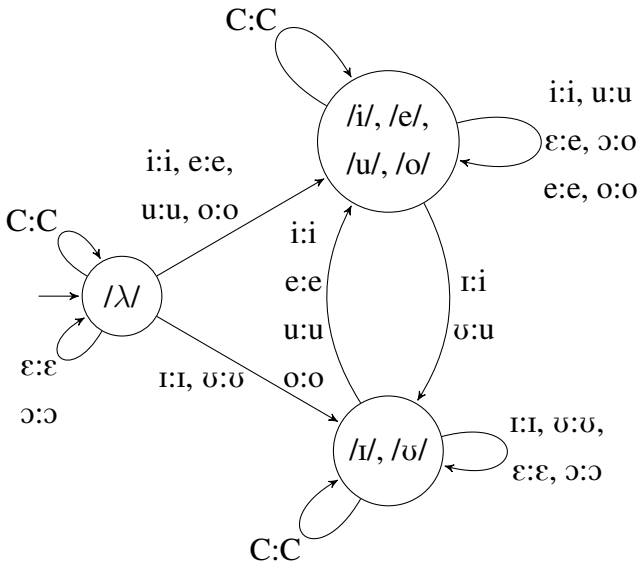


Figure 3.5: An $ITSL_2$ transducer that produces Karajá ATR harmony

Jurgec (2011, pp. 105-118) analyzes the behaviour of derived [œ] in Icelandic umlaut as another instance of iciness, citing data from Anderson (1972, 1974), Orešnik (1975, 1977), and Árnason (1992). The facts of Icelandic umlaut are as follows. Suffix /ʏ/ causes preceding /a/ to become [œ], as shown by the monosyllabic roots in (30). There is a separate process of vowel reduction that raises unstressed (i.e., non-initial) [œ] to [ʏ], and these derived instances of [ʏ] can propagate the umlaut further to the left, as shown by the polysyllabic roots in (31). One group of polysyllabic roots, however, exceptionally resist the process of raising, keeping the derived instances of unstressed [œ]. Unlike derived [ʏ], derived [œ] does not propagate harmony further to the left, as shown by the polysyllabic roots in (32).

(30) Icelandic umlaut, monosyllabic bases (Jurgec, 2011, p. 107)

a.	st[a]ð	'place.ACC.SG'	st[œ]ð[ʏ]m	'place.DAT.PL'
b.	b[a]nki	'bank.NOM.SG'	b[œ]nk[ʏ]m	'bank.DAT.PL'
c.	g[a]ta	'street.NOM.SG'	g[œ]t[ʏ]m	'street.NOM/ACC.PL'

(31) Icelandic umlaut, polysyllabic bases (Jurgec, 2011, p. 107)

a.	f[a]tn[a]ð	'suit.NOM.SG'	f[œ]tn[ʏ]ð[ʏ]m	'suit.DAT.PL'
b.	b[a]k[a]ri	'baker.NOM.SG'	b[œ]k[ʏ]r[ʏ]m	'baker.DAT.PL'
c.	b[a]n[a]ni	'banana.NOM.SG'	b[œ]n[ʏ]n[ʏ]m	'banana.DAT.PL'

(32) Icelandic umlaut, exceptional stems (Jurgec, 2011, p. 108)

a.	[a]tl[a]s	'atlas.NOM.SG'	[a]tl[œ]s[ʏ]m	'atlas.DAT.PL'
b.	sk[a]nd[a]li	'scandal.NOM.SG'	sk[a]nd[œ]l[ʏ]m	'scandal.DAT.PL'
c.	[a]lm[a]n[a]k	'calendar.NOM.SG'	[a]lm[a]n[œ]k[ʏ]m	'calendar.DAT.PL'

For now, I assume that that vowels are marked for stress in the input. Input marking of stress is necessary if we want to analyze the Icelandic pattern using an FST that tracks just a single tier, despite the fact that stress is fully predictable, always falling on the first syllable. It is possible for an FST to generate the pattern without underlying indication of stress, but such an FST must track multiple tiers simultaneously (see Section 4.3.5 for more details). I furthermore assume that the input alphabet treats exceptional instances of /a/ (i.e., ones that undergo umlaut but not raising) as separate from non-exceptional instance of /a/. The justification for this latter choice is that the examples presented in Jurgec (2011) seem mostly to be loanwords, and loanword behaviour is often analyzed using diacritics of some sort (e.g., indexed constraints in OT: see Itō & Mester, 1999). I use /A/ to represent an underlyingly stressed /a/ and use /@/ for the exceptional cases of /a/. I note that it is not necessary to make this distinction in the output alphabet, where there is only one type of [a]. Under these assumptions, the Icelandic pattern can be captured as either an ITSL_2 function or an OTSL_2 function, though I present only the ITSL_2 analysis since the analysis in Section 4.3.5 will require input sensitivity.

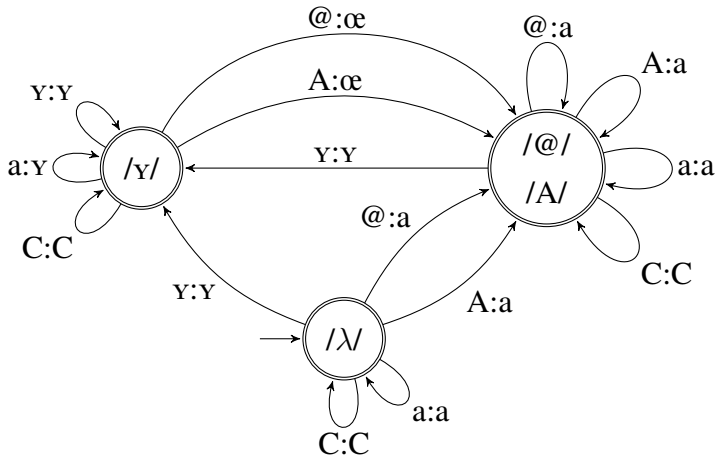


Figure 3.6: A right-to-left ITSL_2 transducer that computes Icelandic umlaut

The transducer in Figure 3.6, which reads from right to left, computes a simplified version of Icelandic umlaut where the only underlying vowels are /ʏ/ and the various types of /a/. Jurgec (2011) does not present any examples that tell us the behavior of other underlying vowels, but I assume that umlaut cannot skip intervening vowels, and so they would act as blockers. To cut down on visual clutter, I combine the /@/ and /A/ states into a single bubble in the figure, since the transitions out of both states are the same. On the tier are /ʏ/, /@/, and /A/, and the state that

interests us is the one labelled /ʏ/. While in the /ʏ/ state, reading an unstressed /a/ produces [ʏ] since umlaut and rounding can apply simultaneously. Since plain /a/ is not on the tier, applying umlaut and raising does not cause us to change state, so a chain of unstressed instances of /a/ will all undergo umlaut and raising. If we read /A/ or /@/ while in the /ʏ/ state, we produce [œ] since we apply umlaut but not raising. Doing so moves us out of the /ʏ/ state, meaning that subsequent instances of /a/ will not undergo umlaut or raising.

3.2.5 Iterativity and symmetry

A notable asymmetry between local and non-local processes is that the vast majority of attested long-distance processes are enforced iteratively (Kaplan, 2008; Hansson, 2010). Segments that are targeted and affected by a trigger of some process tend to become triggers themselves, propagating the process through the word until it reaches the word edge or encounters a blocker. Recall from Section 2.2.2 that whereas the OSL functions readily model iterative application, the ISL functions are necessarily non-iterative. Thinking that the ITSL functions would not be iterative and therefore not as useful, Burness & McMullin (2019) opted to define only the OTSL functions explicitly, leaving the definition of the ITSL functions implied. We have seen above, however, that there are patterns like tongue root harmony in Karajá that necessarily require sensitivity to input values and are therefore ITSL but not OTSL. Not only that, but iterative application *is* in fact possible in an ITSL function under specific circumstances. Specifically, iterative application can be ITSL when only one value of a binary feature is active (e.g., [+ATR] in Karajá, analyzed in the previous section), since leaving segments with the opposite value off the tier prevents them from halting the rule's progress through a word. Processes of this type are sometimes called *asymmetric* or *dominant-recessive* in the literature.

Asymmetric iterativity is describable using either an ITSL or OTSL function (in fact, the ITSL and OTSL transducers will look nearly identical), but the two function classes assign a slightly different interpretation to the trigger-target relations. Where an OTSL_2 function has each transformed target act as a new trigger, the ITSL_2 function has a single trigger targeting all subsequent relevant elements. The OTSL_2 interpretation is perhaps more typical in the literature, though an analysis along the lines of the ITSL_2 interpretation exists in Walker (2014), where it is argued that rounding harmony in Bayina Oroqen cannot be analysed as an instance of targets becoming triggers.

When the process is symmetric (i.e., all values of some feature are active) and feature changing (i.e. the process can replace pre-existing input values for some feature), iterativity cannot be modelled by an ITSL function but can be modelled with an OTSL function. One such process is the sibilant harmony in Samala, also known as Ineseño Chumash. In this language, the rightmost sibilant's output specification for the feature [±anterior] decides the output specification for

[±anterior] of all sibilants to the left (Applegate, 1972; Poser, 1982, 1993; McCarthy, 2007; Hansson, 2010; Heinz & Idsardi, 2010; McMullin, 2016). That both values are active can be seen by comparing the forms in (33), which show that a [−anterior] sibilant can affect a [+anterior] sibilant and vice versa. That the rule is feature-changing can also be seen by comparing (33), which show that the same morpheme (underlined) can trigger and undergo harmony. A right-to-left OTSL_2 transducer computing the rule is depicted in Figure 3.7 for reference, where '?' stands for any segment that is not a sibilant.

(33) Samala sibilant harmony (Applegate, 1972)

a.	/s-api-tʃʰo-us/	[sapitsʰolus]	'he has a stroke of good luck'
b.	/s-api-tʃʰo-us-waʃ/	[ʃapitʃʰoluʃ-waʃ]	'he had a stroke of good luck'

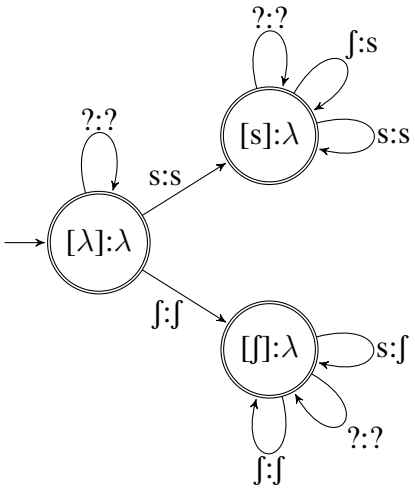


Figure 3.7: A right-to-left OTSL_2 transducer that produces Samala sibilant harmony

On the opposite end of the spectrum, there do exist unambiguously non-iterative long-distance patterns. For example, underlyingly round vowels in Kazakh spread their rounding only to the immediately following vowel (Balakaev, 1962; McCollum, 2018; McCollum & Kavitskaya, 2018) as shown by the data in (34). This pattern is easy enough to account for with an ITSL_2 function if the tier includes all vowels. When all vowels are on the tier, a round vowel can only cause the very next vowel to become round, since reading a vowel causes a change of state. An OTSL function cannot properly model this non-iterativity since the target becomes round and would therefore be on the tier and itself count as a trigger.

(34) Kazakh non-iterative rounding harmony (McCollum & Kavitskaya, 2018)

a.	/mojən-də/	[mojʊn-də]	'neck-ACC'
b.	/tʊr-məs-ə-nəŋ/	[tʊr-mʊs-ə-nəŋ]	'live-NMZR-POSS.3-GEN'
c.	/kino-m-əz-dəŋ/	[kino-m-ʊz-dəŋ]	'movie-POSS.1-PL-GEN'

A more intriguing non-iterative pattern comes from Harari, where the 2nd person singular feminine subject suffix /-i/ causes the rightmost coronal obstruent to become palatalized (Rose, 2004).[8] Other instances of suffixal /i/ do not cause palatalization and so I will assume the trigger is a special indexed symbol /i$_x$/, similar to what I did for Icelandic in Section 3.2.4. The palatalization targets the rightmost coronal obstruent as shown by the data in (35a-b). Coronal sonorants (except /r/) are optionally palatalized if they come after the obstruent target as shown in (35c) but never palatalize if they come before the obstruent target as shown in (35d). For the purposes of this section, I will assume that sonorant palatalization is obligatory where it would optionally be allowed.

(35) Harari long-distance palatalization (Rose, 2004)

a.	/sidab-i$_x$/	[sidʒab-i]	'insult-2SG.F'
b.	/sibar-i$_x$/	[ʃibar-i]	'break-2SG.F'
c.	/xidan-i$_x$/	[xi-dʒaɲ-i]	'cover-2SG.F'
d.	/dinabtʃ'-i$_x$/	[dinabtʃ'-i]	'be.frightened-2SG.F'

As I will present shortly, the facts above are easy enough to account for using an ITSL function, though in the interest of full disclosure, the pattern does have some complications that cannot be handled within a TSL function. First, if the only coronal consonant is a sonorant, that sonorant will obligatorily palatalize unless it is root initial (Rose, 2004). Second, root-initial obstruents can optionally resist palatalization if a following sonorant has been palatalized (Rose, 2004). These issues set aside, the Harari pattern can be modelled by the right-to-left ITSL$_2$ transducer in Figure 3.8, where 'T' represents a coronal obstruent, 'N' represents a coronal sonorant, 'P' represents a palatal obstruent, 'J' represents a palatal sonorant, and '?' represents all other segments (aside from the triggering suffix vowel). I assume that the tier consists of the triggering suffix vowel and coronal obstruents, excluding palatal consonants and coronal sonorants.

The state that interests us in this transducer is the one labelled /i$_x$/. Out of this state, all segments surface faithfully except for coronal consonants. If we read a coronal sonorant while in this state, we palatalize it and loop back since coronal nasals are not on the tier. If we read a coronal obstruent while in this sate, we palatalize it and move to the /T/ state since coronal obstruents are on the tier. The palatalization will thus affect all coronal sonorants, up until a coronal obstruent is encountered, which becomes palatalized but prevents further palatalization.

3.2.6 Double, (non-)initial, and (non-)final triggers

All of the long-distance patterns discussed so far share one striking thing in common: they are are TSL$_k$ for $k = 2$. This is not necessarily a surprising fact when we consider the typical interpretation

[8] Though the process is optionally iterative, especially in women's speech (Rose, 2004).

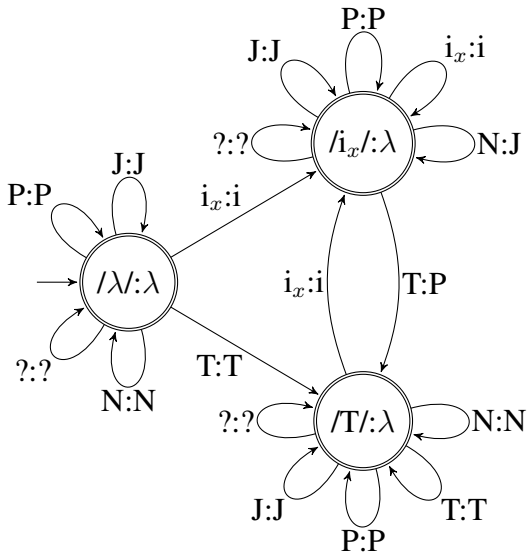


Figure 3.8: A right-to-left ITSL_2 transducer that produces Harari palatalization

of long-distance processes. Long-distance harmony is usually construed as the feature value of one segment spreading in some direction to affect other eligible segments, so it makes sense that a window of length 2 is sufficient in the tier-based context, since it lets us remember the most recent "relevant" element. Long-distance dissimilation can also generally be reduced to the influence of the most recent relevant element. The prominence of $k = 2$ is also interesting with regards to learnability. Burness & McMullin (2019) showed that while any OTSL_k function can be learned efficiently from positive data when the tier is known in advance, only the OTSL_2 functions can be efficiently learned by their algorithm from positive data when the tier is *not* known in advance.[9] Nonetheless, there are several behaviours that require a value of $k > 2$ for various reasons.

A first example is the rounding harmony in one variety of Oroqen. Some sources analyze the harmony as only being triggered by a sequence of two rounded vowels (Zhang et al., 1989; Li, 1996; Zhang, 1996; Zhang & Dresher, 1996; Walker, 2001, 2014). The data in (36) show the definite article suffix becoming round when preceded by two round vowels, but not when preceded by a single round vowel (even if this vowel is long). Note that the language also contains a process of ATR harmony, hence the alternation between [ɔ] and [o]. An alternative analysis of the same data comes from Dresher & Nevins (2017), who analyze the harmony as triggered specifically by non-initial vowels. They analyze the process in this way because the Russian loanword [kinɔ] 'film' causes rounding harmony in the definite object suffix /-wa/ giving us [kinɔ-wɔ] instead of *[kinɔ-wa] as predicted by the double-trigger analysis (Dresher & Nevins, 2017). There also exists a plural suffix /-nOr/ for kinship terms that is always round, and this suffix causes rounding harmony

[9]This of course does not necessarily mean that learning tiers is impossible when $k > 2$, only that no method currently exists for doing so efficiently. More will be said on this topic in Chapter 7.

regardless of the rounding in preceding vowels, giving us [ətʃəxə-nɔr-wɔ-t] 'paternal.uncle-PL-DEF-ACC' instead of *[ətʃəxə-nɔr-wa-t], for example (Dresher & Nevins, 2017). This again is expected under the non-initial trigger analysis, but not under the double trigger analysis.

(36) Oroqen rounding harmony (Walker, 2001)

a.	/ɔlɔ-wa/	[ɔlɔ-wɔ]	'fish-DEF.OBJ'
b.	/tʃoŋko-wa/	[tʃoŋko-wo]	'window-DEF.OBJ'
c.	/mɔː-wa/	[mɔː-wa]	'tree-DEF.OBJ'

Interestingly, both the double trigger analysis and non-initial trigger analysis can be implemented as an OTSL function if we set k to be greater than or equal to 3. The states in a TSL_k transducer are labelled with sequences of *up to* length $k-1$. Accordingly, until enough tier elements have been encountered, the transducer can be in a state labelled with fewer than $k-1$ segments,[10] but it will cycle through states labelled with $k-1$ elements once that number is reached. For example, an OTSL_3 transducer with the alphabet $\{\text{V, C}\}$ and the tier $\{\text{V}\}$ will enter the state 'V' upon producing its first vowel, and will not move to the state 'VV' until a second vowel is produced. After that point it will remain in the state 'VV', never returning to the state 'V'. This is illustrated in Figure 3.9. When implementing the double-trigger analysis as an OTSL_3 function, we would ensure that rounding harmony is only enforced out of a state whose label consists of two round vowels. Similarly, when implementing the non-initial trigger analysis as an OTSL_3 function we would ensure that rounding harmony is only enforced when the state label contains two vowels and the rightmost of these is round.

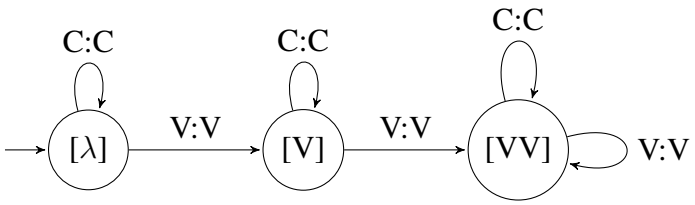


Figure 3.9: Example of state labels shorter than the maximum

Similar to how the right-hand vowel of a two-vowel state label in a progressive TSL_3 function is necessarily non-initial, a one-vowel state would necessarily reflect the first vowel when all vowels are on the tier. With $k = 3$, then, we can also model non-iterative harmony that is triggered specifically by the first or last vowel in a word. In Megisti Greek, regressive non-iterative backness harmony is triggered only by the last vowel in a word (van Oostendorp & Revithiadou, 2005; McCollum & Kavitskaya, 2018). Harmony preceding a final back vowel is shown in (37a-b), and lack of harmony before a non-final back vowel is shown in (37c).

[10]The initial state in all TSL_2 transducers above fits this description since its label has a length of 0.

(37) Megisti Greek non-iterative backness harmony
Triggered only by final vowels (van Oostendorp & Revithiadou, 2005)

a. /sits-a/ [sutsa] 'fig.tree-NOM.F'
b. /filak-s-e/ [filekse] 'guard-3SG.PST'
c. /anofli/ [anefli] 'lintel'

Trigger status is purely positional in this pattern and has nothing to do with stress placement (McCollum & Kavitskaya, 2018), so we cannot rely on a dichotomy between stressed and unstressed vowels to derive the restriction on triggers. In this pattern (or its mirror image), we need a tier of all vowels and $k = 3$. A state labelled with a single vowel will then correspond to having seen only the first vowel in a left-to-right machine and correspond to having only seen the last vowel in a right-to-left machine. Restricting harmonic transitions so that they can only come from these states will limit triggers to the desired word edge.

The above goes to show that increasing k can account for certain behaviours not captured by the TSL_2 region. Increasing k does not directly capture the causes of the behaviours, however. Consider again the double trigger interpretation of Oroqen data, where rounding harmony is triggered only by a neighbouring pair of round vowels. I captured this by restricting harmony so that it applies only when leaving a state labelled with two rounded vowels, but there is nothing stopping us from saying that rounding harmony is only enforced when leaving a state labelled with some other arbitrary combination of vowels. This is not necessarily an issue, and a similar problem faces any approach that uses FSTs since transitions have no direct relationship to the states that they leave. That being said, it might be interesting to look for means of maintaining $k = 2$ when modelling patterns like those in Oroqen and Megisti Greek, although I leave the development of such methods for future research.

Indeed, Eastern Meadow Mari displays a pattern of positional triggering that cannot be handled by increasing k. All suffix vowels in the language harmonize with the initial vowel (Vaysman, 2009; Walker, 2011). The words in (38) show alternations in the nominative singular second person plural possessive suffix, and the words in (39) show alternations in the dative suffix. Walker analyzes the data as an interaction between the suffix vowel and the first vowel because of the forms (38e) and (39c) with disharmonic bases. In the first we have [u] triggering a back allophone across front [e], and in the second we have [e] triggering a front allomorph across back [a].

(38) Eastern Meadow Mari backness harmony, nominative singular 2nd person plural possessive suffix (Walker, 2011, p. 148)

a. em-dæ ‘your (pl) medicine’
b. tʃødræ-tæ ‘your(pl) forest’
c. tʃijæ-tæ ‘your (pl) paint’
d. tyrə-tæ ‘your (pl) edge’
e. uβer-ta ‘your (pl) news’
f. kutko-ta ‘your (pl) ant’
g. olək-ta ‘your (pl) meadow’
h. rəwəʒ-ta ‘your (pl) fox’

(39) Eastern Meadow Mari backness harmony, dative suffix (Walker, 2011, pp. 148-149)
a. imɲə-læn ‘horse (dat)’
b. kyzə-læn ‘knife (dat)’
c. meraŋ-læn ‘hare (dat)’
d. keŋeʒ-læn ‘summer (dat)’
e. olma-lan ‘apple (dat)’
f. munə-lan ‘egg (dat)’
g. təlzə-lan ‘moon (dat)’

The privileged status of initial vowels in Eastern Meadow Mari does not seem to derive from secondary factors such as stress, which falls on the *rightmost* full vowel (Vaysman, 2009). From the TSL perspective, we do not expect suffix vowels to harmonize specifically with the first vowel across all other vowels without a principled means of singling out just the initial vowels as belonging to the tier or a principled means of rendering non-initial vowels invisible. In harmony of this sort, the source and target of harmony would be *maximally* distant from each other as opposed to *minimally* distant, going counter to the definition of the TSL functions. Patterns such as this one, wherein the privileged status and special behaviour of initial vowels cannot be derived from other factors, pose a strong challenge for tier-based functions. We could, of course, give all initial vowels a special diacritic to distinguish them from non-initial vowels, but a more satisfying potential solution would be to appeal to structure-sensitive means of tier projection like those in Graf & Mayer (2018) and De Santo & Graf (2019), which allow the identity of surrounding material to play a part. This could provide a means of narrowing the set of relevant vowels in a word to just the vowel in the initial syllable, and this vowel would trivially be the most local relevant vowel by virtue of its uniqueness. I leave the confirmation of this conjecture for future research.

3.3 Extending TSL functions

As is hopefully clear from the above sections, the TSL functions are quite versatile, able to model a wide range of segmental behaviours in long-distance processes such as blocking, as well as other properties of long-distance systems such as (non-)iterativity. The TSL functions are, however, not sufficient to capture *all* long-distance phonology. This section briefly presents three specific shortcomings that will be addressed in the next two chapters. In order, these are the inability to model multiple simultaneous processes, the inability to model bidirectional application, and the inability to model processes that have two-sided contexts.

3.3.1 Multiple processes

Consider the Tamashek dialect of Tuareg, which contains a process of long-distance regressive sibilant harmony and a process of long-distance regressive labial dissimilation (Heath, 2005; McMullin, 2016). The sibilant harmony can be seen in words where the causative prefix /s-/ is followed non-locally by another sibilant, whereupon it takes that other sibilant's values for anteriority, voicing, and pharyngealization as shown by the data in (40) from Heath (2005, p. 442). Note that the language has considerable vowel allophony, and I write 'V' where Heath (2005) does not provide the surface vowel quality. The labial dissimilation can be seen in words where a prefix /m/ (such as in the mediopassive) is followed non-locally by a labial consonant other than /w/, whereupon the prefix /m/ will dissimilate to [n]. Data for this process is shown in (41) from Heath (2005, p. 472). That both processes can occur simultaneously is demonstrated by the word in (42) from Heath (2005, p. 462), which contains the causative and mediopassive prefixes together.

(40) Sibilant harmony: causative /s-/

-s-VŋŋV-	'cook'
-s-VsVfVr-	'treat (patient)'
-sˤ-VsˤuhV-	'strengthen'
-ʃ-VluʃV-	'clean sand from'
-z-VjVzzV	'scrutinize'

(41) Labial dissimilation: mediopassive /m-/

-m-VrtVj-	'become mixed'
-n-VkmVm-	'be squeezed'

(42) Both prefixes/processes

ɑ-zˤ-ənː-ət-əlməzˤ	'spitting saliva'

The combination of sibilant harmony and labial dissimilation cannot, however, be computed by a single TSL function. To see why, consider what happens when we try with an OTSL_2 function

whose tier consists of all sibilants and labial consonants. Producing a sibilant will push the most recent labial consonant (if any) out of the $k-1$ window, and producing a labial consonant will push the most recent sibilant (if any) out of the $k-1$ window. At any given point, then, we can only know how to correctly map an input /s/ or only know how to correctly map an input /m/. Increasing k does not eliminate the issue, since any number of sibilants can in principle occur between two labial consonants, and any number of labial consonants can in principle occur between two sibilants.

A further difficulty for TSL functions posed by Tamashek Tuareg is that long-distance regressive labial dissimilation interacts with a local process of regressive nasal place assimilation. The complication arises from the fact that the local process overrides the long-distance process: /m/ fails to dissimilate specifically when it is immediately followed by an oral labial stop, in order to avoid a heterorganic cluster. This can be seen in a word like (43) from Heath (2005, p. 476) where the reciprocal prefix /Vm-/ comes immediately before a /b/. We are faced with a paradox if we attempt to model this interaction as a single TSL function. In order to know when an input labial needs to dissimilate we need to ignore everything that is not a labial consonant, but in order to know when an input labial needs to obey place assimilation we cannot ignore anything.

(43) Local blocking of dissimilation, reciprocal /Vm-/
-æm-bæbbɑ- 'carried each other'

Whether such interactions between multiple local and/or long-distance processes are problematic depends on our conception of a language's phonological system: is it a series of input-output maps or is it a single "master" input-output map? The latter position is explored by Chandlee & Heinz (2018) and Chandlee et al. (2018) in the context of formal language theory and automata theory. In an appendix to their article, Chandlee et al. (2018) prove that the ISL functions are not closed under composition (i.e., given two ISL functions $f(w)$ and $g(w)$, the single function equivalent to $f(g(w))$ is not guaranteed to be ISL),[11] and yet the main body of their article shows that certain opaque rule interactions such as counterfeeding and counterbleeding can nevertheless be modelled using a single ISL function. The next chapter investigates how multiple TSL functions can and should be combined, taking inspiration from Chandlee & Heinz's (2018) and Chandlee et al.'s (2018) approach, as well as work by Aksënova & Deskmukh (2018) and McMullin et al. (2019) on the combining of multiple TSL languages.

3.3.2 Bidirectional application

One aspect of long-distance patterns is tacitly assumed above, namely that they tend to apply in a single direction. This seems to be the norm, but there are patterns that apply in both directions,

[11] Although Chandlee & Lindell (under review) conjecture that closure is guaranteed when neither of the functions being combined contains a "null cycle" where infinite deletion is possible.

typically from the root outwards. Take for instance vowel harmony in Degema, where prefixes *and* suffixes take on the [±ATR] specification of the root (Archangeli & Pulleyblank, 2007; Kari, 2007). The data in (44) show both parts of a circumfix alternating to match the root's value for [±ATR]. The combination of regressive harmony to prefixes and progressive harmony to suffixes gives the impression of a harmony system that begins from the centre. TSL functions obligatorily start at either the left or right edge of the input, so even if we make a distinction between root and affix vowels by having the latter be unspecified for [±ATR] in the input, we will only correctly treat either the prefixes or the suffixes.

(44) Degema ATR harmony (Kari, 2007)

a.	[ɓól]	'hold'	[u-ɓóˈl-ə́m]	'holding'
b.	[gɛ́n]	'look'	[ʊ-gɛ́ˈn-ám]	'looking'
c.	[ɗúm]	'create'	[o-ɗúˈm-ə́m]	'creator'
d.	[hɔ́r]	'sharpen'	[ɔ-hɔ́ˈr-ám]	'sharpener'

A deceptively simple solution presents itself: why not apply the same TSL function once going from left to right, then a second time going from right to left? The first pass will fill in the missing [±ATR] value on suffixes and the second pass will fill in the missing [±ATR] values for prefixes. A similar idea of decomposing a single bidirectional process into an ordered pair of opposite-direction subsequential functions is how Heinz & Lai (2013) account for stem-controlled vowel harmony, though they caution against freely composing subsequential functions due to a theorem from Elgot & Mezei (1965). The theorem states that running a first subsequential transducer in one direction and then a second subsequential transducer in the reverse direction can describe any regular function. This happens because the first transducer can "annotate" the input string to give the opposite-direction transducer information about distant upcoming material. Heinz & Lai (2013) accordingly suggest that when we need to split a single process into two parts, the first transducer's output alphabet must be equal to its input alphabet (which is then the input alphabet to the second transducer), and must not be allowed to produce an output longer than the input. These restrictions prevent some of the annotation necessary for producing truly regular functions, and functions describable in two steps without intermediate markup are known as the *weakly deterministic* functions. As an extension of this line of work, Chapter 5 will show that the majority of attested bidirectional patterns fit into an even more restrictive class than the Weakly Deterministic functions. The *Tier-based Weakly Local* functions proposed there decompose a single process into two TSL processes that apply in opposite-directions while ensuring that the first process does not perform any unnecessary annotation.

3.3.3 Two-sided contexts

Earlier in Section 2.2.1, I mentioned that ISL functions are able to model two-sided contexts since they can wait up to a finite number of steps before transforming an input element. This ability is not available to ITSL functions, however, since non-tier elements are not remembered in any way by the function (e.g., they are not recorded in the sate labels of an ITSL function's transducer). A challenge for the TSL hypothesis, then, is that there do exist patterns that can be interpreted as being affected by unbounded information in one direction as well as bounded information in the other. Consider the lowering harmony in c'Lela, where suffix high vowels become mid if they are preceded by a non-high vowel in the root (Dettweiler, 2000; Pulleyblank, 2002; Archangeli & Pulleyblank, 2007; Michel, 2009; Jurgec, 2011). When there is more than one vowel following the triggering root vowel, the harmony only affects the very last one, which will always be in word-final position thanks to the syllabic structure of c'Lela (Jurgec, 2011, p. 186). The data in (45) show that the class membership suffix /-i/ will lower to [e] when it is the only suffix following a non-high root vowel, but will be unaffected when it is followed by another suffix vowel.[12]

(45) C'Lela lowering harmony (Dettweiler, 2000)

a.	/zis-i/	[zis-i]	'long-I.CLASS'
b.	/rek-i/	[rek-e]	'small-I.CLASS'
c.	/zis-i-ni/	[zis-i-ni]	'long-I.CLASS-ADJM'
d.	/rek-i-ni/	[rek-i-ne]	'small-I.CLASS-ADJM'

The basic facts of lowering are simple enough to account for with an ITSL_2 or OTSL_2 function: the tier consists of non-high vowels, and input high vowels will be output as non-high while in a non-high state. Accounting for the fact that targets are only lowered in word-final position, on the other hand, lies outside the powers of the TSL class. In order to confirm that an input vowel is in word-final position, we need to postpone its output and attempt to read the next input element. If there is no next input element, we know that vowel is word final, but if there is a next input element, we know that vowel is not word final. Once we ascertain the final or non-final status of the vowel, we can transform it accordingly, along with any other material that needs to be output. A paradox ensues: for the postponement strategy to work we need to track a tier containing everything (otherwise we will forget the identity of the element we just postponed), but to properly apply lowering harmony we need to track a tier containing only non-high vowels. Interestingly, the method proposed in the next chapter for modelling multiple processes simultaneously can also resolve the paradox imposed by the c'Lela pattern and others like it that are sensitive to unbounded information in one direction but bounded information in the other.

[12] Dettweiler (2000) was unable to determine the exact meaning of of the suffix /-ni/, and so treats it as an adjectival marker of uncertain status.

Related to hybrid bounded/unbounded cases like c'Lela lowering harmony is what Jardine (2016) calls *unbounded circumambience*, where the application of a process depends on potentially non-local information in both directions simultaneously. Take for instance Tutrugbu vowel harmony, as analyzed by McCollum & Essegbey (2018) and McCollum et al. (2020). Dominant [+ATR] root vowels spread their value leftwards to prefix vowels, and when all prefix vowels are of the same height, the harmony will reach the left word edge (McCollum & Essegbey, 2018; McCollum et al., 2020), as seen in (46a) for a string of [+high] prefixes and (46b) for a string of [−high] prefixes. Roots are underlined for clarity. Root vowels that are [+ATR] spread this value leftwards to prefix vowels, and when all prefix vowels are of the same height, the harmony will reach the left word edge (McCollum & Essegbey, 2018; McCollum et al., 2020), as seen in (67a) for a string of [+high] prefixes and (67b) for a string of [−high] prefixes. Roots are underlined for clarity. The behaviour of strings of prefixes with different heights depends on the height of the leftmost vowel. Harmony spreads to all prefix vowels when the leftmost vowel is [−high] as seen in (67c), but harmony is blocked by [−high] vowels when the leftmost vowel is [+high] as seen in (67d), although harmony still spreads up until that [−high] blocker (McCollum & Essegbey, 2018; McCollum et al., 2020). The outcome of an input [−high] vowel can therefore depend simultaneously on non-local information to its left (the height of the leftmost vowel) and non-local information to its right (the ATR specification of the root), creating clear instances of unbounded circumambience.

(46) Conditional blocking by low vowels in Tutrugbu ATR harmony (McCollum et al., 2020)

a.	/bʊ-tí-ʃē/	[bu-tɪ́-ʃē]	'1PL-NEG-grow'
b.	/ka-ba-ʃē/	[ke-be-ʃē]	'2SG-FUT-grow'
c.	/ka-tí-ba-ʃē/	[ke-tɪ́-be-ʃē]	'CL7-NEG-FUT-grow'
d.	/ɪ-ba-dɪ-wu/	[ɪ-ba-di-wu]	'1SG-FUT-ITV-climb'

Jardine (2016) hypothesized that unbounded circumambient processes were not weakly deterministic, although O'Hara & Smith (2019), Smith & O'Hara (2019), and Lamont et al. (2019) show that loopholes in the definition can be exploited to model several unbounded circumambient patterns. I conjecture that cases of unbounded circumambience like the one in Tutrugbu do not need the full power afforded by the weakly deterministic functions, but could (like I will propose for bi-directional harmony) be modeled as a pair of opposite-direction single-tiered or multi-tiered functions. This possibility is pursued in Chapter 5.

Finally, I would like to point out that less obvious instances of unbounded circumambience can arise from strongly local implementations of spreading (see Section 3.2.1 for the initial discussion of these approaches). Under the assumption of strongly local spreading in a system of backness vowel harmony, for example, transparent consonants would be co-articulatorily front between two

front vowels and co-articulatorily back between two back vowels, which at a first glance seems to require just an expanded output alphabet with co-articulated versions of each consonant. Additional output symbols may not be enough on their own, though, depending on what happens to consonants between two vowels with conflicting backness values. Do consonants in this environment take on the backness of the preceding vowel, the following vowel, some combination of both, or neither?. The last two possibilities would require us to know the identity of both flanking vowels, and these could in principle be an arbitrary distance away (e.g., we could have an arbitrarily long string of consonants between two vowels). This goes to show that while harmony processes may be TSL when our attention is limited to categorical changes, they can become more complex when we consider their phonetic implementation.

3.4 Chapter summary

The chapter started with an overview of how tiers have been used in phonological theory to model long-distance processes. I then showed how previous computational work incorporated tiers into the strictly local languages and functions to yield the Tier-based Strictly Local languages and functions. Following that, I demonstrated how the TSL functions can account for observed properties of attested long-distance processes including segmental transparency, parasitic harmony, segmental blocking, icy targets, and iterativity. Finally, I discussed some behaviours that lie outside the capabilites of the TSL functions and which are the focus of the next two chapters. Chapter 4 will consider how giving functions access to multiple tiers permits the modelling of interactions between several simultaneous processes and the modelling of processes that are sensitive to unbounded information in one direction while also being sensitive to bounded information in the other. Chapter 5 will consider how to achieve bidirectional application and some instances of unbounded circumambience while maintaining the restrictiveness of tier-based strict locality.

Chapter 4

Functions with multiple tiers

The previous chapter showed how the simple inclusion of a tier into a Strictly Local (SL) function greatly extends our empirical coverage. Tiers naturally reflect the idea that segments can be irrelevant to a process, and that the non-local triggering of a process becomes local if we ignore irrelevant segments. Various properties of attested long-distance processes including transparency, parasitism, blocking, and iciness were shown to result from the existence and content of a tier, without the need for additional theoretical mechanisms. The Tier-based Strictly Local (TSL) functions are thus a natural and worthwhile extension of the SL functions, but they nonetheless have their own drawbacks. As they have so far been defined, the TSL functions are equipped with just a single tier, making them incapable of modelling simultaneous phonological processes that would require different tiers, as well as individual processes that require multiple pieces of information to be applied correctly.

The current chapter addresses this shortcoming by defining an extension of the TSL functions that allows for multiple tiers. These functions represent a big jump in computational power, however, and so the chapter also discusses how to avoid overgeneration. The particular restriction on multi-tiered functions that I propose comes in two parts. First, each input element is associated with a set of tiers that on their own can fully determine what the element is mapped to. Second, this *target-specified* set of tiers must form a strict superset-subset hierarchy. In this way, we can track multiple, related sources of information when deciding how to process a particular input element. After formalizing the restricted subclasses of multi-tiered functions, I show how they permit simple and intuitive analyses of otherwise challenging phonological phenomena. Finally, I compare my proposal to the subsequential functions, showing that this other model generates pathological patterns that are unattested and/or incredibly difficult to learn. Insofar as the multi-tiered functions can generate the same attested behaviours as the subsequential functions while not predicting these problematic patterns, they provide a more appropriate model of long-distance phonology.

4.1 Formalizing and restricting multi-tiered functions

The main method that I will use to tackle the above issues is the inclusion of multiple tiers in a single function. It turns out that it is pretty easy to define functions that operate relative to local contexts on multiple tiers. I will call these the Multi-tiered Strictly k-Local (MTSL$_k$) functions, for lack of a better name. Note that I use a large subscripted conjunction symbol to collapse a series of formulae that are identical aside from using different tiers.

Definition 4.1 Input Multi-tiered Strictly k-Local function**.**
A function $f : \Sigma^* \rightarrow \Delta^*$ is IMTSL$_k$ if there is a finite set Θ of tiers $T \subseteq \Sigma$ such that for all $w_1, w_2 \in \Sigma^*$:

$$[\bigwedge_{T \in \Theta} [\mathtt{suff}_T^{k-1}(w_1) = \mathtt{suff}_T^{k-1}(w_2)]] \implies [\mathtt{tails}_f(w_1) = \mathtt{tails}_f(w_2)]$$

Definition 4.2 Output Multi-tiered Strictly k-Local function**.**
A function $f : \Sigma^* \rightarrow \Delta^*$ is OMTSL$_k$ if there is a finite set Θ of tiers $T \subseteq \Delta$ such that for all $w_1, w_2 \in \Sigma^*$:

$$[\bigwedge_{T \in \Theta} [\mathtt{suff}_T^{k-1}(f^p(w_1)) = \mathtt{suff}_T^{k-1}(f^p(w_2))]] \implies [\mathtt{tails}_f(w_1) = \mathtt{tails}_f(w_2)]$$

In words, there is a finite set of tiers such that if two input strings share a $k-1$ suffix *on all tiers*, then they will have the same tails.[1] Notice how this is a direct extension of the TSL$_k$ functions, since the singleton tier of a TSL$_k$ function will satisfy the above definition. Like the OSL and TSL functions, reading direction divides what I am calling the MTSL functions into two overlapping but distinct classes. I will continue to speak of the MTSL functions (and their variations to be explored below) as a single class, though I will specify when a given function requires right-to-left directionality (i.e., directionality is henceforth assumed to be progressive unless stated otherwise).

As a simple example, consider the Kikongo language, whose pattern of nasal consonant harmony exists parallel to a process of height harmony among vowels (Ao, 1991; Odden, 1994; Hyman, 1998; Rose & Walker, 2004; Hansson, 2010; Aksënova & Deskmukh, 2018). Nasal harmony will cause /l/ and /d/ to become [n] when preceded by a nasal consonant at any distance. For its part, the height harmony will cause an underlying high vowel to become mid if preceded by a non-high vowel at any distance. Data showing the effect of both processes on the perfective active and perfective passive suffixes are provided in (47). We can capture both processes simultaneously using an MTSL$_2$ function with at least two tiers: T_1 containing all and only non-high vowels and T_2 containing all and only the nasal consonants. Two input strings w_1 and w_2 will then have the

[1] One could also allow each tier to specify its own value for k. I do not pursue this option here, though, since in the case that some tiers need a higher value of k than others, one can always just use the highest needed value on all tiers.

same tails if they have the same 1-length suffix on T_1 (i.e., if they both contain or both do not contain a non-high vowel) *and* the same 1-length suffix on T_2 (i.e., if they both contain or both do not contain a nasal consonant).

(47) Kikongo nasal and height harmony (Aksënova & Deskmukh, 2018)

a.	/-suk-idi-/	[-suk-idi-]	'wash-PERF.ACT'
b.	/-nik-idi-/	[-nik-ini-]	'ground-PERF.ACT'
c.	/-meng-idi/	[-meng-ene-]	'hate-PERF.ACT'
d.	/-suk-ulu/	[-suk-ulu-]	'wash-PERF.PASS'
e.	/-nik-ulu-/	[-nik-unu-]	'ground-PERF.PASS'
f.	/-meng-ulu/	[-meng-ono-]	'hate-PERF.PASS'

Of course, allowing any conceivable number of tiers and allowing any conceivable relationship between their contents is too powerful. One pathological behaviour that can arise from excessively free tier sets would be 'gang-up' effects. For example, given a collection of disjoint tiers that each contain a single element, we can describe processes where the output of some input element depends on the exact set of preceding elements regardless of their order. A similar behaviour can arise from overlapping but disjoint tiers. Given $T_1 = \{\text{s}, \text{ʃ}, \text{t}\}$ and $T_2 = \{\text{s}, \text{ʃ}, \text{n}\}$ we could describe a process of sibilant harmony that is blocked only when *both* 't' and 'n' intervene, regardless of their order. The same tiers could also allow an even more bizarre pattern where intervening 't' blocks sibilant harmony only if 'n' does not also intervene (and/or vice versa). To the best of my knowledge, such processes are unattested and should be excluded. How best, then, to limit tier sets and thus prevent overgeneration? I will propose that each input element or *target* separately specifies its own dedicated set of tiers. Each target pays attention only to the tiers belonging to the set that it specifies, and no others. Additionally, tiers within the same one of these *target-specified* sets must fall into a superset-subset relationship, but tiers from different target-specified sets will be free to relate in any way.

This proposal is loosely inspired by Nevins' (2010) analysis of vowel harmony as a search procedure initiated by the harmony target, rather than a spreading process initiated by the harmony source. In Nevins' model of vowel harmony, an underspecified vowel will look in a predetermined direction for the nearest element from which it can copy a value for its missing feature. Which elements count as relevant donors will depend on the identity of the needy target, and can vary according to parameters such as whether they are contrastive for the sought feature. To illustrate, consider how this *Search and Copy* procedure would handle Hungarian backness vowel harmony (see also Section 3.2.1). Suffix vowels would be unspecified for the feature [±back] and would search to their left for the nearest vowel that is contrastive for the feature [±back], from which they will copy the value. The vowels [i] and [e] have no [+back] counterparts in Hungarian (unlike [y]

and [ø] which have the counterparts [u] and [o] respectively) making them redundantly [−back] and therefore transparent to harmony.

Andersson et al. (2020) show that a Search and Copy procedure will generate an Input Tier-based Strictly Local (ITSL) maps when a single feature is being sought in a single direction. An ITSL transducer looks effectively like the opposite of its corresponding Search and Copy rule, acting instead as some kind of *Remember for Copying* procedure. The transducer's tier contains only relevant donor segments and rather than starting from the target's position and seeking a donor in one direction, we start from the word edge and read in the opposite direction. The transducer preemptively remembers the most recently encountered potential donor (using the current state's label), allowing us to transform the target appropriately upon reading it.

Below, I will show that a subset of the MTSL functions seem sufficient for phonological analysis, specifically those functions that can be interpreted as accomplishing multiple Remember for Copying procedures (or something along the lines of *Remember for Opposing* in the case of dissimilation). To explicitly delineate this subclass of the MTSL functions, it is first necessary to narrow our attention from functional tails (which are infinite sets) to just the contribution of single input elements. Informally, the contribution of σ relative to w is the portion of $f^p(w\sigma)$ uniquely and directly attributable to σ. In other words, it is what we would append to the output upon reading σ in a transducer after having read w (see the paragraph below for a concrete illustration). It will sometimes be necessary to consider the material that would correspond to a word-end symbol (i.e., the material supplied by the final string associated to a state in a transducer). I accordingly assume that all input strings end in the special symbol $\ltimes$ that is not part of the input alphabet Σ. Contributions are formally defined as follows, where the notation $x^{-1} \cdot y$ represents the string y with x subtracted from its front (e.g., $a^{-1} \cdot aba = ba$ and $\lambda^{-1} \cdot bcd = bcd$).

Definition 4.3 Contribution**.**
Given a function $f : \Sigma^* \to \Delta^*$ and some $w \in \Sigma^*$:

- *For $\sigma \in \Sigma$: $\texttt{cont}_f(\sigma, w) = \texttt{lcp}(f(w\Sigma^*))^{-1} \cdot \texttt{lcp}(f(w\sigma\Sigma^*)) = f^p(w)^{-1} \cdot f^p(w\sigma)$*
- *For $\ltimes \notin \Sigma$: $\texttt{cont}_f(\ltimes, w) = \texttt{lcp}(f(w\Sigma^*))^{-1} \cdot f(w) = f^p(w)^{-1} \cdot f(w)$*

Recall the ISL_2 treatment of palatalization in Japanese from Section 2.1.2 and consider the input /hosimasu/. This input gets mapped to the string [hoɕimasu] which has the same length as the input, but the relevant transducer does not transform each input element to a single output element as this mapping unfolds. For instance, the contribution of each /s/ is actually the empty string [λ] in this mapping since the corresponding transducer needs to wait and see whether they are immediately followed by /i/. As a consequence of this postponement, the contribution of the /i/ following the first /s/ is actually [ɕi] and the contribution of the /u/ following the second /s/ is actually [su].

Consider now what a function would look like if each input element specified its own tier such that the contribution of that input element can be determined by that tier alone. Looking at the other tiers provides information that is either redundant or irrelevant for determining the contribution of that input element. Such a function would effectively be conducting multiple Remember for Copying/Opposing procedures in parallel. I formalize this idea of *target specification* by defining a restricted subclass of MTSL functions in the following way. I refer to this as the strong version of target specification, since I also define a slightly more permissive version below.

Definition 4.4 Target specification (strong)**.**
An IMTSL$_k$ function $f : \Sigma^* \rightarrow \Delta^*$ is strongly target specified *if for each $i \in \Sigma \cup \{\ltimes\}$ there is a tier $T_i \subseteq \Sigma$ such that for all $w_1, w_2 \in \Sigma^*$:*

$$[\mathtt{suff}^{k-1}_{T_i}(w_1) = \mathtt{suff}^{k-1}_{T_i}(w_2)] \Longrightarrow [\mathtt{cont}_f(i, w_1) = \mathtt{cont}_f(i, w_2)]$$

An OMTSL$_k$ function $f : \Sigma^* \rightarrow \Delta^*$ is strongly target specified *if for each $i \in \Sigma \cup \{\ltimes\}$ there is a tier $T_i \subseteq \Delta$ such that for all $w_1, w_2 \in \Sigma^*$:*

$$[\mathtt{suff}^{k-1}_{T_i}(f^p(w_1)) = \mathtt{Suff}^{k-1}_{T_i}(f^p(w_2))] \Longrightarrow [\mathtt{cont}_f(i, w_1) = \mathtt{cont}_f(i, w_2)]$$

Representing Kikongo's pair of long-distance processes using a strongly target-specified MTSL$_2$ function appropriately captures the intuition that the most recent vowel has no bearing on whether /l/ or /d/ becomes [n] and the intuition that the presence of a preceding nasal consonant has no bearing on whether /i/ lowers to [e] or /u/ lowers to [o]. The phoneme /l/ specifies a tier containing only nasal consonants, and $\mathtt{cont}_f$(l, w) will be [n] if the tier suffix of w or $f^p(w)$ is a nasal consonant, or will be [l] if the tier suffix of w or $f^p(w)$ is empty (i.e., there is no preceding nasal consonant). The phoneme /d/ happens to specify the same tier, and behaves identically to /l/ except that $\mathtt{cont}_f$(d, w) is [d] when the tier suffix of w or $f^p(w)$ is empty. Lowering harmony among vowels is then enforced by having /i/ and /u/ each specify a tier of only non-high vowels. The contributions of /i/ and /u/ relative to an input string will then be the corresponding mid vowel if the tier suffix of w or $f^p(w)$ is a non-high vowel, else the contributions are faithful.

The strong version of target specification works well when the output of each input element depends on just one piece of information, and has desirable consequences for learnability (see Chapter 7). In reality, however, the output of a given input element often depends on multiple pieces of information, and for that reason I also consider a weaker version of target specification. The weak version allows each $i \in \Sigma \cup \{\ltimes\}$ to specify its own *set* of associated tiers, so long as the set forms a strict superset-subset hierarchy. What this means is that if an input element /x/ specifies multiple tiers, each tier in the x-specified set must be a strict subset of the next largest tier in the x-specified set (if such a larger tier exists), but the tiers in the x-specified set are free to relate in any way to the tiers specified by a different input element /y/.

Definition 4.5 Target specification (weak)**.**
An IMTSL$_k$ function $f : \Sigma^* \to \Delta^*$ is weakly target specified *if for each $i \in \Sigma \cup \{\ltimes\}$ there is a finite superset-subset hierarchy Θ_i of tiers $T \subseteq \Sigma$ such that for all $w_1, w_2 \in \Sigma^*$:*

$$[\bigwedge_{T \in \Theta_i} [\mathtt{suff}_T^{k-1}(w_1) = \mathtt{suff}_T^{k-1}(w_2)]] \implies [\mathtt{cont}_f(i, w_1) = \mathtt{cont}_f(i, w_2)]$$

An OMTSL$_k$ function $f : \Sigma^* \to \Delta^*$ is weakly target specified *if for each $i \in \Sigma \cup \{\ltimes\}$ there is a finite superset-subset hierarchy Θ_i of tiers $T \subseteq \Delta$ such that for all $w_1, w_2 \in \Sigma^*$:*

$$[\bigwedge_{T \in \Theta_i} [\mathtt{suff}_T^{k-1}(f^p(w_1)) = \mathtt{suff}_T^{k-1}(f^p(w_2))]] \implies [\mathtt{cont}_f(i, w_1) = \mathtt{cont}_f(i, w_2)]$$

Requiring an input segment's multiple associated tiers to fall into a strict superset-subset hierarchy may seem arbitrary; nevertheless, doing so has positive theoretical consequences. First and foremost, without any restrictions on the structure of the specified tier set, we would simply be providing an alternative definition of the general MTSL functions, so weak target specification must impose *some* restriction to be useful at all. Second, this particular restriction will prove beneficial for learning, since it drastically reduces the number of possible tier combinations that can influence a given input element. Aksënova & Deskmukh (2018) show that even with a small inventory of 10 elements, there are 1022 ways to create a superset-subset pair, 511 ways to create a pair of fully disjoint sets, and 27990 ways to create a pair of partially overlapping sets. This last number is already 95% larger than the other two combined, and the difference only increases as the size of the alphabet grows. Furthermore, not only do we have a remarkably reduced hypothesis space, but Chapter 7 will show how the superset-subset structure can itself be exploited to efficiently learn at least some weakly target-specified MTSL functions from positive data. Finally, research into multi-tiered languages suggests that we generally do not see a single element appear in phonotactic restrictions on two partially overlapping tiers, whereas we do see cases of a single element appearing in phonotactic restrictions on multiple tiers that fall into a superset-subset relationship (Aksënova & Deskmukh, 2018; McMullin et al., 2019). For example, sibilants in Imdlawn Tashlhiyt agree in both anteriority and voicing, but the voicing agreement is blocked by voiceless obstruents (Elmedlaoui, 1995; Hansson, 2010; McMullin, 2016). McMullin (2016) and Aksënova & Deskmukh (2018) analyzed this as the result of anteriority harmony being enforced on a tier containing only sibilants and voicing harmony being enforced on a superset tier containing all sibilants as well as the voiceless obstruents. I therefore conclude that the superset-subset criterion is a valid restriction, and Section 4.3 will provide ample further support by showing how the weakly target-specified functions permit intuitive analyses of otherwise challenging phenomena. Before this demonstration can happen, the next section provides some necessary comments on representing multi-tiered functions.

4.2 Visualizing multi-tiered functions

Including multiple tiers increases computational complexity, but it does not push us beyond the subsequential boundary. An MTSL function is then a type of subsequential function and, like the SL and TSL functions, can be modeled with an appropriate finite-state transducer. The only substantial difference between a TSL transducer and an MTSL transducer is that states in an MTSL transducer must be labelled with a *set* of strings rather than just a single string. The tiers are in one-to-one correspondence with the strings in the label, and a tier's string represents the suffix on the associated tier. Unfortunately, MTSL transducers get fairly large fairly quickly, since the number of potentially necessary states is equal to the product of each tier's cardinality; a simple case using one tier of four elements and one tier of five elements can require up to $4 \times 5 = 20$ states. This is not necessarily an issue from a computational perspective, since even a massive transducer is relatively inexpensive in terms of memory, and it is always possible to minimize a transducer.[2] It does, however, make MTSL transducers impractical for illustrative purposes. This section therefore outlines an alternative way of visualizing the inner workings of multi-tiered functions that will be used for the remainder of this chapter.

Usually, we are only interested in the output behaviour of one or two input elements, rather than the output behaviour of the entire input alphabet. It is also usually enough to consider a small handful of mappings to illustrate the overall workings of a phonological function, rather than show what will happen in every conceivable case. Accordingly, I will showcase various multi-tiered functions by providing strategic derivations step by step, focusing on the transformation of specific input elements and the exact causes thereof. This is easily done with a table as in Table 4.1, where each row corresponds to one step in a transduction and the columns correspond to various bits of information about the step in question. The table shows a derivation for the Kikongo mapping /meng-ulu/ $\rightarrow$ [meng-ono] 'hate-PERF.PASS' in which both nasal consonant harmony and vowel height harmony apply.

In this OMTSL$_2$ derivation, we are only interested in the behaviour of /u/ and /l/, so only the tiers associated to these input elements are considered in the table. Underlying /u/ specifies the tier T_u containing all non-high vowels and underlying /l/ specifies the tier T_l containing all nasal consonants. The noteworthy steps in this derivation are steps 5 through 7. Once we reach step 5, the /u/ tier window contains [e] and the /l/ tier window contains [n]. Step 5's input is /u/ which must be output as [o] because the /u/ tier window contains a non-high vowel. The produced [o] is a non-high vowel and so enters the /u/ tier window, pushing the previously remembered [e] out of memory. Step 6's input is /l/ which must be output as [n] because the /l/ tier window

[2]Given an MTSL transducer as input, this minimization would provide the smallest subsequential transducer that computes the same function.

Step	/u/ Tier Suffix (non-high vowels)	/l/ Tier Suffix (nasal consonants)	Input	Output
1	[]	[]	/m/	[m]
2	[]	[m]	/e/	[e]
3	[e]	[m]	/n/	[n]
4	[e]	[n]	/g/	[g]
5	[e]	[n]	/u/	[o]
6	[o]	[n]	/l/	[n]
7	[o]	[n]	/u/	[o]

Table 4.1: Kikongo derivation with simultaneous nasal and height harmony

contains a nasal consonant. A vacuous change in tier contents then occurs: the produced [n] is a nasal consonant and so enters the /l/ tier window, pushing the previously remembered [n] out of memory. Finally, step 7 is a repeat of step 5 except that the transformation /u/ $\rightarrow$ [o] happens in response to the mid vowel [o] rather than the mid vowel [e]. The full input and output strings can be obtained by concatenating the entries in the input and output columns, in order.

It is easy to confirm from the derivation that the Kikongo pattern is a strongly target-specified OMTSL$_2$ function. The output fate of a given input element (if it ever changes) depends on the most recently produced element belonging to just one tier: in the case of /u/ a tier of non-high vowels and in the case of /l/ a tier of nasal consonants. Such strongly target-specified MTSL functions are perhaps the least interesting type of multi-tiered functions from a theoretical standpoint since they are mainly a means to describe a collection of non-interacting non-local processes. Of much more theoretical interest are the weakly target-specified MTSL functions, since assigning a hierarchy of tiers to an input element allows us to track multiple related sources of information and mediate their influences. This is best seen through examples, plenty of which are provided in the next Section, organized according to the type of behaviour they exhibit.

4.3 Capabilities of multi-tiered functions

In the above, I have shown how the strongly target-specified MTSL functions (and by extension the weakly target-specified ones) can model multiple long-distance processes when each does not interact with any other patterns. This section now considers a range of individual patterns and pattern interactions that lie outside the range of strongly target-specified MTSL functions. I show that each of the considered patterns and interactions are simply and intuitively captured by the weakly target-specified MTSL functions defined in Section 4.1. In order, I consider cases where a

local process overrides a non-local one (Section 4.3.1), cases where a non-local process overrides a local one (Section 4.3.2), cases where a single target is subject to multiple distinct harmonies (Section 4.3.3), cases where some segments act as harmony triggers only in the absence of canonical triggers (Section 4.3.4), and cases that require a finite amount of lookahead (Section 4.3.5).

4.3.1 Local overriding non-local

Recall from Section 3.3.1 that Tamashek Tuareg contains a regressive process of long-distance labial dissimilation which fails to apply when an [n] deriving from /m/ would be directly adjacent to a labial consonant. To model this with a weakly target-specified OMTSL_2 function, we can associate /m/ with a first tier $All = \Delta$ containing all elements in the output alphabet and a second tier $Lab = \{\text{m, b, f}\}$ containing all and only the labial obstruents. Two derivations are sufficient to show how this setup can achieve the desired interaction between the local and non-local process. To account for the regressive directionality, I assume that we read the input string backwards, reversing the output string once we are done to get the correct segment order.

The first derivation is shown in Table 4.2 for the mapping /ɑ-m-ænɑm/ → [ɑ-n-ænɑm] 'one who is fond' from Heath (2005, p. 540). The step that interests us is step 5, where we need to decide what to do with the /m/ being read. The first tier specified by /m/ is tracking everything and its window contains [ɑ], meaning that the immediately previous output segment is a vowel. We are therefore not directly adjacent to a consonant, and therefore not under any pressure from the rule of place assimilation. The second tier specified by /m/ is tracking only labial consonants and its window contains [m], which means that a labial consonant occurs somewhere previously in the output string. Combining this knowledge with our knowledge of the contents in the first tier's window, we know that this labial consonant is *not* adjacent to whatever we produce on this step. We therefore need to apply the rule of non-local labial dissimilation, writing [n] for /m/.

Step	/m/ Tier 1 (all elements)	/m/ Tier 2 (labials)	Input	Output
1	[]	[]	/m/	[m]
2	[m]	[m]	/ɑ/	[ɑ]
3	[ɑ]	[m]	/n/	[n]
4	[n]	[m]	/ɑ/	[æ]
5	[æ]	[m]	/m/	[n]
6	[n]	[m]	/ɑ/	[ɑ]

Table 4.2: Tamashek Tuareg derivation with successful dissimilation (regressive)

The second derivation is shown in Table 4.3 for the mapping /æm-bæbbɑ/ → [æm-bæbbɑ] 'car-

ried each other' from Heath (2005, p. 476). The step that interests us is again step 5, where we need to decide what to do with the /m/ being read. The first tier specified by /m/ is tracking everything and its window contains [m], meaning that the immediately previous output segment is a labial consonant. We therefore need to make sure that whatever we produce obeys place assimilation, and so we write [m] for /m/. The contents of the second tier's window are irrelevant for this step of this derivation, since local place assimilation takes priority over long-distance dissimilation.

Step	/m/ Tier 1 (all elements)	/m/ Tier 2 (labials)	Input	Output
1	[]	[]	/ɑ/	[ɑ]
2	[ɑ]	[]	/b/	[b]
3	[b]	[b]	/æ/	[æ]
4	[æ]	[b]	/b/	[b]
5	[b]	[b]	/m/	[m]
6	[m]	[m]	/æ/	[æ]

Table 4.3: Tamashek Tuareg derivation with dissimilation blocked (regressive)

More generally, the behaviour of /m/ in this function can be summarized as follows. If $\texttt{suff}^1_{All}(f^p(w))$ is a labial obstruent, then $\texttt{cont}_f(\text{m}, w)$ = [m] since /m/ is required to assimilate in place with the adjacent obstruent. If $\texttt{suff}^1_{All}(f^p(w)) \neq$ [b] and $\texttt{suff}^1_{Lab}(f^p(w)) \neq \lambda$, then $\texttt{cont}_f(\text{m}, w)$ = [n] since /m/ is not immediately adjacent to a labial consonant, but nonetheless is preceded non-locally by a labial and must dissimilate. In all other cases, /m/ surfaces faithfully as [m]. Note especially how $Lab \subset All$, and so the function meets the definition of weak target specification.

4.3.2 Non-local overriding local

Local processes do not always override non-local processes; indeed, the opposite is witnessed in the Samala language, also known as Ineseño Chumash. The language is well-known for its process of regressive sibilant harmony, whereby the anteriority of the rightmost sibilant overrides the anteriority of all sibilants to the left. As it turns out, the language contains an additional rule affecting the sibilant /s/. Namely, /s/ regressively palatalizes to [ʃ] when immediately followed by [t], [n], or [l] (Applegate, 1972; Poser, 1982, 1993; McCarthy, 2007; Hansson, 2010; McMullin, 2016). The long-distance process is given priority here, such that the local sequences [sn], [st] and [sl] are permitted precisely when palatalization would create a disharmonic sequence of non-local sibilants. Some accounts of the data claim that the local process takes priority; see Heinz & Idsardi

(2010) for a discussion and resolution of this inconsistency. Data exemplifying the two processes and their interaction are provided in (48) through (50) taken from Applegate (1972, p. 117-120).

(48) Unbounded sibilant harmony
/s-xalam-ʃ/ → [ʃ-xalamʃ] 'it is wrapped'

(49) Local palatalization
/s-niʔ/ → [ʃ-niʔ] 'his neck'

(50) Harmony overrides palatalization
/s-net-us/ → [s-net-us] 'he does it to him'

We can also describe this with a weakly target-specified OMTSL_2 function. In this case, the contribution of /s/ is dependent on two tiers. The first tier $All = \Delta$ contains everything, and it is the tier relative to which local palatalization is considered. The second tier $Sib = \{\text{s}, \text{ʃ}\}$ contains only the sibilant sounds, and it is the tier relative to which sibilant harmony is considered. Once again, two derivations are sufficient to show how this setup can achieve the desired interaction between the local and non-local process. Furthermore, as I did for the Tamashek Tuareg analysis above, I assume that we read the input string backwards to account for the regressive application of both processes.

The first derivation is shown in Table 4.4 for the mapping /s-niʔ/ → [ʃ-niʔ] 'his neck'. The step that interests us is step 4, where we need to decide what to do with the /s/ being read. The first tier specified by /s/ is tracking everything and its window contains [n]. We are therefore directly adjacent to a consonant that can trigger palatalization, but will this palatalization be blocked by harmony? The second tier specified by /s/ is tracking only sibilants and its window is empty, which means that no sibilant has yet been produced in the output string. Combining this knowledge with our knowledge of the contents in the first tier's window, we can see that we do in fact need to apply the rule of local palatalization, writing [ʃ] for /s/.

Step	/s/ Window 1 (all elements)	/s/ Window 2 (sibilants)	Input	Output
1	[]	[]	/ʔ/	[ʔ]
2	[ʔ]	[]	/i/	[i]
3	[i]	[]	/n/	[n]
4	[n]	[]	/s/	[ʃ]

Table 4.4: Samala derivation with local palatalization

The second derivation is shown in Table 4.5 for the mapping /s-net-us/ → [s-net-us] 'he does it to him'. The step that interests us is step 6, where we need to decide what to do with the /s/ being

read. The second tier specified by /s/ is tracking only sibilants and its window contains [s], which means that a [+anterior] sibilant has previously been produced somewhere in the output string. We therefore need to make sure that whatever we produce obeys sibilant harmony, and so we write [s] for /s/. The contents of the first tier's window are irrelevant for this step of this derivation, since non-local sibilant harmony takes priority over local palatalization.

Step	/s/ Window 1 (all elements)	/s/ Window 2 (sibilants)	Input	Output
1	[]	[]	/s/	[s]
2	[s]	[s]	/u/	[u]
3	[u]	[s]	/t/	[t]
4	[t]	[s]	/e/	[e]
5	[e]	[s]	/n/	[n]
6	[n]	[s]	/s/	[s]

Table 4.5: Samala derivation with harmony overriding palatalization

More generally, the behaviour /s/ in this function can be summarized as follows. If $\texttt{suff}^1_{Sib}(f^p(w)) = \lambda$ and $\texttt{suff}^1_{All}(f^p(w))$ is one of [t], [n], or [l], then $\texttt{cont}_f$(s, w) = [ʃ] since it is adjacent to a palatalization trigger but not preceded non-locally by a [−anterior] sibilant. If $\texttt{suff}^1_{Sib}(f^p(w))$ = [ʃ], then $\texttt{cont}_f$(s, w) = [ʃ] since /s/ is preceded non-locally by a [−anterior] sibilant and must harmonize. In all other contexts, /s/ surfaces faithfully as [s]. Once again, $Sib \subset All$ and so the function meets the definition of weak target specification.

4.3.3 Split harmony

In the patterns analyzed above, the element undergoing (or failing to undergo) a change is ultimately reacting to the presence of a single other element. While single triggers are typically the case, there do exist processes where a single element reacts to multiple other elements. An example of this comes from the Imdlawn dialect of Tashlhiyt. The language's causative prefix /s-/ agrees in both anteriority and voicing with a following sibilant, as shown in (51), although the voicing dimension of the harmony is blocked by an intervening voiceless obstruent (Elmedlaoui, 1995; Hansson, 2010; McMullin, 2016), as shown in (52).

(51) Anteriority and voicing harmony

/s-uga/	[sː-uga]	'CAUS-be.evacuated'
/s-asːtwa/	[s-asːtwa]	'CAUS-settle'
/s-bːukʃːa/	[ʃ-bukʃːa]	'CAUS-be.full.to.overflowing'
/s-bruzːa/	[z-bruzːa]	'CAUS-crumble'
/s-gruʒːm/	[ʒ-gruʒːm]	'CAUS-be.extinguished'

(52) Voicing harmony blocked

/s-ukz/	[sː-ukz]	'CAUS-recognize'
/s-mħaraʒ/	[ʃ-mħaraʒ]	'CAUS-get.angry.with.each.other'

We can also describe this with a weakly target-specified OMTSL$_2$ function. In this case, the contribution of /s/ is dependent on two tiers. Voicing harmony is considered relative to the first tier Voi which contains all sibilants (voiced and voiceless) plus all voiceless obstruents. Anteriority harmony is considered relative to the second tier Ant which contains only the sibilants (voiced and voiceless). Once again, two derivations are sufficient to show how this setup can achieve the desired result. These derivations read the input string backwards to account for the regressive application of both processes.

The first derivation is shown in Table 4.6 for the mapping /s-gruʒːm/ → [ʒ-gruʒːm] 'CAUS-be.extinguished'. The step that interests us is step 6, where we need to decide what to do with the /s/ being read. The second tier specified by /s/ is tracking only sibilants and this window contains [ʒ]. We are therefore preceded by a [+voice] and [−anterior] sibilant and so are potentially subject to both voicing and anteriority harmony. The anteriority harmony will always apply, but we need to inspect the other tier window to confirm whether to apply voicing harmony. The first tier specified by /s/ is tracking sibilants in addition to voiceless obstruents and its window contains [ʒ]. The first and second windows match, which tells us that no voiceless obstruent has been seen since the most recent sibilant. We therefore need to apply both harmony rules, and so we write [ʒ] for /s/.

Step	/s/ Window 1 (sibilants and voiceless obstruents)	/s/ Window 2 (sibilants)	Input	Output
1	[]	[]	/m/	[m]
2	[]	[]	/ʒ/	[ʒ]
3	[ʒ]	[ʒ]	/u/	[u]
4	[ʒ]	[ʒ]	/r/	[r]
5	[ʒ]	[ʒ]	/g/	[g]
6	[ʒ]	[ʒ]	/s/	[ʒ]

Table 4.6: Imdlawn Tashlhiyt anteriority harmony, with unblocked voicing harmony

The second derivation is shown in Table 4.7 for the mapping /s-mħaraʒ/ → [ʃ-mħaraʒ] 'CAUS-get.angry.with.each.other'. The step that interests us is step 7, where we need to decide what to do with the /s/ being read. The second tier specified by /s/ once again contains [ʒ], so we potentially need to apply both voicing and anteriority harmony. The first tier contains [ħ], however, which tells us that a voiceless obstruent comes between /s/ and the most recent sibilant. We therefore need only apply the anteriority rule, and so we write [ʃ] for /s/.

Step	/s/ Window 1 (sibilants and voiceless obstruents)	/s/ Window 2 (sibilants)	Input	Output
1	[]	[]	/ʒ/	[ʒ]
2	[ʒ]	[ʒ]	/a/	[a]
3	[ʒ]	[ʒ]	/r/	[r]
4	[ʒ]	[ʒ]	/a/	[a]
5	[ʒ]	[ʒ]	/ħ/	[ħ]
6	[ħ]	[ʒ]	/m/	[m]
7	[ħ]	[ʒ]	/s/	[ʃ]

Table 4.7: Imdlawn Tashlhiyt anteriority harmony, with blocked voicing harmony

More generally, the behaviour /s/ in this function can be summarized as follows. If $\texttt{suff}^1_{Voi}(f^p(w)) = \texttt{suff}^1_{Ant}(f^p(w))$ is a sibilant, then an input /s/ harmonizes in both anteriority and voicing with that sibilant. Note that both tiers contain all sibilants and so the two tier suffixes can never be different sibilants. If $\texttt{suff}^1_{Ant}(f^p(w))$ is a sibilant and $\texttt{suff}^1_{Voi}(f^p(w))$ is a voiceless obstruent, then /s/ harmonizes in anteriority but not voicing with $\texttt{suff}^1_{Ant}(f^p(w))$. The voicing dimension of the harmony is blocked because $Ant \subset Voi$ and the voiceless obstruent necessarily intervenes between the triggering sibilant and the target sibilant as a consequence of the superset-subset relationship. In all other cases, /s/ surfaces faithfully as [s].

In this Imdlawn Tashlhiyt case, the harmony target agrees with the harmony source along two dimensions, but there are cases where a single target can agree with multiple sources in specific configurations. Recall from Section 3.2.1 that suffixal vowels in Turkish typically agree in backness with the next vowel to the right. An interesting exception, however, concerns the consonants /k/, /g/, and /l/ which are contrastively [+back] and have the [−back] counterparts /kʲ/, /gʲ/, and /lʲ/ (Clements & Sezer, 1982). If one of these consonants intervenes, the suffixal vowel will take its backness value from the consonant (Nevins, 2010). It would seem that all we must do is include the contrastive consonants to the tier, but this is insufficient when we consider the additional process of rounding vowel harmony in Turkish. Suffixal high vowels typically agree in backness *and*

rounding with the next vowel to the left as shown in (53) from Nevins (2010, p. 29), unless one of the consonants contrastive for [±back] intervene. If such a consonant intervenes, the suffixal high vowel takes its backness value from the consonant and its rounding value from the vowel, as shown in (54) from Nevins (2010, p. 54).

(53) Two harmonies, same source

ip-in	'rope-GEN'
sap-ɯn	'stalk-GEN'
jyz-yn	'face-GEN'
son-un	'end-GEN'

(54) Two harmonies, two sources

usulʲ-y	'system-ACC.SG'
sualʲ-i	'question-ACC.SG'

Let us assume that the harmonizing suffixal high vowels lack values for [±back] and [±round] in their underlying form, which I will represent as /I/. The potentially split harmony can be analyzed with the following weakly target-specified OMTSL$_2$ function where the contribution of /I/ is dependent on two tiers. Backness harmony is considered relative to the first tier B which contains all vowels plus all consonants contrastive for [±back]. Rounding harmony is considered relative to the second tier R which contains only the vowels. Once again, two derivations are sufficient to show how this setup can achieve the desired result.

The first derivation is shown in Table 4.8 for the mapping /son-In/ → [son-un] 'end-GEN'. The step that interests us is step 4, where we need to decide what to do with the /I/ being read. The second tier specified by /I/ is tracking only vowels and this window contains [o]. We are preceded by a [+back] and [+round] vowel and might take both feature values from this one vowel, but before we can do so, we need to check the first tier to see whether there is a closer source for backness. The first tier specified by /I/ is tracking vowels in addition to voiceless obstruents and its window contains [o]. The first and second windows match, which tells us that no consonant contrastive for [±back] has been seen since the most recent vowel. We therefore copy the backness *and* the rounding of [o], writing [u] for /I/.

The second derivation is shown in Table 4.9 for the mapping /usulʲ-I/ → [usulʲ -y] 'question-ACC.SG'. The step that interests us is step 5, where we need to decide what to do with the /I/ being read. The second tier window once again contains a [+back] and [+round] vowel, so we can potentially take both our rounding and backness value from this one source. The first tier contains [lʲ], however, which tells us that there is an even closer source for the feature [±back]. We therefore take our backness value from [lʲ] and our rounding value from [u], writing [y] for /I/.

More generally, the behaviour of /I/ in this function can be summarized as follows. If $\texttt{suff}^1_B(f^p(w)) = \texttt{suff}^1_R(f^p(w))$, then the nearest source of backness and the nearest source

Step	/I/ Window 1 (Cs contrastive for back and vowels)	/I/ Window 2 (vowels)	Input	Output
1	[]	[]	/s/	[s]
2	[]	[]	/o/	[o]
3	[o]	[o]	/n/	[n]
4	[o]	[o]	/I/	[u]
5	[u]	[u]	/n/	[n]

Table 4.8: Turkish derivation with single for backness and rounding

Step	/I/ Window 1 (Cs contrastive for back and vowels)	/I/ Window 2 (vowels)	Input	Output
1	[]	[]	/u/	[u]
2	[u]	[u]	/s/	[s]
3	[u]	[u]	/u/	[u]
4	[u]	[u]	/lʲ/	[lʲ]
5	[lʲ]	[u]	/I/	[y]

Table 4.9: Turkish derivation with different sources for backness and rounding

of rounding are both the nearest leftward vowel. If, however, $\texttt{suff}^1_B(f^p(w)) \neq \texttt{suff}^1_R(f^p(w))$, then one of the relevant consonants must intervene between the suffixal vowel and the nearest leftward vowel. In this case, the suffixal vowel will take on the backness value of the consonant; the source of rounding will, however, always be the nearest leftward vowel.

4.3.4 Preferential triggers

A fourth type of behaviour captured by weak target specification comes from cases of harmony that can be triggered by a special class of segments in the absence of a more "preferred" trigger. One such pattern exists in Uyghur, where vowels are subject to a progressive backness harmony for which the non-low front vowels [i] and [e] are reported to be transparent (Lindblad, 1990; Vaux, 2000; Mayer & Major, 2018). The language's locative suffix shows that the harmony is an active process; its vowel alternates between front [æ] and back [a] as shown in (55) taken from Mayer & Major (2018).

(55) Uyghur vowel harmony

aʁinæ-dæ ‘friend-LOC’
qoichi-da ‘shepherd-LOC’

Interestingly, dorsal consonants can trigger harmony in the absence of a harmonic vowel, with velar consonants causing front allomorphs of suffixes and uvulars causing back allomorphs (Lindblad, 1990; Vaux, 2000; Mayer & Major, 2018), as shown in (56). The harmony “prefers” to be triggered by vowels, however, which is witnessed by the fact that a back vowel can trigger back allormorphs across a velar consonant (Lindblad, 1990; Vaux, 2000; Mayer & Major, 2018), as shown in (57). Dorsal consonants only decide the front/back status of a suffix when the base lacks non-transparent vowels altogether, in a sense acting as a “last-resort trigger” (Lindblad, 1990; Vaux, 2000; Mayer & Major, 2018).

(56) Harmony with a dorsal
gezit-tæ ‘newspaper-LOC’
qiʁiz-da ‘Kyrgyz-LOC’

(57) Harmony across a dorsal

rak-ta	*rak-tæ	‘shrimp-LOC’
mæʃq-tæ	*mæʃq-ta	‘exercise-LOC’

For ease of exposition I will treat the suffix vowel as having no underlying value for [back], representing it as /A/. To capture the pattern as a weakly target-specified OMTSL$_2$ function, we need two tiers. The first tier Dor contains all non-transparent vowels in addition to dorsal consonants, and the second tier Vow contains all non-transparent vowels but excludes the dorsal consonants. Like above, two derivations are sufficient to show how this setup can achieve the desired result.

The first derivation is shown in Table 4.10 for the mapping /gezit-tA/ → [gezit-tæ] ‘newspaper-LOC’. The step that interests us is step 7, where we need to decide what to do with the /A/ being read. The first tier specified by /A/ is tracking only non-transparent vowels and its window is empty. There is thus no available vowel trigger for harmony. The second tier, however, is not empty. This tier is tracking non-transparent vowels as well as dorsal consonants and contains [g]. There is thus a dorsal consonant available as a last-resort trigger for backness harmony. Velar consonants condition front allomorphs, and so we write front [æ] for /A/.

The second derivation is shown in Table 4.11 for the mapping /rak-tA/ → [rak-ta] ‘shrimp-LOC’. The step that interests us is step 5, where we need to decide what to do with the /A/ being read. The first tier contains a velar consonant [k], so in the absence of a vowel trigger, this will condition a front allomorph. The second tier, however, is not empty and contains [a] which conditions back allomorphs. We have an available vowel trigger and so ignore the potential dorsal consonant trigger in favour of the preferred vowel trigger, writing [a] for /A/.

Step	/A/ Window 1 (harmonic vowels and dorsals)	/A/ Window 2 (harmonic vowels)	Input	Output
1	[]	[]	/g/	[g]
2	[g]	[]	/e/	[e]
3	[g]	[]	/z/	[z]
4	[g]	[]	/i/	[i]
5	[g]	[]	/t/	[t]
6	[g]	[]	/t/	[t]
7	[g]	[]	/A/	[æ]

Table 4.10: Uyghur backness harmony with a last-resort dorsal consonant trigger

Step	/A/ Window 1 (harmonic vowels and dorsals)	/A/ Window 2 (harmonic vowels)	Input	Output
1	[]	[]	/r/	[r]
2	[]	[]	/a/	[a]
3	[a]	[a]	/k/	[k]
4	[k]	[a]	/t/	[t]
5	[k]	[a]	/A/	[a]

Table 4.11: Uyghur backness harmony across a dorsal consonant

More generally, the behaviour of the vowel in the locative suffix can be described as follows. If $\texttt{suff}^1_{Vow}(f^p(w)) \neq \lambda$, then there is a non-transparent vowel in the stem that triggers harmony. If, however, $\texttt{suff}^1_{Vow}(f^p(w)) = \lambda$ and $\texttt{suff}^1_{Dors}(f^p(w)) \neq \lambda$, then all stem vowels are transparent, but there is a dorsal consonant available to trigger harmony as a last resort. The last-resort nature of dorsal consonant triggers results directly from the fact that they are on the larger but not the smaller tier.

4.3.5 Finite lookahead

Sections 4.3.1 and 4.3.2 took advantage of the fact that the superset-subset requirement imposed by weak target specification allows us to track a tier of everything. The final capability of weakly target-specified MTSL functions that I will showcase also takes advantage of such a maximally inclusive tier. Recall from section 2.2.1 that by virtue of representing the $k-1$ most recent input elements and their order, the state space of an ISL_k transducer permitted us to delay (in a sense) the

output of an input element up to a finite number of transduction steps. Delaying an input element's output in this manner allowed us to account for the effects of material on both sides of that input element, up to a fixed distance. Being able to track local and non-local information simultaneously opens up an intriguing possibility for weakly target-specified IMTSL functions. Thanks to the input-oriented local tier we can finitely postpone output decisions, making it possible to model patterns where the triggering context is (potentially) non-local in one direction but necessarily local in the other.

Consider the lowering harmony in c'Lela, presented earlier in Section 3.3.3, where suffix high vowels become mid if they are preceded by a non-high vowel in the root (Dettweiler, 2000; Pulleyblank, 2002; Archangeli & Pulleyblank, 2007; Michel, 2009; Jurgec, 2011). As I previously noted, the fact that c'Lela lowering harmony targets only vowels in absolute word-final position creates a paradox if we try to analyze it with a basic TSL function. In order to confirm that an input vowel is in word-final position, we need to postpone its output and attempt to read the next input element, but to properly apply lowering harmony we need to track a tier containing only non-high vowels. The c'Lela data are repeated in (58) for quick reference. They show that the class membership suffix /-i/ will lower to [e] when it is the only suffix following a non-high root vowel, but will be unaffected when it is followed by another suffix vowel.

(58) C'Lela lowering harmony (Dettweiler, 2000)

a.	/zis-i/	[zis-i]	'long-I.CLASS'
b.	/rek-i/	[rek-e]	'small-I.CLASS'
c.	/zis-i-ni/	[zis-i-ni]	'long-I.CLASS-ADJM'
d.	/rek-i-ni/	[rek-i-ne]	'small-I.CLASS-ADJM'

Analyzing c'Lela harmony as a weakly target-specified IMTSL_2 function is slightly more complex than the OMTSL_2 analyses from the previous sections, but is still nonetheless intuitive. The process specifically affects /i/ and /u/ so at a first glance it seems we need only consider the tier(s) specified by those two vowels. Since, however, we will be postponing output decisions in some cases, we in fact need to consider the tier(s) specified by every input element, including the word-end symbol $\ltimes$. Fortunately, the same tier hierarchy can be used for every input element in this case. All elements need a first tier All equal to the input alphabet Σ and need a second tier Low that contains only the non-high vowels. As usual, two derivations are sufficient for demonstrative purposes. The first is shown in Table 4.12 for the mapping /zis-i-ni/ $\rightarrow$ [zis-i-ni] 'long-I.CLASS-ADJM'. No harmony occurs in this derivation since there is no non-high vowel in the input. The derivation serves to show that when the second tier is empty, input alphabet elements are immediately reproduced faithfully upon being read, and nothing is written upon reading the word-end symbol.

Step	Window 1 (all segments)	Window 2 (non-high vowels)	Input	Output
1	/ /	/ /	/z/	[z]
2	/z/	/ /	/i/	[i]
3	/i/	/ /	/s/	[s]
4	/s/	/ /	/i/	[i]
5	/i/	/ /	/n/	[n]
6	/n/	/ /	/i/	[i]
7	/i/	/ /	⋉	[]

Table 4.12: C'Lela derivation with no harmony

The second derivation is shown in Table 4.13 for the mapping /rek-i-ni/ → [rek-i-ne] 'small-I.CLASS-ADJM'. Steps 3 to the end are relevant to discussion here. On step three, the first and second tier both have /e/, so /e/ is the most recently read element. We will not have postponed anything on the previous step, so we can output the incoming /k/ immediately as just [k]. On step four, the second tier has /e/ and we are reading /i/, so we potentially need to apply lowering harmony, but we do not yet know whether this /i/ is word-final and so output nothing to delay our decision. Step five has /i/ on the first tier and /e/ on the second which tells us that the output for an /i/ was delayed on the previous step. By virtue of reading something other than ⋉ we now know that that instance of /i/ is not word final, so we output it faithfully alongside the /n/ being read on the current step. For the same reason as on step four, the output for an incoming /i/ is delayed on step six. Finally, we read ⋉ on step seven, telling us that the input element on the previous step is in word-final position. The fact that the second tier contains /e/ tells us that the output for the /i/ on the first tier was delayed on the previous step. This instance of /i/ is required to lower since it is word final and preceded by a non-high vowel. We therefore write [e].

Postponing outputs by storing local information while also storing non-local information is not only useful in cases with strictly word-initial or word-final targets. Simultaneous local and non-local information storage equally lets us model word-internal patterns where the outcome of a locally-triggered process is affected (but not blocked) by a long-distance dependency. A particularly intriguing case of this comes from the Päri language, where coronal harmony (along the dental vs. alveolar dimension) is largely a static co-occurrence restriction that holds within roots but participates indirectly in locally-triggered mutations at root edges. Dental and alveolar consonants cannot co-occur within a Päri root, except for liquid consonants, which are redundantly alveolar and can freely appear in dental or alveolar roots (Andersen, 1988, 1999; Hansson, 2010). Underlying /l/ mutates into a coronal stop when immediately followed by certain vowel-initial

Step	Window 1 (all segments)	Window 2 (non-high vowels)	Input	Output
1	/ /	/ /	/r/	[r]
2	/r/	/ /	/e/	[e]
3	/e/	/e/	/k/	[k]
4	/k/	/e/	/i/	[]
5	/i/	/e/	/n/	[in]
6	/n/	/e/	/i/	[]
7	/i/	/e/	⋉	[e]

Table 4.13: C'Lela derivation with harmony

prefixes, becoming alveolar [nd] unless there is a dental stop preceding it at any distance, in which case it becomes dental [n̪d̪], giving the impression that mutation feeds harmony. A single IMTSL$_2$ function can accomplish the mutation and harmony in one fell swoop as shown by the derivation in Table 4.14 for the derivation /t̪ùol-a$_x$/ → [t̪úon̪d̪-à] 'snake-my' from (Hansson, 2010, p. 58). Note that the derivation abstracts away from the tone alternations, and I treat mutation-triggering vowels as separate entities by giving them a subscript x.

Step	Window 1 (all segments)	Window 2 (non-liquid coronals)	Input	Output
1	/ /	/ /	/t̪/	[t̪]
2	/t̪/	/t̪/	/u/	[u]
3	/u/	/t̪/	/o/	[o]
4	/o/	/t̪/	/l/	[]
5	/l/	/t̪/	/a$_x$/	[n̪d̪a]
6	/a$_x$/	/t̪/	⋉	[]

Table 4.14: Päri root mutation feeding coronal harmony

Like with the c'Lela analysis above, we actually need to keep track of the tiers specified by every input element if we are to appropriately model postponement, although as was the case for c'Lela we can use the same two tiers for all input elements in Päri. The first tier includes everything, where the second tier contains only the coronal consonants (excepting liquids). Nothing is written to the output on step 4 in Table 4.14 since we need to wait and see whether the /l/ that was just read is immediately followed by a mutation-triggering vowel /V$_x$/. The next step confirms that this is indeed the case, and because the second tier contains a dental stop, we need to mutate and harmonize at the same time, producing [n̪d̪] alongside the vowel. Had we instead read ⋉ or some

segment that does not trigger mutation, we would have simply output the /l/ as is (alongside the other segment, if any).

Using combined local and non-local information as in the c'Lela and Päri analyses raises an important question: how long are we allowed to postpone the output of an input segment? Similar to how most example patterns in Chapter 3 were TSL for $k = 2$, all examples in this Chapter so far have been MTSL for $k = 2$. With $k = 2$ we can postpone an output by a single transduction step, but we may at times need to wait even longer, requiring $k > 2$. Recall that Icelandic umlaut will raise non-initial /a/ all the way to [ʏ] but will only raise initial /a/ to [œ] since initial vowels receive primary stress. The analysis of Icelandic umlaut in Section 3.2.4 marked vowels as underlyingly stressed or unstressed, but the simplicity of Icelandic's stress system (primary stress in the first syllable) makes this approach admittedly heavy handed. Since umlaut applies regressively and we would therefore be reading the input string backwards, we can determine whether an underlying vowel receives primary stress by postponing outputs. Upon reading an /a/, we would stop writing to the output until we read the word-edge symbol or another vowel, at which time we would use the local tier to output everything that we postponed (altering /a/ as necessary according to the subset tier of umlaut triggers). According to the definition of an MTSL function, however, we can only postpone an output up to a finite number of times. The largest number of consonants between two vowels in the data from Jurgec (2011, pp. 105-109) is 3 as in the word [elskar] 'love.3SG.PRES' which would require $k = 4$.

4.4 Comparison with subsequential functions

Starting with the SL functions, this thesis has up to this point been a presentation of increasingly powerful function classes, arguing that the climb in computational complexity is necessary to capture the attested range of behaviours in long-distance phonology. All of the presented classes are strictly less powerful than the subsequential functions, which have previously been put forth as the appropriate characterisation of long-distance segmental processes (Heinz & Lai, 2013; Luo, 2017; Payne, 2017). In what follows, I will argue that the (weakly target-specified) MTSL functions offer a better model of long-distance phonological patterns than the subsequential functions. I do so by discussing two patterns that have been identified as pathological and demonstrate that they are not MTSL but can be modelled as a subsequential function. The first behaviour is what I call *modulo counting*, and is discussed in Section 4.4.1 using the example of parity-sensitive harmony. The second is what I call *minimum distance requirements*, and is discussed in Section 4.4.2 using the example of strictly beyond-transvocalic dissimilation. The problematic patterns require transducer states to represent configurations that are impossible to describe using tier suffixes alone, leaving them outside the reach of the MTSL functions.

4.4.1 Modulo counting

Within constraint-based approaches to phonology such as Optimality Theory (OT: Prince & Smolensky, 2004), harmony and dissimilation are frequently analyzed using the Agreement-by-Correspondence (ABC) framework (Rose & Walker, 2004; Hansson, 2010; Bennett, 2015). In an ABC analysis, *correspondence* constraints require similar segments to stand in correspondence, and *identity* constraints require corresponding segments to agree on certain features. Harmony occurs when one of two corresponding segments changes its underlying feature value in order to satisfy an identity constraint. Dissimilation occurs when one of two would-be corresponding segments changes its underlying feature value in order to become sufficiently dissimilar and avoid entering into correspondence. While the ABC framework is generally successful at modelling long-distance harmony and dissimilation patterns, McMullin (2016) and McMullin & Hansson (2016) point out that commonly-used ABC constraints can produce a bizarre pattern of transvocalic harmony that is sensitive to the parity of potential harmony targets.

In a transvocalic sibilant harmony system, sibilants agree in anteriority specifically if they are separated by at most a single vowel. When there is a sequence of multiple such sibilants, we expect the anteriority value of the first sibilant to be inherited by all sibilants in the chain. Essentially, the value gets passed from each sibilant to the next, skipping over one vowel on each pass. In the parity-sensitive system, however, the chain of sibilants will organize itself into non-overlapping pairs, with each first member of a pair passing its value to the second member of the pair. Furthermore, there is no interaction between sibilants in different pairs. As a result, the system permits disharmonic sequences so long as the disharmony straddles a 'pair boundary'. For example, we would expect that /saʃaʃasasa/ maps to [sasasasasa] but it will instead map to [$s_i$as$_i$a$\int_j$a$\int_j$as$_k$a], where pair membership is marked with subscripts. Importantly, the grouping into pairs is done without any reference to prosodic structure; it might look like harmony is being limited to within a single binary foot, but the pair boundaries do not necessarily line up with any prosodic boundary.

To my knowledge, there are no experiments that investigate the learnability of parity-sensitive harmony. Nonetheless, I believe that it is safe to consider parity-sensitive behaviour pathological for segmental phenomena, and this would extend to modulo counting for any positive integer (i.e., patterns that need to count in multiples of some integer). Accordingly, models of long-distance segmental processes should exclude the possibility of modulo counting.

Parity sensitivity (and modulo counting more generally) is necessarily not MTSL, since a sequence can be arbitrarily long and we would need an MTSL function's window (as represented by the state labels of its FST) to contain this entire sequence in order to determine its parity (or its remainder modulo a different integer). One can, of course, simulate modulo counting up to a

particular multiple with a sufficiently large value of k, but the MTSL function's memory window will eventually get saturated and cannot become unsaturated. For example, let us try modelling parity-sensitive sibilant harmony using an ITSL_4 transducer whose tier is $T = \{\text{s}, \text{ʃ}\}$. Suppose that we are on the last steps of reading /soso-ʃ/, /sososo-ʃ/ and /sososososo-ʃ/. In the first, the state label /ss/ has an even number of sibilants and so we do harmonize with the most recent one. In the second, the state label /sss/ has an odd number of sibilants and so we harmonize with the most recent one. In the third, the state label /sss/ leads us to (falsely) believe that there is an odd number of preceding sibilants, and so we harmonize. This demonstration uses a basic ITSL function but the same point would apply to an IMTSL function. Namely, state labels will eventually become saturated and cannot subsequently become unsaturated, so the modulo counting can only be enforced up to a particular multiple.

Unfortunately for the subsequential hypothesis, parity sensitivity (and modulo counting more generally) is readily captured by a subsequential FST, as shown in Figure 4.1. To improve the transducer's readability, I consider a segment inventory of only vowels and sibilant fricatives and consider only inputs with perfect consonant-vowel alternation. Neither of these assumptions crucially affect the discussion. The rule computed by the transducer is expressed in (59), where V represents a vowel, S represents a sibilant and $\{\}^{2n}$ represents an even number of instances (including 0) of some sequence.

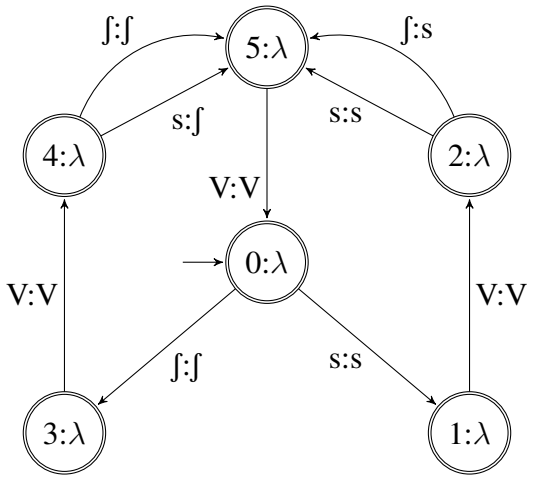


Figure 4.1: A subsequential FST that computes parity-sensitive transvocalic harmony

$$\text{(59)} \quad \left[\begin{array}{c} +\text{sibilant} \end{array} \right] \longrightarrow \left[\begin{array}{c} \alpha\text{anterior} \end{array} \right] \Big/ \{\text{SV}\}^{2n} \left[\begin{array}{c} +\text{sibilant} \\ \alpha\text{anterior} \end{array} \right] \text{V} \_\_\_\_$$

To see how this transducer computes parity-sensitive transvocalic harmony, consider the input /soʃoʃoso/. Reading the first sibilant-vowel syllable produces [so] bringing us to state 2. The next

sibilant-vowel pair syllable must harmonize since it forms a pair with the previous one, so we write [so] for /ʃo/. Doing so returns us to state 0, which means that the following syllable starts a new pair so /ʃo/ is free to surface faithfully bringing us to state 4. Finally, the fourth syllable forms a pair with the previous one, so we write [ʃo] for /so/. This gives us the full mapping /soʃoʃoso/ → [sosoʃoʃo].

4.4.2 Minimum distance requirements

In a pattern with *minimum distance requirements*, a trigger affects a target only if it is some set distance or further from the target. An example of such a system would be *strictly beyond-transvocalic dissimilation*, where two segments dissimilate only when they are separated by *at least* a vowel and a consonant; no dissimilation occurs across just a vowel (i.e., transvocalic contexts). McMullin (2016) and McMullin & Hansson (2019) ran a series of artificial grammar learning experiments, finding that strictly beyond-transvocalic patterns of liquid harmony and dissimilation are not reliably acquired in an experimental setting. Even when given unambiguous training data, participants tended to either infer a more typical unbounded pattern or else not infer any pattern. I accordingly propose that phonological processes should be prevented from containing minimum distance requirements like those in a beyond-transvocalic system. The subsequential hypothesis fails in this regard, since states in a subsequential FST can represent arbitrary configurations such as "n or more syllables away from the most recent x", and so minimum distance requirements are readily enforceable by a subsequential FST.

The FST in Figure 4.2 provides a concrete demonstration of how beyond-transvocalic dissimilation is subsequential. For ease of interpretability I consider only inputs with perfect consonant-vowel alternation, although this assumption does not crucially affect the discussion. The rule computed by the transducer is expressed in (60), where V represents a vowel, C represents a consonant other than [l], and $\{\text{CV}\}^+$ represents a string of one or more CV syllables. To see how this transducer computes first-last harmony, consider the input /lolokol/. Starting in state 0, we produce [l] upon reading word-initial /l/ bringing us to state 1. Next we read /o/ which produces [o] and brings us to state 2. On the following step, we read another /l/ but this is not separated from the previous one by a beyond-transvocalic distance and is produced faithfully as [l], moving us back to state 1. The next input elements are /o/, /k/ and /o/ which together produce [oko] and move us to state 4. After that comes another /l/ but this time we *are* separated from the previous /l/ by a beyond-transvocalic distance. Accordingly, we dissimilate and produce [r], so the full mapping is then /lolokol/ → [lolokor]. Once state 4 is reached, the machine can only cycle between states 4 and 5, so if any more instances of /l/ are read in this situation, all of these will dissimilate to become [r].

(60) $l \longrightarrow r \;/\; lV\{CV\}^{+} \_\_\_\_$

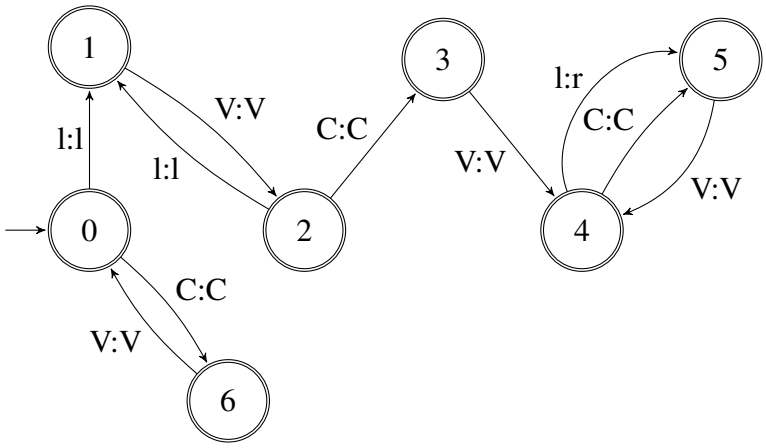


Figure 4.2: A subsequential FST that computes beyond-transvocalic liquid dissimilation

To impose a minimum distance requirement for the triggering of a process using an MTSL function, the trigger and intervening material must be on the same tier, else we could not know the distance between trigger and target. There is, however, a finite maximum of material that can factor into the state label, and the trigger will consequently get pushed out of this window eventually and forgotten. MTSL functions are subject to a *maximum distance limit* if they wish to impose a minimum distance requirement, a constraint that does not apply to subsequential functions.

Unfortunately, while MTSL functions cannot compute true minimum distance limits, they can capture a similar behaviour where a particular intervener is mandatory. Consider the IMTSL_2 transducer in Figure 4.3, where the left-hand portion of a state label represents the most recent element from the tier of all consonants (including liquids) and where the right-hand portion of a state label represents the most resent element from the tier of just liquid consonants. This transducer computes what one might call strictly trans-consonantal dissimilation, since at least one non-liquid consonant must intervene between a potential trigger and target for dissimilation to occur. If we require inputs to have perfect consonant-vowel alternation, this becomes indistinguishable from strictly beyond-transvocalic dissimilation. From the point of view that the (weakly target-specified) MTSL functions are an accurate characterization of long-distance phonology, it is then surprising that McMullin & Hansson's (2016, 2019) participants failed to learn the target pattern hidden in the training data since they were exposed to words with perfect consonant-vowel alternation.

Taken at face value, this particular overgeneration is a point in disfavour of the MTSL hypothesis, although learnability considerations offer a potential way to mitigate its seriousness. As will be shown in Chapter 7, learning tier-based functions can be divided into two major steps: discovering

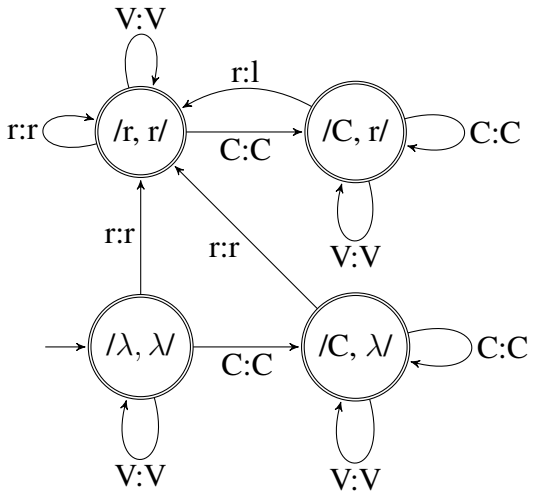


Figure 4.3: An IMTSL_2 transducer that computes strictly trans-consonantal dissimilation.

the necessary tiers and building a transducer using those tiers. The method for discovering the tiers of a weakly target-specified MTSL function that will be developed in Chapter 7 relies on statistical comparisons, and so is highly susceptible to the amount and quality of the training data. One could accordingly argue that being MTSL or lower in complexity is a necessary but not sufficient condition for being a possible long-distance phonological pattern. McMullin & Hansson's (2016, 2019) results could then be interpreted to reflect the possibility that the distributional evidence necessary to learn certain MTSL functions makes them unlikely to be acquired under realistic conditions.

4.5 Chapter summary

In order to address particular shortcomings of the TSL functions, this chapter began by extending them so that they can operate relative to multiple tiers. I deemed the full power of unrestricted tier sets to be excessive, however, and so I additionally defined strong and weak versions of a property called target specification, which is inspired by search-and-copy models of vowel harmony like the one in Nevins (2010). When a multi-tiered function is weakly target-specified, each input element specifies the tier(s) that are relevant to determining its output fate, and each of these target-specified sets must form a strict superset-subset hierarchy. I showed how weakly target-specified functions can account for a variety of behaviours that are otherwise challenging to model. These included split harmony, trigger preference, finite lookahead, and opaque interactions between local and non-local processes. Finally, I compared my weakly target-specified functions with the subsequential functions, which are a current model of long-distance phonological processes. The comparison demonstrated that the subsequential functions can capture modulo counting behaviour and can

enforce minimum distance requirements, two pathological patterns that are impossible to describe as a multi-tiered function. Taking these results together, I conclude that the weakly target-specified functions are a worthwhile extension of tier-based functions that greatly expand their typological coverage with minimal overgeneration.

Chapter 5

Bi-directional cases

All processes analyzed thus far share an important property aside from their adherence to tier-based strict locality: they are unidirectional, or at the very least can be modelled by a single unidirectional transducer. In these processes, any unbounded influence applies from a consistent direction; when some input element is subject to a potentially long-distance dependency, that dependency either always influences instances of that element from the right or always influences instances of that element from the left. While this generally seems to be the case for segmental phenomena, there do exist bidirectional patterns where instances of some input element respond to rightward information in some mappings but leftward information in others.

A common tool in the computational literature known as function composition (described in section 5.1) can very easily handle bidirectionality, but special care must be taken since free composition can have undesirable consequences. Previous work by Heinz & Lai (2013) proposes that bidirectional processes all fall into what they call the Weakly Deterministic functions, which are the result of composing a pair of subsequential functions that preclude the possibility of abstract intermediate encoding. I showed in Section 4.4, however, that the subsequential functions may not be a fully accurate model of long-distance phonology, and this Chapter will show that it is sufficient to compose two tier-based functions. I use the term Tier-based Weakly Local (TWL) to describe the class of functions that results from such composition, and demonstrate how they accurately model the various ways in which a process can apply bidirectionally. The Chapter also analyzes a handful of patterns that exhibit what Jardine (2016) calls *unbounded circumambience*, where an input element seeming responds to non-local information in both directions simultaneously. Unbounded circumambience can to varying degrees receive a TWL analysis, and these analyses lead to a discussion of how segmental and tonal phonology may differ from a computational perspective.

5.1 Weak Determinism and Tier-based Weak Locality

Recall from Section 3.3.2 that ATR harmony in Degema applies both progressively and regressively, with the ATR specification of the root dictating the ATR value of prefix and suffix vowels (Archangeli & Pulleyblank, 2007; Kari, 2007). The relevant data from earlier are repeated in (61). Though it should intuitively be a simple process, it is demonstrably not subsequential (let alone TSL or MTSL). If we try to describe it with a progressive subsequential function that reads from left to right, we will not always correctly handle prefix vowels since we can only delay output decisions a finite number of steps and a prefix vowel may be arbitrarily far from the closest root vowel. A mirror argument holds if we try to model the pattern as a regressive subsequential function reading from right to left, so there is no single subsequential function that can model the process. That being said, each of these attempted subsequential analyses only fails for affix vowels on one side of the root; an appropriate progressive function will always be correct for suffix vowels and an appropriate regressive function will always be correct for prefix vowels. What we could do, then, is apply a progressive function to get the correct suffix vowels and then a regressive function to get the correct prefix vowels (or vice versa).

(61) Degema ATR harmony (Kari, 2007)

a.	[ɓól]	'hold'	[u-ɓóˡl-ɔ́m]	'holding'
b.	[gɛ́n]	'look'	[ʊ-gɛ́ˡn-ám]	'looking'
c.	[ɗúm]	'create'	[o-ɗúˡm-ɔ́m]	'creator'
d.	[hɔ́r]	'sharpen'	[ɔ-hɔ́ˡr-ám]	'sharpener'

Achieving a desired result by combining two (or more) functions like this is known as *functional composition*. More formally, given a function f and a function g, their composition $f \circ g$ is the function h that you would get by applying g to the output of f. Essentially, the composition is a single function that performs the work of both functions in a single step. Instead of having f produce an intermediate form that g then transforms into the output, h goes directly from the input string to the output string. Care must be taken when modelling phonological processes through composition, since we run the risk of overgeneration if we do not place sufficient restrictions on the pairs of functions that can be composed. For instance, Elgot & Mezei (1965) proved that with the right "intermediate" alphabet, any regular function h can be described as the composition of a progressive subsequential function f with a regressive subsequential function g. They showed that introducing special characters into the intermediate alphabet lets the first function annotate the input string, giving the second function information that it otherwise wouldn't have but that is necessary for producing the correct output.

Theorem 5.1 Adapted from Elgot & Mezei (1965)**.**
A function $h : \Sigma^* \to \Delta^*$ is regular if and only if there exists a progressive subsequential function $f : \Sigma^* \to \Gamma^*$ and a regressive subsequential function $g : \Gamma^* \to \Delta^*$ with $\Sigma \subseteq \Gamma$ such that $h = f \circ g$

I showed in Section 4.4 that the subsequential functions can produce pathological patterns, so it follows that a composition of subsequential functions can produce the same pathologies. Nevertheless, it is worth demonstrating how unwanted effects can result from the right choice of intermediate alphabet, to the exclusion of other factors. The particular pattern I will generate in this way is what Wilson (2003, 2006) calls *sour grapes* vowel harmony, adapting a term from Padgett (1995). In such a system, a vowel will only trigger harmony if it can affect all subsequent vowels. Potential harmony triggers in a sour grapes process are greedy and stubborn: they look down the line hoping to see that they will affect all subsequent vowels, but if any subsequent vowel would resist their influence (for whatever reason), they will abstain from triggering harmony at all. For ease of exposition, suppose we have the input alphabet $\Sigma = \{+, -, \times\}$ where $+$ is a potential harmony trigger, $-$ is a potential harmony undergoer, and $\times$ is a harmony resistor. To achieve sour grapes behaviour, we can use an intermediate alphabet that augments the input alphabet with a symbol that represents a tentative application of harmony; let this symbol be $\sim$. With this intermediate alphabet in hand, we first read in the progressive direction, replacing $-$ with $\sim$ if it is preceded by $+$ unless an $\times$ intervenes. Following that, we read in the regressive direction, changing $\sim$ to $+$ if we have not seen $\times$, otherwise we replace $\sim$ with $-$. The first progressive run applies harmony, but in a non-committal fashion by using the special symbol $\sim$; the regressive run then either confirms or vetoes the harmony based on the presence/absence of a resistor.

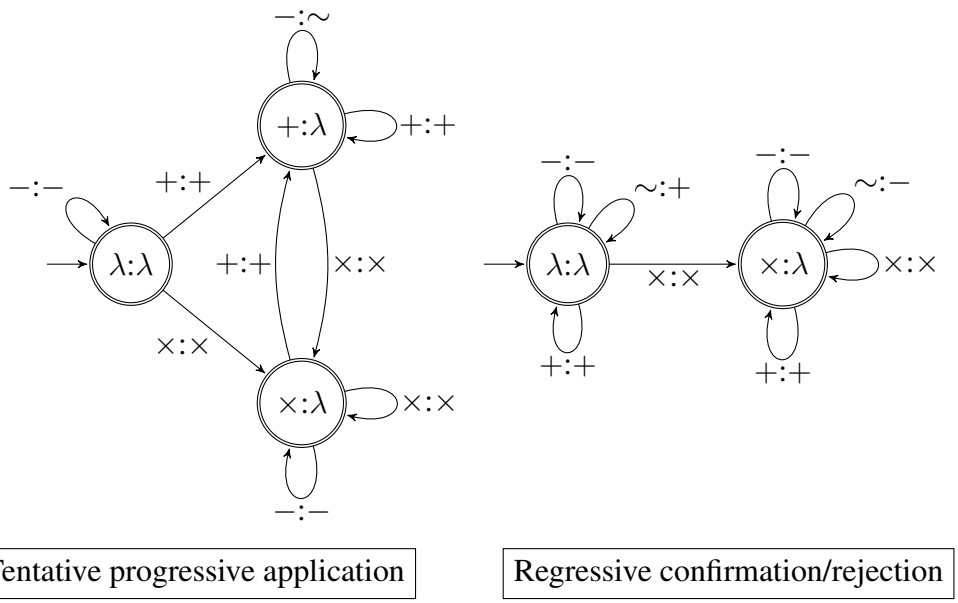


Figure 5.1: Producing sour grapes harmony through intermediate annotation

Sour grapes harmony is a valuable example of the dangers associated with intermediate alphabets since it can be decomposed into two very simple processes when alphabets are unconstrained. In fact, as I have described it above, it is the result of two Tier-based Strictly Local (TSL) functions. Each function is both ITSL_2 and OTSL_2, the choice having no effect on the necessary tier contents. The first needs a tier containing $+$ and $\times$ while the second needs a tier containing only $\times$. Transducers representing these functions are displayed in Figure 5.1. Raising this same sour grapes example, Heinz & Lai (2013) argue that, when modelling bidirectional phonological processes, the first function should be *alphabet preserving*, meaning that its input and output alphabets are the same. They additionally point out that the same kind of annotation can be achieved without any special characters so long as the alphabet has at least 2 elements since we can encode extra information by increasing the length of the input string. This is best seen through an example, so consider the transducers in Figure 5.2 which also compute sour grapes harmony. On the first progressive pass, we write two elements for every input element: $+$ and $\times$ simply get doubled, and $-$ becomes $-+$ if it is preceded by $+$ unless $\times$ intervenes, else it gets doubled. Now, on the second regressive pass, we only write to the output on every other step, in a sense transforming the pairs of elements into single elements. The sequences $++$, $--$, and $\times\times$ get rewritten as $+$, $-$, and $\times$ respectively. More importantly, our tentative application of harmony is encoded as $-+$, and the pair is transformed to $+$ unless we have seen $\times$, in which case it is transformed to $-$. The first function is once again TSL for the tier containing $+$ and $\times$, although the second is properly subsequential since it needs to count in multiples of two (i.e., it is parity-sensitive).

To model bidirectional processes while avoiding the possibility of sour grapes systems (and other pathological patterns that can be derived through intermediate annotation or encoding), Heinz & Lai (2013) define what they call the Weakly Deterministic (WD) functions. These are functions that can be decomposed into a progressive subsequential function and a regressive subsequential function such that the first is alphabet preserving and does not increase the length of the string. Heinz & Lai (2013) then go on to show that root-controlled vowel harmony systems and dominant-recessive vowel harmony systems are WD. Root control and dominance will be described in Sections 5.2.1 and 5.2.2 respectively. I will go one step further and show in the next Section that these behaviours can be decomposed into a pair of (M)TSL functions, thus circumventing aspects of the subsequential functions that I identified in Section 4.4 as problematic. To draw attention to the parallels between my new class and the class defined by Heinz & Lai (2013), I will call these the Tier-based *Weakly* Local (TWL) functions.

Definition 5.1 Weakly Deterministic function (adapted from Heinz & Lai 2013)**.**
A function $h : \Sigma^* \to \Delta^*$ is Weakly Deterministic if and only if there is a pair of opposite-direction subsequential functions $f : \Sigma^* \to \Sigma^*$ and $g : \Sigma^* \to \Delta^*$ such that $|f(w)| \leq |w|$ and $h = f \circ g$

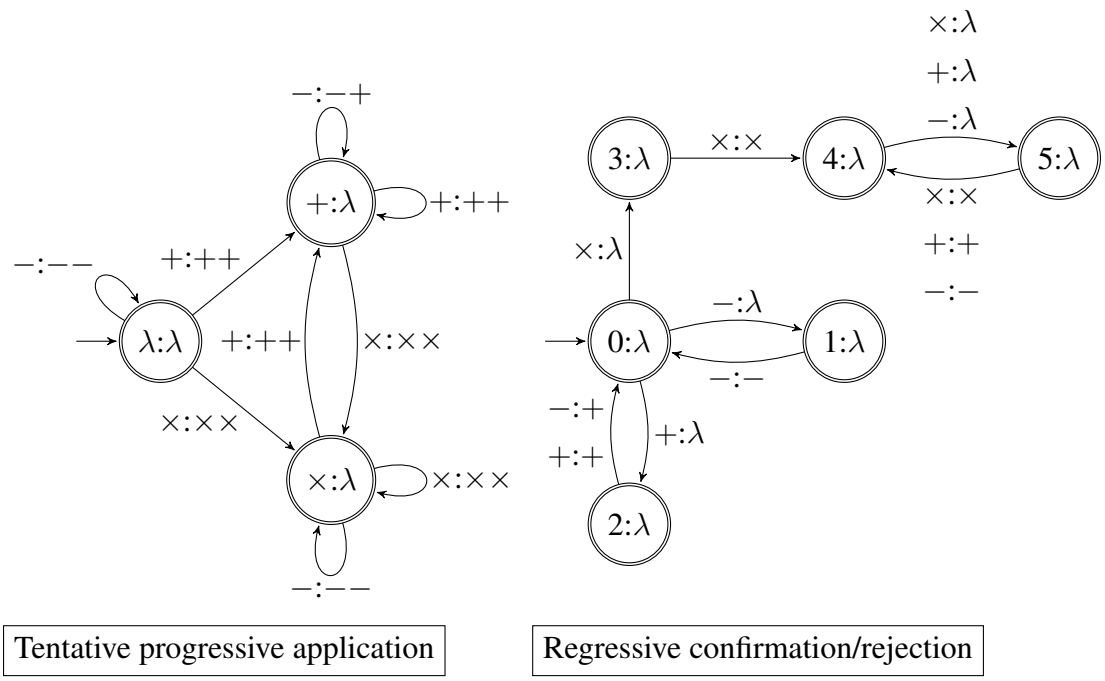


Figure 5.2: Producing sour grapes harmony through intermediate encoding

Definition 5.2 Tier-based Weakly Local function**.**
A function $h : \Sigma^* \rightarrow \Delta^*$ is Tier-based Weakly Local if and only if there is a pair of opposite-direction MTSL functions $f : \Sigma^* \rightarrow \Sigma^*$ and $g : \Sigma^* \rightarrow \Delta^*$ such that $|f(w)| \leq |w|$ and $h = f \circ g$

The above definition of TWL functions includes the requirement that the first function not increase the length of the string, but I note that this may not be necessary. From my understanding, encoding is only a viable strategy when the second function can count in multiples since for the encoding to work properly there must be some number n such that the first function writes n elements at each step and such that the second function only writes to the output every n steps. Keeping track of multiples is a properly subsequential ability and is impossible to accomplish by looking at tier suffixes alone. Furthermore, the above definitions depart slightly from that of Heinz & Lai (2013) in that the first function may be progressive or regressive, so long as the second function has the opposite directionality. I make this departure because some processes analyzed below and in Appendix A can only be correctly modelled when the first function is regressive. Payne (2017) and McCollum et al. (2020) also make this departure, but Meinhardt et al. (2020) argue for an alternative definition of Weak Determinism wherein the pair of sub-functions must be able to occur in either order. Their definition is designed specifically to exclude functions that must be decomposed into an *interacting* pair of opposite-direction subsequential functions. They do so to address the fact that, as we will see, Heinz and Lai's definition does not exclude all conceivable methods for smuggling information into an intermediate form. This alternative *interaction-free*

version of Weak Determinism is still under development (the current definition considers only *synchronous* transducers, which produce 0 or 1 output elements on each step) so I do not present it in any detail here. I will, however, mention below whether each of the analyzed patterns meet this extra criterion of zero interaction.

5.2 Capabilities of Tier-based Weak Locality

Having motivated and defined Tier-based Weak Locality, I will now describe in more detail three behaviours that are readily captured by a TWL function. The first two have already been identified in the literature as being WD, and so part of this Section's contribution is to show that they are even less complex than has previously been reported. I additionally discuss one behaviour not yet explored in the literature on Weak Determinism, which I call direction sensitivity. In order, I analyze root control (Section 5.2.1), dominant-recessive systems (Section 5.2.2), and direction sensitivity (Section 5.2.3).

5.2.1 Root control

Degema ATR harmony, which I used to motivate the TWL functions, is a classic example of root control. Within such a system, the [$\pm$ATR] specification of all vowels in a word is not determined by the leftmost or rightmost vowel, but rather the [$\pm$ATR] specification on the vowels in the root. When describing such harmony as a TWL function, the ITSL or OTSL status of either component function does not matter, nor does it matter whether the first function is progressive or regressive. As an aside, this latter fact that the order of the two component functions does not matter makes the overall function meet the "no interaction" requirement of the definition for Weak Determinism set out by Meinhardt et al. (2020). For illustrative purposes, though, I will describe it using a progressive ITSL_2 function followed by a regressive OTSL_2 function. The transducers for these functions are shown side by side in Figure 5.3, where $+$ represents a [$+$ATR] vowel, $-$ represents a [$-$ATR] vowel, $\emptyset$ represents a vowel unspecified for [$\pm$ATR], and C represents a consonant.

The first transducer will maintain the unspecified status of vowels until a specified vowel is encountered, upon which it moves to one of the non-initial states and adds a specification to any subsequent unspecified vowel. Assuming that all and only the root vowels are specified for [$\pm$ATR] in underlying forms, this transduction will give just the suffix vowels a value for [$\pm$ATR], and this value will match that of the rightmost vowel in the root. The second transducer works identically, giving just the prefix vowels an [$\pm$ATR] value that matches that of the leftmost vowel in the root. Running both transducers ensures that all unspecified affix vowels receive a value for [$\pm$ATR], assuming of course that a root exists in the first place. While I am not aware of any disharmonic roots

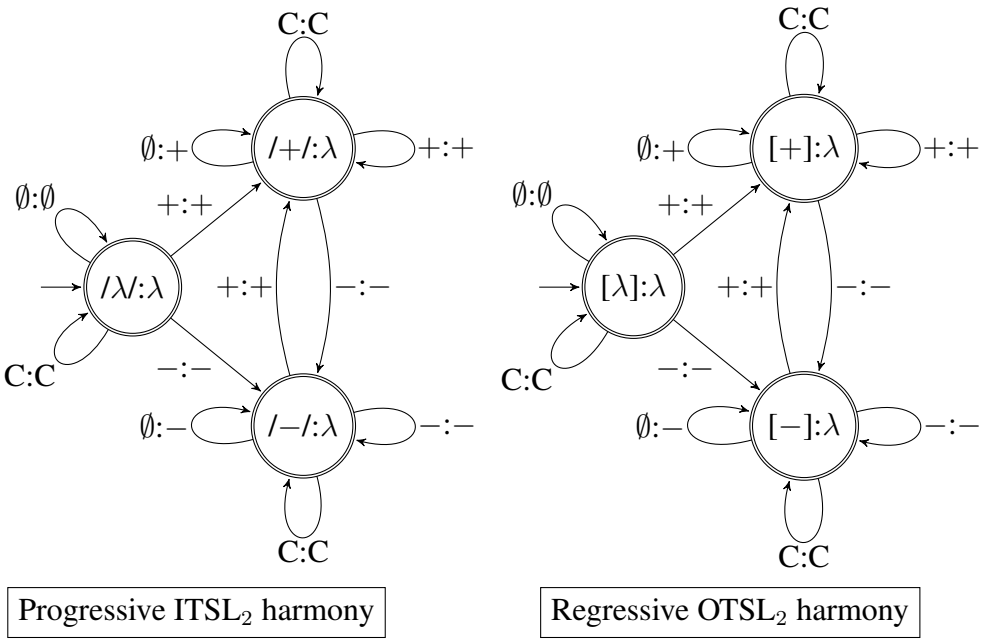


Figure 5.3: Root-controlled ATR harmony as a TWL function

in Degema, the analysis here considers them as a possibility by ensuring that pre-specified vowels always surface faithfully. It is also worth pointing out that, even though the analysis presents the two component functions as distinct entities, they are the same function aside from directionality.

5.2.2 Dominant-recessive systems

A phenomenon very similar to root control is exhibited by the ATR vowel harmony in Diola Fogny. As in Degema, harmony progresses outwards in both directions from a source, although rather than roots acting as a source, instances of [+ATR] fill this role. When there is a [+ATR] vowel anywhere in the underlying form (inside or outside of the root) then all vowels will be [+ATR] on the surface, but if no [+ATR] vowel exists in the underlying form, then all vowels surface as [−ATR] by default (Sapir, 1965; Ringen, 1975, 1979; Baković, 2000; Walker, 2012). Example (62a) shows a [+ATR] root as the harmony source, example (62b) shows the suffix /-ul/ ‘towards speaker’ as the harmony source, and example (62c) shows default [−ATR] in the absence of any [+ATR] source. Systems like this where one value of a feature will spread everywhere (if present) are called dominant recessive. The value that spreads regardless of its position is called the dominant value, and the other default value is called the recessive value.

(62) Diola Fogny ATR harmony (Walker, 2012)

a. /nɪ-jitum-ɛn-ʊ/ [ni-jitum-en-u] 1SG-lead.away-CAUS-2PL
b. /nɪ-baj-ul-ʊ/ [ni-bəj-ul-u] 1SG-have-towards.speaker--2PL
c. /nɪ-baj-ɛn-ʊ/ [nɪ-baj-ɛn-ʊ] 1SG-have-CAUS-2PL

When we go to model dominant-recessive harmony as a TWL function, it does not matter whether the first functions is progressive or regressive, nor does the ITSL or OTSL status of either function matter. Once again, we are meeting Meinhardt et al.'s (2020) requirement of no interaction, in addition to Heinz & Lai's (2013) requirements of an alphabet- and length-preserving first function. All that is necessary is for the tier of each function to contain only [+ATR] vowels. For concreteness, and to highlight the similarity to root control, I will describe dominant-recessive harmony here using a progressive ITSL_2 function followed by a regressive OTSL_2 function. The transducers for these functions are shown side by side in Figure 5.4, where + represents a [+ATR] vowel, − represents a [−ATR] vowel, ∅ represents a vowel unspecified for [±ATR], and C represents a consonant.

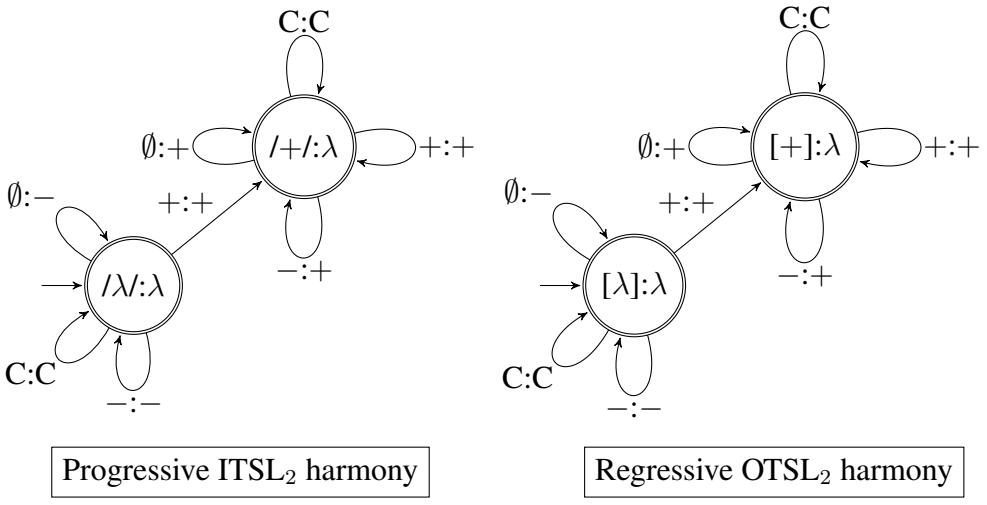


Figure 5.4: Dominant-recessive ATR harmony as a TWL function

The first transducer will faithfully reproduce [−ATR] vowels and rewrite unspecified vowels as their [−ATR] counterparts until a [+ATR] vowel is encountered, upon which it moves to the non-initial state and ensures that every subsequent vowel stays or becomes [+ATR]. This transduction will make any vowels to the right of a [+ATR] vowel either stay or become [+ATR], accomplishing half of the desired outcome. The second transducer works identically, but makes any vowel to the left of a [+ATR] vowel either stay or become [+ATR], accomplishing the other half of the desired outcome. In the absence of any underlying [+ATR] vowel, all [−ATR] vowels will have stayed [−ATR] and all unspecified vowels will have become [−ATR] after the first transduction; the second transducer will do nothing to change this default result. Once again, even though the

analysis presents the two component functions as distinct entities, they are the same function aside from directionality.

5.2.3 Direction sensitivity

Coincidentally, vowel harmony in Degema and Diola Fogny could both be analyzed as a single process applying in one direction and then the other, but we can and should expect the two directions to differ. One possible way that the progressive and regressive halves of a bi-directional process can differ involves blocking/transparency. Take for instance the process of emphasis spreading in Southern Palestinian Arabic. A consonant that is underlyingly pharyngealized will cause both preceding and following segments to become pharyngealized, possibly spreading the pharyngealization all the way to the word edge (Davis, 1995; Jurgec, 2011; Rose & Walker, 2011). Importantly, progressive spreading is blocked by the palatal consonants /j/ and /ʃ/ and the high front vowels /i/ and /y/ as shown in (63), whereas regressive spreading is unrestricted as shown in (64). For typographical convenience, consonants with secondary pharyngealization are marked in the data with underlining.

(63) Unrestricted regressive emphasis spreading in Southern Palestinian Arabic (Davis, 1995, pp. 473-474)

a.	/ballaas̲/	[ballaas]	‘thief’
b.	/naʃaat̲/	[naʃaat]	‘energy’
c.	/tamʃiit̲a/	[tamʃiita]	‘hair styling’
d.	/xayyaat̲/	[xayyaat]	‘tailor’

(64) Progressive emphasis spreading in Southern Palestinian Arabic blocked by front segments (Davis, 1995, pp. 473-474)

a.	/t̲eef-ak/	[teef-ak]	‘your sword’
b.	/ʕat̲ʃaan/	[ʕat̲ʃaan]	‘thirsty’
c.	/t̲iin-ak/	[t̲iin-ak]	‘your mind’
d.	/majas̲s̲as̲iʃ/	[majassasiʃ]	‘it did not become solid’
e.	/s̲ayyaad/	[sayyaad]	‘hunter’
f.	/d̲ajjat/	[dajjaat]	‘types of noise’

There is no reason to assume in this case that progressive spreading occurs before regressive spreading or vice versa, so I will arbitrarily have the regressive process apply first. Both processes can be analyzed as ITSL functions, although OSL analyses are more intuitive since each is a typical instance of local iterative spreading. A small caveat is necessary for this analysis: neither function is fully alphabet preserving, since they can both produce non-phonemic segments (e.g., pharyngealized vowels). That being said, neither process is length-increasing and neither introduces

abstract "intermediate" symbols, so the analysis fits in the spirit (if not the letter) of weak determinism and tier-based weak locality. Also, despite slightly breaking the requirements of Heinz & Lai's (2013) definition of Weak Determinism, the two component functions nonetheless do not interact and so fulfill the definition argued for by Meinhardt et al. (2020). Transducers for the two functions are shown side by side in Figure 5.5, where lowercase 'x' represents a non-pharyngeal non-front segment, upper case 'X' represents a pharyngeal non-front segment, lowercase 'f' represents a non-pharyngeal front segment, and upper case 'F' represents a pharyngeal front segment. Certain states are collapsed into single circles to reduce visual clutter.

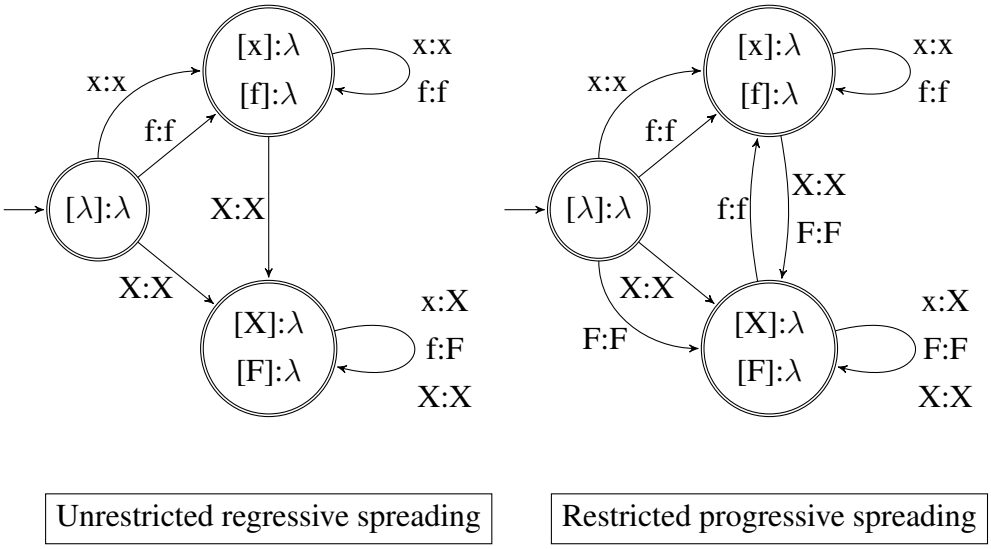


Figure 5.5: Direction-sensitive blocking in Southern Palestinian Arabic

The first regressive transducer faithfully reproduces non-pharyngeal segments until a pharyngeal segment is encountered, after which it will produce only pharyngeal segments. This transduction will make all segments to the left of a pharyngeal segment either stay or become pharyngeal, accomplishing half of the desired outcome. The second progressive transducer operates nearly identically, with the exception that non-pharyngeal front segments are always produced faithfully and push the machine into a non-pharyngeal state. As a result, pharyngealization spreads rightward, affecting all segments until a blocker is encountered. Blocking is not always limited to the progressive direction; Maasai ATR harmony presents a mirror-image of the Southern Palestinian Arabic pattern, with unrestricted progressive spreading and restricted regressive spreading. More specifically, the low vowel /a/ blocks regressive harmony, but undergoes and transmits progressive harmony (Tucker & Mpaayei, 1955; Archangeli & Pulleyblank, 2007; Rose & Walker, 2011). Since an analysis of the Maasai pattern would not differ substantially from my analysis of the Southern Palestinian Arabic pattern aside from the reversal in the directionality of blocking, I do not analyze Maasai harmony in any detail here.

Differences in iterativity are another major way in which a process can be direction sensitive. As an example, consider the dominant-recessive ATR vowel harmony in Nawuri. In the regressive direction, an instance of [+ATR] will spread and affect all non-low vowels, with the low vowel /a/ being transparent (Casali, 2002; Rose & Walker, 2011), as shown in (65a-b). In the progressive direction, however, an instance of [+ATR] will spread only to the adjacent vowel, and only if that vowel is high (Casali, 2002; Rose & Walker, 2011), as shown in (65c-d). I abstract away from the processes of vowel coalescence and front vowel centralization (see Casali, 2002), as these would not significantly alter the analysis below aside from requiring a composition of MTSL functions instead of just TSL functions in order to address the additional details.

(65) Asymmetrical iterativity in Nawuri ATR harmony (Casali, 2002, p. 25)

a.	/ɪ-sɪ ɪ-bʊ ɔ-bu-to/ NC-sand INCMPL-be NC-room-in	[isiiboobuto]	‘sand is in the room’
b.	/ɛ-kɔɔlɪ a-fulee/ PROG-he.receive NC-money	[ekoolaafulee]	‘he is collecting money’
c.	/a-fuu fʊʊtɪ-sa/ NC-air breathe-ADJ	[afuufuutɪsa]	‘air for breathing’
d.	/gɪ-buu tʊʊ-sa/ NC-stone throw-ADJ	[gibuutuusa]	‘a stone for throwing’

Once again, there is no reason to assume in this case that progressive harmony occurs before regressive harmony or vice versa, so I will arbitrarily have the regressive process apply first. That being said, while the regressive function can be either ITSL or OTSL, the progressive function absolutely must be ITSL in order to derive its non-iterative nature. Transducers for the two functions are shown side by side in Figure 5.6, where lowercase ‘h’ represents a [−ATR] high vowel, upper case ‘H’ represents a [+ATR] high vowel, lowercase ‘m’ represents a [−ATR] mid vowel, upper case ‘M’ represents a [+ATR] mid vowel, and ‘C’ represents a consonant. The tier for the first function includes just the [+ATR] vowels, and the tier for the second function includes all vowels. Certain states are collapsed into single circles to reduce visual clutter.

The first regressive transducer doesn’t alter any input segments until a [+ATR] vowel is encountered, after which all non-low vowels will be output as their [+ATR] counterpart, regardless of their input specification. This transduction will make all non-low vowels to the left of a [+ATR] vowel either stay or become [+ATR], accomplishing half of the desired outcome. The second progressive transducer, for its part, makes almost no changes to its input, only altering an input high vowel if the immediately preceding vowel was [+ATR]. As a result, an instance of [+ATR] will spread rightward, but only by one step, and only if the target vowel is high. In Nawuri bi-directional harmony, it is the progressive half that is non-iterative; the opposite state of affairs can be found

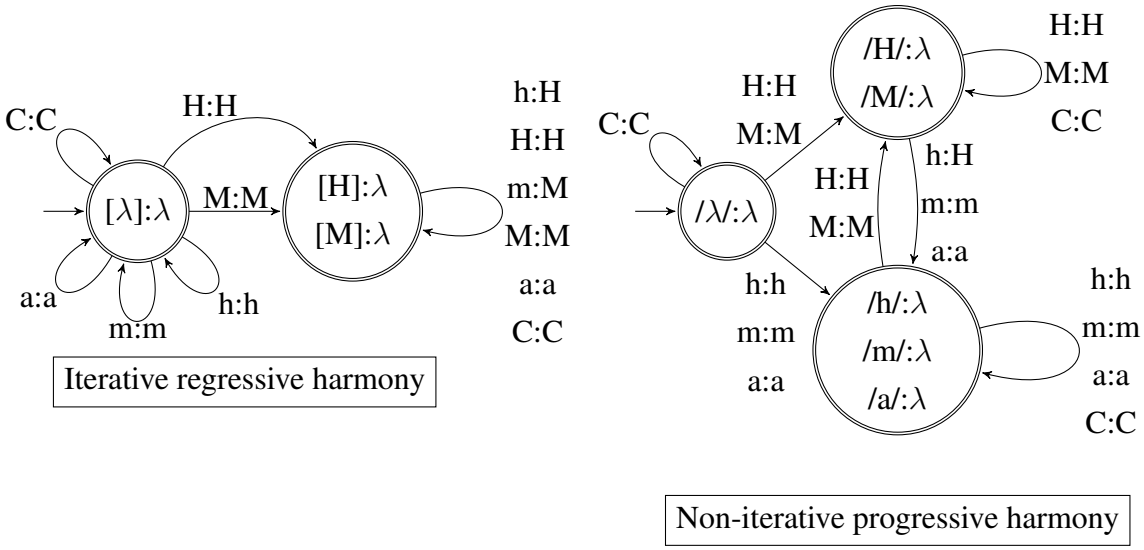


Figure 5.6: Direction-sensitive iterativity in Nawuri

in Epena Pedee. A nasal vowel in Epena Pedee will spread its nasality in both directions, but to differing degrees. Progressively, the spreading will continue until an obstruent is encountered, whereas regressively, only the immediately preceding syllable onset will be affected (Harms, 1985; Jurgec, 2011; Rose & Walker, 2011). Since an analysis of the Epena Pedee pattern would not differ substantially from my analysis of the Nawuri pattern aside from swapping the (non-)iterativity of the progressive and regressive halves, I do not analyze Epena Pedee nasal spread in any detail here.

5.3 Unbounded circumambience

Jardine (2016) investigated what he calls *unbounded circumambience*, where the application of a process depends on potentially non-local information in both directions simultaneously, and argued that only tonal phenomena can exhibit this property. Jardine hypothesized that unbounded circumambient processes were not WD and therefore concluded that the WD barrier separates segmental phenomena from tonal phenomena. Since the publication of his article, however, it has been found that many unbounded circumambient processes meet Heinz & Lai's (2013) definition of Weak Determinism (O'Hara & Smith, 2019; Smith & O'Hara, 2019; Lamont et al., 2019), and a growing number of unbounded circumambient segmental phenomena are being discovered (McCollum et al., 2020). In the sections that follow, I will consider some of the segmental phenomena reported by Jardine (2016) and McCollum et al. (2020) to exhibit unbounded circumambience, and show that they can (to varying degrees) be analyzed as TWL functions. The remainder of their reported instances of unbounded circumambience are also shown to be TWL in Appendix A. Importantly, these analyses rely on a specific ordering of the component functions, meaning that while they

meet the definition of Weak Determinism put forth by Heinz & Lai (2013), they crucially rely on process interaction and therefore disobey the definition of Weak Determinism put forth by Meinhardt et al. (2020). I divide the processes into two types based on how they achieve unbounded circumambience. First are the instances where two opposite-direction sub-processes conflict with each other (Section 5.3.1). Second are the instances where a first sub-process can provide crucial information to the second opposite-direction process without needing to use abstract intermediate symbols or increasing the length of the string (Section 5.3.2). Finally, Section 5.3.3 analyzes sour-grapes-like tonal spreading in Copperbelt Bemba and shows that while it is WD, it is not TWL. The section also discusses how segmental and tonal phonology may differ computationally, the former being entirely tier-based and the latter not.

5.3.1 Conflicting processes

A feature common to all the bidirectional processes analyzed thus far is that their two component processes have similar goals with similar outcomes. What if, however, the two processes had different and conflicting goals with mutually exclusive outcomes? The rule that applies second in such a scenario could undo the work of the first rule, or be blocked by effects of the first rule, possibly creating the impression that an input segment is simultaneously responding to non-local material in both directions. Here I will analyze the ATR harmony in Turkana and Karimojong as an instance of process antagonism. Other languages with unbounded circumambience that can be analyzed in a similar way include Sanskrit (Hansson, 2010; Jardine, 2016; Ryan, 2017; Graf & Mayer, 2018), Liko (McCollum et al., 2020), and Yidiɳ (Dixon, 1977; Crowhurst & Hewitt, 1995; Payne, 2017). Analyses for those additional languages are presented in Appendix A.

Turkana and Karimojong have a system of ATR vowel harmony where [+ATR] is the dominant value, although they depart from typical dominant-recessive systems in that each contains a small number of dominant [−ATR] suffixes. Unbounded circumambience arises when the low vowel /a/ receives conflicting demands from a preceding [+ATR] dominant vowel and a following [−ATR] dominant vowel, since either conditioning vowel can be an arbitrary distance away. The Turkana and Karimojong patterns are essentially the same so I will only present the Karimojong data. In Karimojong, the low vowel /ɑ/ remains [ɑ] in the absence of any preceding dominant [+ATR] morpheme, and normally becomes [o] when preceded by a dominant [+ATR] morpheme, but becomes [ɔ] if also followed by a [−ATR] dominant suffix (Novelli, 1985; Lesley-Neuman, 2012; McCollum et al., 2020).[1] Each of these tendencies is illustrated in (66), where the dominant [+ATR] morphemes are shown in bold and the dominant [−ATR] suffix is shown in italics. Regressive

[1]Sources for the nearly identical Turkana pattern include Dimmendaal (1983), Noske (1996, 2000), Baković (2000), and McCollum et al. (2020).

spreading of [+ATR] does not affect /ɑ/ and is blocked by /ɑ/ (Novelli, 1985; Lesley-Neuman, 2012; McCollum et al., 2020), but this is not crucial to the analysis.

(66) Conflicting dominance in Karimojong ATR harmony (McCollum et al., 2020, p. 244)

a.	/ɑkɪ-tɔ-dɔŋ-ɑn-ɑkin/	[ɑkɪ-tɔ-dɔŋ-ɑn-ɑkin]
	INF-CAUS-handle.firmly-FREQ-DAT	
b.	/ɑkɪ-**zi-don**-ɑn-ɑkin/	[ɑki-**zi-doŋ**-on-okin]
	INF-CAUS-castrate-FREQ-DAT	
c.	/ɑkɪ-**doŋ**-ɑn-*ɔr*/	[ɑkɪ-**dɔŋ**-ɔn-*ɔr*]
	INF-castrate-FREQ-ITV	

Notice especially how the dominant [−ATR] suffix /-*ɔr*/ causes the dominant [+ATR] root **/doŋ/** to become [−ATR] in (66c). Based on this observation, it would seem that progressive harmony applies first, failing to override the [−ATR] specification of the dominant suffix. After that, regressive harmony applies from the dominant [−ATR] suffix to override all values of [+ATR] that come before it. Underlying /a/ thus becomes [ɔ] in two steps: it first gets transformed into [o] by progressive [+ATR] harmony, and this [o] then gets transformed to [ɔ] by regressive [−ATR] harmony. For example, where all changes are underlined, the input /ɑkɪ-**doŋ**-ɑn-*ɔr*/ from (66c) first gets mapped to [ɑkɪ-**doŋ**-o̲n-*ɔr*] through progressive harmony and then this intermediate stage gets mapped to the actual output [ɑkɪ-**dɔ̲ŋ**-ɔ̲n-*ɔr*] through regressive harmony. Compare this to the input /ɑkɪ-**zi-don**-ɑn-ɑkin/ from (66b) which first gets mapped to [ɑkɪ-**zi-doŋ**-o̲n-o̲kin] through progressive harmony and then to [ɑki̲-**zi-doŋ**-on-okin] through regressive harmony. Each of these sub-processes is straightforwardly analyzed as an OTSL_2 function if dominant [−ATR] vowels are represented separately from recessive [−ATR] vowels in the input and output alphabets. Progressive harmony would have a tier consisting of [+ATR] vowels and dominant (but not recessive) [−ATR] vowels, to which regressive harmony adds [ɑ] (turning it into a blocker). Importantly, both progressive and regressive harmony are alphabet preserving when analyzed in this way, and neither process is length increasing, so the Karimojong/Turkana pattern fulfills all the requirements of Tier-based Weak Locality.

5.3.2 Complementary processes

The previous section showed how conflicts between two opposite-direction (M)TSL patterns can cause unbounded circumambience. Alternatively, the two processes can work together, such that the first process "informs" the second in some way, without increasing the length of the string *or* using abstract intermediate symbols. Here I will analyze the ATR harmony in Tutrugbu and Tafi as an instance of process complementarity. Another language with unbounded circumambience that can be analyzed in a similar way is Yaka (Jardine, 2016), which is analyzed in Appendix A.

Tutrugbu vowel harmony is the main focus of McCollum et al. (2020), and the closely related language Tafi has a nearly identical pattern (Bobuafor, 2013; McCollum et al., 2020). Root vowels that are [±ATR] spread this feature-value leftwards to prefix vowels, and when all prefix vowels are of the same height, the harmony will reach the left word edge (McCollum & Essegbey, 2018; McCollum et al., 2020), as seen in (67a) for a string of [+high] prefixes and (67b) for a string of [−high] prefixes. Roots are marked in bold for clarity. Strings of prefixes with different heights are more complex and their behaviour depends on the height of the leftmost vowel. Harmony spreads to all prefix vowels when the leftmost vowel is [−high] as seen in (67c), but harmony is blocked by [−high] vowels when the leftmost vowel is [+high] as seen in (67d), although harmony still spreads up until that [−high] blocker (McCollum & Essegbey, 2018; McCollum et al., 2020). The outcome of an input [−high] vowel can therefore depend simultaneously on non-local information to its left (the height of the leftmost vowel) and non-local information to its right (the ATR specification of the root), creating clear instances of unbounded circumambience.

(67) Conditional blocking by low vowels in Tutrugbu ATR harmony (McCollum et al., 2020)

a.	/bʊ-tí-**ʃē**/	[bu-t**ɪ́**-**ʃē**]	‘1PL-NEG-grow’
b.	/ka-ba-**ʃē**/	[ke-be-**ʃē**]	‘2SG-FUT-grow’
c.	/ka-tí-ba-**ʃē**/	[ke-t**ɪ́**-be-**ʃē**]	‘CL7-NEG-FUT-grow’
d.	/ɪ-ba-dɪ-**wu**/	[ɪ-ba-di-**wu**]	‘1SG-FUT-ITV-climb’

An important observation is that whether the initial vowel becomes [+ATR] can be determined on a single regressive pass through the string. Furthermore, if the leftmost prefix vowel becomes [+ATR], then all prefix vowels become [+ATR]. What we can do, then, is have a first regressive function determine whether harmony affects the leftmost vowel, and then have a second progressive function spread harmony from the leftmost vowel up until the root. Section 4.3.5 showed that IMSTL functions can handle word-edge targets. For the regressive half of Tutrugbu harmony, we need an IMTSL$_3$ function since determining the word-initial status of a vowel requires us to wait up to two steps upon reading it. The tiers that this regressive functions tracks are T_1 containing all input elements, T_2 containing [+ATR] vowels and [−high] vowels, and T_3 containing just [+ATR] vowels. The progressive function is easier and can be accomplished by an ITSL$_2$ or OTSL$_2$ function that has a tier consisting of [+ATR] vowels.[2] A sample two-part derivation with blocking is shown in Table 5.1 and a sample two-part derivation with no blocking is shown in Table 5.2. Both derivations are described more fully in the paragraphs that follow.

Step 3 of the first derivation enforces [+ATR] harmony since we are reading a [+high] vowel after a [+ATR] vowel with no intervening [−high] vowel; vowels in this environment always

[2]Suffixes are rare in Tutrugbu, and do not undergo harmony (McCollum & Essegbey, 2018; McCollum et al., 2020), which is easily accounted for by having a root boundary symbol on the tier.

Step	Window 1 (all elements)	Window 2 ([+ATR] Vs and [−high] Vs)	Window 3 ([+ATR] Vs)	Input	Output
1	/ /	/ /	/ /	/u/	[u]
2	/u/	/u/	/u/	/w/	[w]
3	/uw/	/u/	/u/	/ɪ/	[i]
4	/wɪ/	/u/	/u/	/d/	[d]
5	/ɪd/	/u/	/u/	/a/	[]
6	/da/	/ua/	/u/	/b/	[]
7	/ab/	/ua/	/u/	/ɪ/	[abɪ]
8	/bɪ/	/ua/	/u/	/⋉/	[]
Step	Window ([+ATR] Vs)			Input	Output
1	/ /			/ɪ/	[ɪ]
2	/ /			/b/	[b]
3	/ /			/a/	[a]
4	/ /			/d/	[d]
5	/ /			/i/	[i]
6	/i/			/w/	[w]
7	/i/			/u/	[u]

Table 5.1: Initial [+high] vowel unaffected by regressive [+ATR] harmony.

become [+ATR] regardless of the height of the leftmost vowel. On step 5 we encounter our first instance of /a/ since encountering the [+ATR] vowel /u/. We do not know whether this /a/ is the word's leftmost vowel so we output nothing. The next step also outputs nothing in case the /b/ being read is not word-initial. Reading /ɪ/ on the next step tells us that the /a/ from two steps earlier was not the leftmost vowel. We do not yet know whether /a/ and /ɪ/ will become [+ATR] but we can output them as [a] and [ɪ] on this step (along with the /b/) and let the second progressive function decide their ultimate output fate. Coincidentally, the next symbol to be read is ⋉ and so that instance of /ɪ/ ended up being word-initial. Progressive harmony now takes over, but the first two vowels are output as [−ATR] since the first [+ATR] vowel is in the third syllable. Upon completion we have derived [ɪ-ba-di-wu] '1SG-FUT-ITV-climb' from /ɪ-ba-dɪ-wu/, as desired.

In the second derivation, our first instance of /a/ after a [+ATR] vowel is found on step 3. We do not know whether this /a/ is the word's leftmost vowel so we output nothing. The next step also outputs nothing in case the /b/ being read is not word-initial. Reading /ɪ/ on the next step tells us

Step	Window 1 (all elements)	Window 2 ([+ATR] Vs and [−high] Vs)	Window 3 ([+ATR] Vs)	Input	Output
1	/ /	/ /	/ /	/ē/	[ē]
2	/ē/	/ē/	/ē/	/ʃ/	[ʃ]
3	/ēʃ/	/ē/	/ē/	/a/	[]
4	/ʃa/	/ēa/	/ē/	/b/	[]
5	/ab/	/ēa/	/ē/	/ɪ/	[abɪ]
6	/bɪ/	/ēa/	/ē/	/t/	[t]
7	/ɪt/	/ēa/	/ē/	/a/	[]
8	/ta/	/aa/	/ē/	/k/	[]
9	/ak/	/aa/	/ē/	/⋉/	[ek]
Step	Window ([+ATR] Vs)			Input	Output
1	/ /			/k/	[k]
2	/ /			/e/	[e]
3	/e/			/t/	[t]
4	/e/			/ɪ/	[i]
5	/e/			/b/	[b]
6	/e/			/a/	[e]
7	/e/			/ʃ/	[ʃ]
8	/e/			/ē/	[ē]

Table 5.2: Initial [−high] vowel affected by regressive [+ATR] harmony.

that the /a/ from two steps earlier was not the leftmost vowel. We do not yet know whether /a/ and /ɪ/ will become [+ATR] but we can output them as [a] and [ɪ] on this step (along with the /b/) and let the second progressive function decide their ultimate output fate. Another instance of /a/ is found on step 7, where we output nothing. Step 8 also outputs nothing since the word-initial or word-medial status of the /k/ is not yet known. We read ⋉ on step 9, so the /a/ encountered two steps earleir is the leftmost vowel and becomes [e] alongside the word-initial [k]. Progressive harmony now takes over, and since the leftmost vowel is [+ATR], all subsequent vowels become [+ATR]. Upon completion we have derived [ke-tɪ́-be-ʃē] 'CL7-NEG-FUT-grow' from /ka-tí-ba-ʃē/, as desired.

Broadly speaking, the regressive function passes a root's [+ATR] value onto all vowels that it can be *completely* sure will become [+ATR]. Those vowels are [−high] prefix vowels in the

word-initial syllable and [+high] prefix vowels between the root and the rightmost [−high] prefix vowel. The first function essentially tells the second function where the [+ATR] span begins, and the second function ensures that any prefix vowels after that starting point are [+ATR] in the output. Importantly, both functions are alphabet preserving and neither increases the length of the input, making the overall function TWL.

5.3.3 Dividing tonal and segmental phonology

Jardine (2016) conjectured that tonal phonology could be more complex than segmental phonology, saying that the former but not the latter could handle unbounded circumambience. Jardine (2016) accordingly proposed that segmental phenomena, but not tonal phenomena, are at most weakly deterministic in complexity. The sections above show that unbounded circumambience is actually more common in segmental phenomena than originally thought. The sections above (and Appendix A) equally show that these instances of unbounded circumambience can actually be handled by TWL functions (and therefore also by WD functions). That being said, it may still be true that tonal phenomena are less restricted than segmental phenomena. Whereas all of the unbounded circumambient segmental patterns known to me can receive a TWL analysis, the pattern of sour-grapes-like tonal spreading in Copperbelt Bemba is WD but not TWL.

In Copperbelt Bemba, high tones spread in one of two ways. High tones that are followed by another high tone trigger ternary spreading, affecting just the following two tone-bearing units (Bickmore & Kula, 2013; Kula & Bickmore, 2015; Jardine, 2016; O'Hara & Smith, 2019; Smith & O'Hara, 2019). A high tone that is not followed by another high tone (i.e., the rightmost high tone) instead triggers unbounded spreading, affecting all tone-bearing units between itself and the end of the word (Bickmore & Kula, 2013; Kula & Bickmore, 2015; Jardine, 2016; O'Hara & Smith, 2019; Smith & O'Hara, 2019). The mapping in (68a) shows unbounded spreading from a lone word-medial high tone. The mappings in (68b-c) show ternary spreading from a word-medial high tone that is followed by another high tone; the rightmost high tone in (b) is two tone-bearing units down the line so we get what looks like a plateau[3], whereas the rightmost high tone in (c) is five tone-bearing units down the line so we switch to low tones after two steps of spreading (68c).

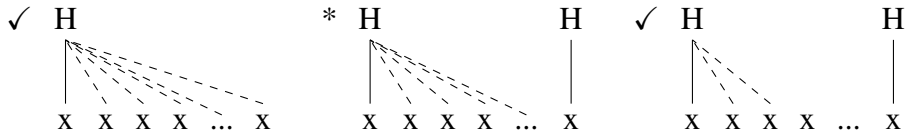
(68) Sour-grapes-like spreading in Copperbelt Bemba (Kula & Bickmore, 2015)

a.	/tu-ka-páapaatik-a/	[tù-kà-páápáátík-á]	'we flatten'
b.	/ú-ku-léet-il-a/	[ú-kú-$^{\downarrow}$léét-él-á]	'to bring for'
c.	/tu-ka-béleeŋg-el-an-a-kó/	[tù-kà-bélééŋg-él-àn-à-kó]	'we'll read for each other'

[3]The rightmost tone gets downstepped before spreading in such cases(Kula & Bickmore, 2015), a detail which the following discussion abstracts away from.

The Copperbelt Bemba pattern is described as "sour grapes like" since all high tones would like to spread progressively to the end of the word, but can only do so if no other high tone follows them at any distance. This is schematized in (69). The first diagram shows a high tone without any following high tone: it is able to spread all the way to the right edge of the word. The second and third diagram show a high tone followed distantly by another: the earlier high tone cannot spread onto all intervening syllables, and is allowed only to spread two steps.

(69) Schematization of sour-grapes-like spreading



O'Hara & Smith (2019) and Smith & O'Hara (2019) show that the Copperbelt Bemba pattern is weakly deterministic thanks to the fact that underlying H always spreads at least two steps to the right. We can take advantage of this *zone of predictability* to smuggle information into the intermediate form without the need for an abstract symbol or the need to lengthen the string. Going first in the regressive direction, we strategically postpone the output of underlying L by up to two steps to ensure that the first underlying H we encounter (i.e. the rightmost underlying H) is followed by the sequence HLL in the intermediate form (unless it is fewer than four spaces from the end). This regressive function then makes sure that instances of underlying H to the left of another underlying H are followed by the sequence LH in the intermediate form, again by strategically postponing the output for underlying L by up to two steps. Following that, a progressive function will turn intermediate HLH into HHH, completing any instances of ternary spreading. The second function will also complete the only instance of unbounded spreading by turning intermediate HHLL into HHHH, and then transforming all subsequent instances of intermediate L into H. It does so because only the rightmost underlying H can exist in the intermediate sequence HHLL. For instance, given underlying /LLHLLL/ from (68a) we get intermediate {LLHHLL} and then output [LLHHHH], and given underlying /LLHLLLLH/ from (68b) we get intermediate {LLHLHLLH} and then output [LLHHHLLH]. For transducers computing these two functions and for more example derivations, see O'Hara & Smith (2019).

Crucially, the second function is not MTSL for any value of k, and is instead properly subsequential. This in turn means that the overall process of sour-grapes-like tone spreading is not TWL for any value of k, and is instead properly WD. To correctly apply unbounded spreading, the second function needs to remember whether the two most recent instances of intermediate H were adjacent, but it is impossible using tier suffixes alone to know whether the current position is preceded unboundedly by a specific local configuration. In the case at hand, it is easy to know

whether we are preceded distantly by two instances of intermediate H using the tier $T_1 = \{\text{H}\}$, but the local tier $T_2 = \{\text{L, H}\}$ can only tell us up to a set distance away from the first H whether those two instances of intermediate H were adjacent. Put another way, the second function needs to be able to "freeze" the intermediate configuration HHLL in memory, which is similar to how subsequential functions handle the minimum distance requirement pathology discussed in Section 4.4.2. Judging from this one example, it would seem that we could separate segmental phonology and tonal phonology by stating that the former is entirely tier-based and that the latter has access to subsequential computation in one or both directions. A single example, however, is very far from sufficient for making such a claim with confidence, and so I leave to future research the question of whether tonal and segmental phonology are truly subject to different limitations.

5.4 Chapter summary

Many phonological patterns cannot be described by a single deterministic function running in a single direction, an obstacle very easily overcome by using function composition. Appropriate caution must be taken when resorting to this powerful computational tool, though, or else we run the risk of overgeneration. Previous work argued that in order to approximate the attested typology of phonological processes, the first in a pair of composed functions must be alphabet preserving (i.e., its input and output alphabets are the same) and must not increase the length of the input string. I showed in this Chapter that we can add a third requirement: each component function must be tier based. Even when adhering to these heavy restrictions, it is possible to describe intricate phenomena such as unbounded circumambience. Interestingly, the one clear counterexample from Jardine (2016) to the MTSL requirement is a tonal pattern, suggesting that segmental and tonal phonology might be subject to different computational limitations, although it is premature to make this claim with any confidence, and the differences in complexity may disappear under different representational assumptions.

Chapter 6

Probabilistic cases

The Strictly Local (SL), Tier-based Strictly Local (TSL), and Multi-tiered Strictly Local (MTSL) functions that I have been using thus far assume that every input has exactly one output (i.e., they are deterministic). Consequently, these functions can only describe either mandatory application or mandatory non-application of a process. Real language data is, however, not always this clean; many phonological processes apply *optionally*, and long-distance processes are no exception to this fact. This chapter will explore how the requirement of determinism can be relaxed in order to describe the probabilistic application of a process, while still maintaining the advantages of (tier-based) strict locality. First, Section 6.1 looks at some optional long-distance patterns and shows how strategically adding transitions to a TSL or MTSL FST and weighting them can describe the desired probabilistic distribution of output forms for a given input. After that, Section 6.2 considers a well-studied phenomenon known as *distance-based decay*, wherein the probability that a long-distance process applies will exponentially diminish as more and more transparent segments intervene between the trigger and target (Zymet, 2015). There I will propose that weighted transducers built according to a TSL or MTSL template can derive distance-based decay in a cognitively plausible manner.

6.1 Weighted probabilistic transducers

Back in Section 3.2.3, I mentioned that Slovenian contained an optional process of sibilant harmony. The transducer in Figure 6.1 shows what an idealized and mandatory version of the Slovenian pattern would look like, ignoring the *blocking* behaviour of [t] and [d] for ease of exposition. An input [+anterior] sibilant (i.e., /s/ or /z/) will map faithfully to itself if no tier-elements have been produced thus far, or if the most recently produced tier element was another [+anterior] sibilant. On the other hand, if the most recently produced tier element was a [−anterior] sibilant (i.e.,

[ʃ] or [ʒ]), an input [+anterior] sibilant will instead palatalise to become its [−anterior] equivalent. Note that transitions labelled '*:*' represent an arbitrary non-sibilant input segment mapping faithfully to itself.

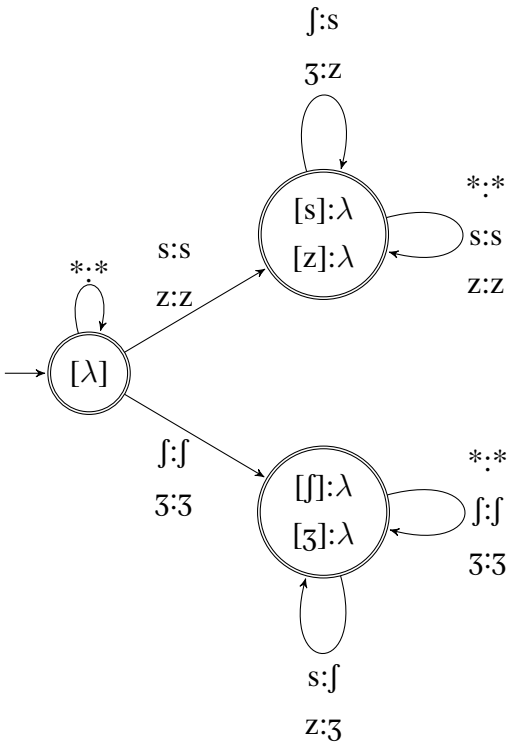


Figure 6.1: An OTSL_2 transducer that produces mandatory sibilant harmony

When this transducer reads an input like /pozabiʃ/ from right to left, it will be in the [−anterior] state as it goes to read /z/, and the corresponding transition will enforce harmony. We want, however, to have the possibility of faithfully producing [z] for /z/ while in the [−anterior] state, since harmony is optional in Slovenian (Jurgec, 2011). We can create the possibility of optional faithfulness by adding transitions from the [−anterior] state to the [+anterior] state labelled 's:s' and 'z:z' and transitions from the [+anterior] state to the [−anterior] state labelled 'ʃ:ʃ' and 'ʒ:ʒ'. Figure 6.2 shows the resulting transducer, which aside from the added non-determinism, exhibits all the required behaviour of an OTSL_2 transducer (i.e., all transitions still land in the state corresponding to the resulting output tier suffix). For clarity, the harmony-enforcing transitions are shown as dotted lines and the harmony-ignoring (faithful) transitions are shown as dashed lines.

Now we have two transitions out of the [−anterior] state for the input /z/, one that enforces harmony and one that enforces faithfulness. Assuming that we choose randomly between the two available transitions, the /z/ in /pozabiʃ/ then has a 50% chance of harmonizing with the nearby [ʃ]. It is very important that the new faithful transition leads to the [+anterior] state rather than looping

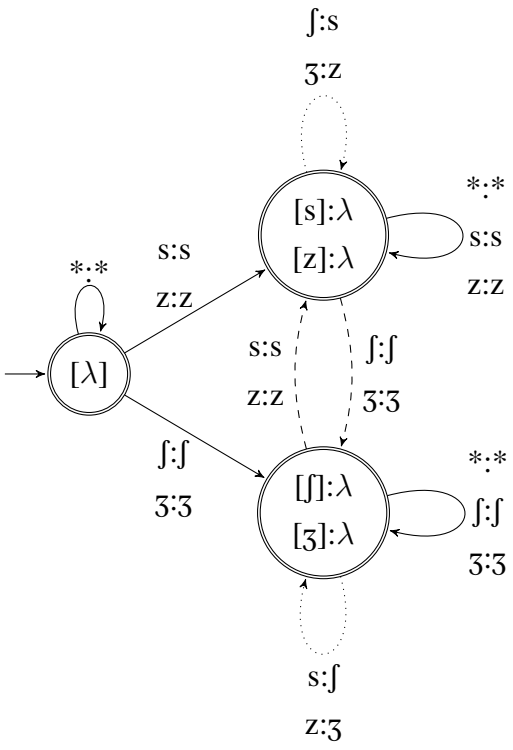


Figure 6.2: A quasi-OTSL$_2$ transducer that produces optional sibilant harmony

back to the [−anterior] state. This is because, while the new faithful transition does not produce harmony, it nonetheless produces a tier element. By ensuring that any additional transitions all lead to the state associated with the updated tier suffix, we maximally preserve the intuitions of the TSL functions, even though we are abandoning determinism and thus no longer meet the definition of a TSL function. Specifically, it is still the case that the set of possible outputs at a given step is directly determined by the most recently produced tier element.

Of course, we may want to achieve a rate of harmony higher than 50% while still allowing for the possibility of faithfulness. To do so, we can assign a numerical weight to each transition in the machine (see Vidal et al. 2005a,b for a more in-depth presentation of probabilistic finite-state machines than the one given in this chapter). When more than one transition could be followed at a given step in the derivation, the probability that we choose a given member from that set of transitions is proportional to its share of the summed weights of the set. For ease of interpretation, I assume that the weight of a transition is equal to the probability that it is followed, meaning that given a state q and an input symbol a, the weights of all transitions leaving q for the input a must add up to 1. Suppose now that the harmony-ignoring (dashed) transitions are weighted 0.2, the harmony-enforcing (dotted) transitions are weighted 0.8, and the remaining transitions are weighted 1. The input /sapozabiʃ/ has three possible outcomes whose probabilities sum to 1: a

fully faithful candidate [sapozabiʃ], a single harmony candidate [sapoʒabiʃ], and a full harmony candidate [ʃapoʒabiʃ]. The fully faithful candidate has a probability of $0.2 * 1 = 0.2$ since the probability of the /z/ remaining faithful is 0.2, and if it does so, the /s/ is guaranteed to be faithful. The remaining output probabilities can be calculated in a similar manner: the single harmony candidate has a probability of $0.8 * 0.2 = 0.16$, and the full harmony candidate has a probability of $0.8 * 0.8 = 0.64$. More generally, a weighted transducer set up in the above manner produces a conditional distribution over a finite set of output strings for each possible input string. The number of output possibilities as well as their shape and share of the probability can of course change from input to input and from transducer to transducer, but the distributions so-defined are crucially related to and dictated by tier-based strict locality.

A particularly interesting case of optionality in a long-distance process comes from Bukusu. Recall from Section 2.1.1 that in Bukusu, a suffixal /l/ is realized as [r] if preceded by an [r] somewhere in the stem (de Blois, 1975; Odden, 1994; Hansson, 2010). For example, compare /xam-il-a/ $\rightarrow$ [xam-il-a] 'milk for' where the applicative suffix /il/ surfaces faithfully and /bir-il-a/ $\rightarrow$ [bir-ir-a] 'pass for' where the same suffix harmonizes with an earlier [r]. Harmony occurs across just a single vowel here, and while harmony can proceed at further distances in Bukusu, it typically becomes optional outside the *transvocalic* context (Hansson, 2010). For example, /ruk-il-a/ 'plait for' may surface as [ruk-il-a] without harmony or as [ruk-ir-a] with harmony. Another long-distance pattern that is cited as being obligatory in transvocalic contexts but optional at further distances would be the sibilant harmony in Kinyarwanda (Kimenyi, 1979; Coupez, 1980; Hansson, 2010; Walker & Mpiranya, 2006; Walker et al., 2008). Such a transvocalic / beyond-transvocalic dichotomy is not possible to describe using a probabilistic OTSL_2 transducer as I did for Slovenian sibilant harmony above, but *is* possible to describe using a probabilistic OMTSL_2 transducer.

Consider the transducer in Figure 6.3, where 'V' stands for an arbitrary vowel and 'C' stands for an arbitrary non-liquid consonant. Ignoring the dashed transition for now, this machine represents a weakly target-specified OMTSL_2 function that computes a fully obligatory version of the Bukusu pattern over a tier of liquid consonants $A = \{\text{r, l}\}$ and a tier of all consonants $B = \{\text{r, l, C}\}$. The left and right symbol of each state label correspond respectively to the suffix on A (i.e., the most recent liquid consonant) and B (i.e., the most recent consonant). To avoid confusion with the state labels, the final string associated to each string appears alongside the special symbol $\ltimes$ which serves as a word-end marker. Reaching the [r, r] state can be interpreted to mean that the most recent consonant we have seen is an [r] (since it is on both the liquid and consonantal tiers). Compare this to being in the [r, C] state, which means that the most recent consonant we have seen is a non-liquid, and that this consonant is preceded by an [r]. In a transducer that does not contain the dashed transition, both of these states enforce the harmonic /l/ $\rightarrow$ [r] change, as indicated with dotted lines. However, by adding the dashed transition, harmony becomes optional just in those

cases where [r] is the most recently produced liquid but not the most recently produced consonant. In a language like Bukusu with mostly open CV syllables, these two states more-or-less reflect the difference between a transvocalic and beyond-transvocalic distance from the most recent liquid consonant.

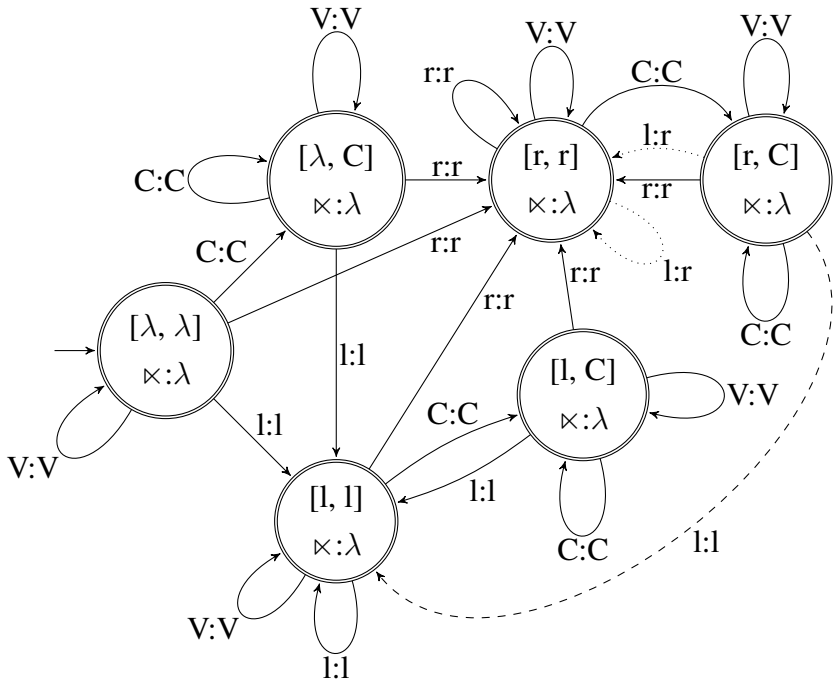


Figure 6.3: A quasi-OMTSL$_2$ transducer that computes Bukusu liquid harmony

An important question arises when we model optional processes using probabilistic transducers. Namely, how do we determine the transition weights that best reflect the target pattern? This type of optimization problem is well-studied in the literature on Probabilistic Finite-state Acceptors (PFAs) which are exactly like probabilistic transducers except that rather than taking an input string and producing an output string, they take an input string and return a numerical value reflecting the input's well-formedness. The Slovenian and Bukusu transducers above can be reinterpreted as acceptors if we think of their transition labels as atomic elements of an alphabet and rewrite input-output pairs as a string of such "transducer steps". Conveniently, reinterpreting the Slovenian and Bukusu transducers in this manner makes them deterministic since, while a given input string can follow potentially multiple paths through them to produce different outputs, a given input-output pair can only be achieved by following a single, specific path through them. Finding the transition weights for a given deterministic PFA that maximise the probability of a set of training data has a well-known, simple, and efficient solution. For each transition,[1] we calculate the number of times

[1]The final strings associated to states are treated as transitions for the purposes of this optimization, effectively

it was followed when reading the sample and divide this number by the total number of times its origin state was visited when reading the sample (Vidal et al., 2005a,b; de la Higuera, 2010). One small modification is needed for our purposes since the weights resulting from the above method will describe a single distribution over input-output pairs, rather than a separate distribution over output strings for each input. This is because the weights of all transitions out of a given state will sum to 1, whereas we want all transitions out of a given state *for a given input element* to sum to 1. To remedy this, we can normalize the transducer by taking each combination of state and input symbol, adding together the weights of all transitions leaving that state for that input element, then dividing each of the implicated transition weights by this sum.

6.2 Distance-based decay

The analyses of the Slovenian and Bukusu cases above are relatively simplistic in that the probability with which the process applies remains constant. In many cases, however, we see that the probability of application is inversely correlated to the distance between trigger and target. This phenomenon is known as *distance-based decay* (Zymet, 2015) and can be observed in Malagasy vowel rounding dissimilation (Zymet, 2015), Hungarian backness vowel harmony (Hayes & Londe, 2006; Hayes et al., 2009), Latin liquid dissimilation Zymet (2015), and Navajo sibilant harmony (Martin, 2005), among others. Current descriptions of the phenomenon are couched within stochastic constraint-based frameworks like Noisy Harmonic Grammar (Coetzee & Pater, 2011) and Maximum Entropy grammar (Goldwater & Johnson, 2003; Hayes & Wilson, 2008). These descriptions propose that the weight of a process-enforcing constraint is scaled down proportionally to the distance between trigger and target (Kimper, 2011; Zymet, 2015). Distant trigger-target pairs incur smaller penalties than more local pairs, and as a result, the process applies at a lower probability in the distant pair than in the more local pair (Kimper, 2011; Zymet, 2015). Focusing on Malagasy and Hungarian, I will show how distance-based decay can equally be captured through minor modifications to a TSL or MTSL transducer.

6.2.1 Malagasy

Malagasy has a process of vowel rounding dissimilation whereby the passive imperfective suffix /-u/ becomes [-i] when preceded by an [u], as can be seen in /babu-u/ $\rightarrow$ [babu-i] 'plunder-PASS.IMP'. Front vowels are opaque to the process as can be seen with /turi-u/ $\rightarrow$ [turi-u] 'preach-PASS.IMP' and /ure-u/ $\rightarrow$ [ure-u] 'massage-PASS.IMP'. In contrast, the vowel /a/ is transparent to dissimilation, as can be seen with /gurabah-u/ $\rightarrow$ [gurabah-i] 'splutter-PASS.IMP'. If we ignore its

acting as a transitions that lead to a dedicated "stopping" state with no associated string of its own.

dashed transition (discussed further below), the OTSL$_2$ transducer in Figure 6.4 captures a mandatory version of the Malagasy pattern just described. Dissimilation specifically affects the passive imperative suffix rather than /u/ in general (Zymet, 2020), so for convenience I assume that this transducer only ever reads verb stems and adds the appropriate suffix allomorph upon reaching the end of the base.

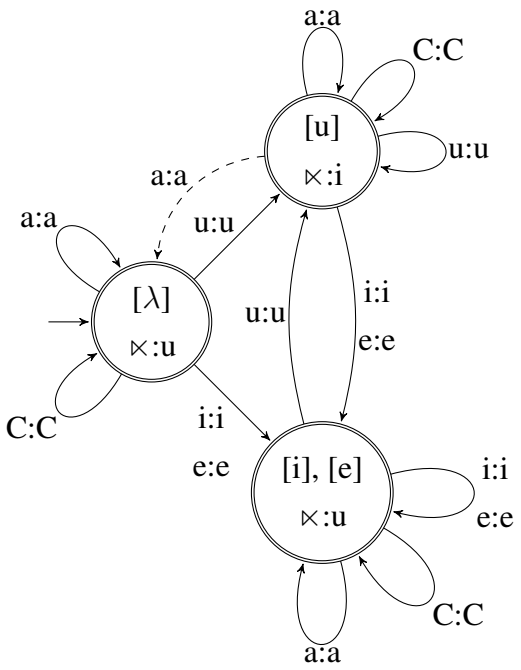


Figure 6.4: A quasi-OTSL$_2$ transducer that produces Malagasy dissimilation

Malagasy dissimilation is not categorical, however, and exhibits distance-based decay. According to Zymet's (2015) survey of de la Beaujardière's (2004) online Malagasy dictionary, the probability of dissimilation is 0.99 (989/993) when the trigger and target are in adjacent syllables, 0.51 (201/397) when they are separated by one transparent syllable, 0.13 (4/32) when they are separated by two transparent syllables, and 0 (0/4) when they are separated by three transparent syllables. Due to the language's mostly open-syllable nature, the number of transparent syllables corresponds with how many transparent vowels fall between the trigger and target, and the probability of dissimilation is roughly $1/2^x$, where x represents the number of intervening transparent vowels. This opens an interesting route to deriving the distance-based decay using the structure of the transducer in Figure 6.4: whenever the transducer reads /a/ and produces [a] while in the [u] state, it has a roughly 50% chance to "forget" that it previously produced an instance of [u]. In this case, the transducer will follow the dashed transition which returns to the [λ] state instead of looping back to the [u] state, and will consequently fail to dissimilate the passive imperative suffix. As more transparent vowels are encountered while in the [u] state, the machine is exponentially less

likely to remember that it encountered a dissimilation trigger, giving us the negative exponential curve in the probability of dissimilation. A cognitive interpretation of forgetful transitions would be that the memory of the most recent tier element decays over time (measured by the number of intervening elements).

Two related questions arise when modelling distance-based decay by augmenting a TSL transducer with forgetfulness parameters. First, how do we decide which forgetful transitions should be added to the TSL transducer? Any transition that fails to produce an element from the tier can presumably be given a forgetful version, but including too many or too few of these could negatively impact the accuracy of our model. Second, given a fixed set of forgetful transitions, how do we determine their optimal weights? I answer the latter question first, since its solution will be considered when approaching the former question.

Recall from Section 6.1 that to optimize the weights of a deterministic acceptor, it is sufficient to read through the provided sample once and count the number of times that each transition is followed. Unfortunately, even after reinterpreting a forgetful TSL transducer as an acceptor, it is still non-deterministic. Given an input-output pair we can generally tell whether forgetting did or did not occur, but we cannot tell exactly where the forgetting took place when there is a sequence of more than one transparent element. Because we cannot always know the exact path that an input-output pair followed through the machine, we cannot accurately count the number of times each transition get traversed when the sample is read. It is, however, possible to estimate these counts given the machine's current transition weights using what are called forward and backward probabilities. Consider the input element x in the string $w = u \cdot x \cdot v$. For any transition of the shape (q, x, q') we can calculate the probability that we are in state q after having read u (the forward probability) and the probability that we produce v when starting in state q' (the backward probability).[2] Multiplying the current weight of a given transition by its forward and backward probability and then dividing by the probability of the whole string gives us the probability that we actually traversed the transition on that reading step (de la Higuera, 2010, pp. 362-363).

By using estimated traversal probabilities as our traversal counts, we can calibrate the weights of a non-deterministic acceptor using the same division operations as for a deterministic acceptor. If we cycle through the estimation and calibration processes just described, the parameter weights will get adjusted by smaller and smaller amounts until they converge. This is known as the Baum-Welch algorithm,[3] originally developed by Baum et al. (1970) and Baum (1972). It is a type of maximum likelihood estimation (MLE) that, metaphorically, climbs the "hill" of sample probabilities by adjusting the available parameter values, and stops when it reaches a peak and cannot increase the sample probability any further. The algorithm is guaranteed to converge on such an

[2]Chapter 5 of de la Higuera (2010) shows how to efficiently calculate forward and backward probabilities.

[3]See chapter 17 of de la Higuera (2010) for a more thorough presentation.

optimum, but non-deterministic machines can have multiple optima in addition to the global optimum, and the algorithm may get trapped in one of these (Vidal et al., 2005b; de la Higuera, 2010). Returning to the hill metaphor, there can be multiple peaks of varying heights and we want the algorithm to find the highest one, but it cannot tell whether the peak it reaches is actually the highest, it simply stops once it finds *any* peak. The only guaranteed way around this is to try several times with different starting values, and then pick the result that gives the best probability, in the hope that the chosen iteration found the global optimum (de la Higuera, 2010, p. 323). Luckily, this was not a serious issue during the tests described further below.

Moving on to the question of which forgetful transitions to include, we could take the stance that we want only the forgetful transitions that significantly affect model performance. So long as the set of forgetful transitions in one optimized transducer are a strict superset of the forgetful transitions in another optimized transducer, it is in theory possible to perform a log-likelihood ratio test to assess whether the additional transitions significantly improve model performance. Given a calibrated transducer, we calculate its log-likelihood by running the training sample through it. For each input-output pair (x, y) we calculate the probability that the machine produces y given x, which is equal to the sum of the probability of all paths through the machine that produce y given x. This might seem difficult to do efficiently since the number of possible paths through a non-deterministic transducer is in the worst-case exponentially proportional to the length of the input string, but we can bypass this issue by calculating forward probabilities (which takes just one pass through the string) and summing over those instead (de la Higuera, 2010, pp. 90-92). If we then take the log of each pair's probability, adding them up gives us the model's log-likelihood. One way to find the best set of forgetful transitions, then, would be to take a forwards selection approach. Starting with no forgetful parameters, we iteratively add the one that would contribute the most until we cannot significantly improve our model any more.

To test the effectiveness of the FST decay model, I created custom Python code that implements the Baum-Welch algorithm and log-likelihood calculation procedures described above, then ran it against the Malagasy data from Zymet (2015). Calibrating the base model affects only the probability that dissimilation occurs while in the [u] state, and the optimal value of 83.73% gives a log-likelihood of -633.30. Most additional forgetful transitions significantly improved model fit on their own,[4] but 'a' had by far the strongest contribution, increasing log-likelihood all the way to -315.34 ($\chi^2_1 = 635.93$, $p = 2.57 \times 10^{-140}$). The next highest contribution came from 'l', which increased log-likelihood to -609.67 ($\chi^2_1 = 47.27$, $p = 6.2 \times 10^{-12}$). A second round of tests using the 'a' model only found two additional forgetful transitions to be significant: these were

[4]Only 'v' ($\chi^2_1 = 3.58, p = 0.06$), 't' ($\chi^2_1 = 2.66, p = 0.1$), 'f' ($\chi^2_1 = 0.49, p = 0.48$) and 'h' ($\chi^2_1 = 0, p = 1$) did not. The last case is particularly interesting in that the optimal weight for a lone forgetful 'h' transition was 0, equivalent to the absence of such a transition.

'dʒ' ($\chi^2_1 = 3.85, p = 0.050$) and 'z' ($\chi^2_1 = 3.93, p = 0.048$). This might seem odd considering how most were highly significant on the first round of tests. Looking at the largely CV syllable structure of Malagasy, though, the presence of an intervening 'a' heavily implies the presence of an intervening consonant. Significant contributions from the lone consonantal parameters may thus have been indirect inheritances from instances of 'a'. Because forgetful 'dʒ' and 'z' transitions are just barely significant given a threshold of $p < 0.05$, I opted not to include either and stop further testing, leaving us with just a forgetful 'a' transition. One reason that 'a' may have near-exclusive entitlement to a forgetful transition is its high similarity to the tier elements, all of which are vowels. Encountering a non-tier element that is highly similar to elements on the tier would intuitively interfere with the maintenance of a tier suffix in long-term memory, although I leave the confirmation of this hypothesis for future research.

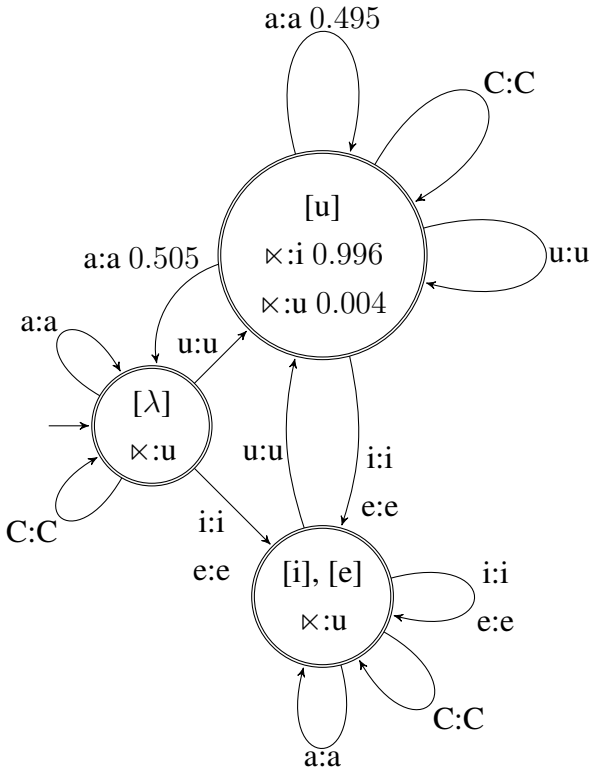


Figure 6.5: An optimized quasi-OTSL$_2$ transducer for Malagasy

The optimized Malagasy transducer is shown in Figure 6.5; all transitions without a displayed weight have a weight of 1. Earlier I mentioned that, as reported by Zymet (2015), the probability of dissimilation is $0.996(989/993)$ when the trigger and target are in adjacent syllables, $0.506(201/397)$ when they are separated by one transparent syllable, $0.125(4/32)$ when they are separated by two transparent syllables, and $0.00(0/4)$ when they are separated by three transparent syllables. A single forgetful transition for [a] pretty faithfully reproduces the probability of

adjacent dissimilation (0.996) and dissimilation across one transparent syllable ($0.996 * 0.495 = 0.493$), but modestly overestimates the probability of dissimilation across two intervening syllables ($0.996 * 0.495^2 = 0.244$) and three intervening syllables ($0.996 * 0.495^3 = 0.121$). Zymet's (2015) constraint-based model more closely reproduces the latter two probabilities, but this may be an instance of overfitting, considering how few forms in the corpus contain $2+$ intervening syllables. In any case, the model with a forgetful [a] transition drastically outperforms the base model, which predicts dissimilation with a probability of 0.837 at any distance.

6.2.2 Hungarian

Including forgetfulness parameters into a single-tiered transducer is sufficient for the Malagasy case, but not all cases of distance-based decay are so easy. Take for instance the backness vowel harmony in Hungarian, to which [i], [e], and [ɛ] are transparent (Hayes & Londe, 2006; Hayes et al., 2009; Kimper, 2011; Ozburn, 2019). While it is generally true that a higher number of transparent vowels between trigger and target will exponentially diminish the probability of harmony, there is an important exception: harmony remains nearly obligatory across a single transparent vowel (Hayes & Londe, 2006; Hayes et al., 2009; Kimper, 2011; Ozburn, 2019). For example, the [ɔ] in [pɔpiːr] 'paper' always triggers the back variant of the dative suffix ([pɔpiːr-nɔk] 'paper-DAT') since it is followed by only one transparent vowel, but the [ɔ] in [ɔspirin] 'aspirin' only optionally triggers the back variant since it is followed by two transparent vowels ([ɔspirin-nɔk] ∼ [ɔspirin-nɛk] 'aspirin-DAT').

This is impossible to model using a single-tiered transducer, even with forgetful transitions. To see why, consider the transducer fragment in Figure 6.6, which determines the appropriate allomorph of the dative suffix /-nEk/ for bases containing only high vowels.[5] The underspecified suffix vowel /E/ must harmonize while in either the /u/ or /y/ state, and defaults to front while in the [λ] state. The vowel /i/ is transparent to harmony and so each of these states has a looping transition labelled 'i:i'. We could try modelling the distance-based decay using the forgetful transitions marked with dashed lines, but these do not distinguish between having one transparent vowel and having two or more transparent vowels between trigger and target. We want forgetfulness to begin applying only in the latter case, but there is no way to set such a threshold on the required number of transparent segments in a single-tiered transducer. In such a transducer, a transparent segment either always or never has the opportunity to cause forgetfulness.

Interestingly, the desired $2+$ threshold can be modelled using a multi-tiered transducer as the base. Suppose we have one tier V that tracks all vowels (i.e., including the transparent [i], [e],

[5] I am assuming here for simplicity that harmony only affects underspecified suffix vowels, and that all base-internal vowels are fully specified in underlying forms.

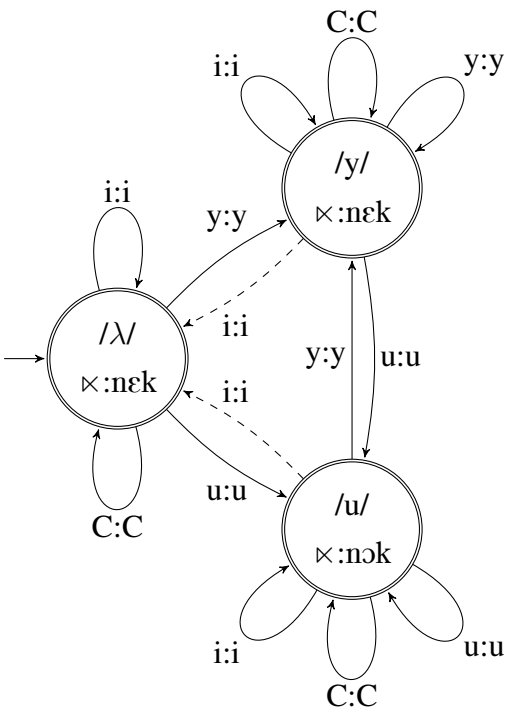


Figure 6.6: A quasi-ITSL$_2$ transducer fragment that enforces Hungarian suffixal harmony

and [ɛ]), and another tier H that tracks only the harmony-triggering vowels (i.e., excluding the transparent [i], [e], and [ɛ]). This allows us to distinguish cases where the most recently produced vowel is a harmony trigger (i.e., the suffixes on V and H coincide) and cases where the most recently produced vowel is transparent but is itself preceded by a harmony trigger (i.e., the suffixes on V and H do not coincide). These two cases constitute two different states in the corresponding multi-tiered transducer, and by ensuring that forgetful transitions only originate from states where the two tier suffixes do not coincide, harmony will be obligatory across a single transparent vowel, although any additional transparent vowels will cause distance-based decay.

Consider now the transducer fragment in Table 6.1, which again determines the appropriate allomorph of the dative suffix /-nEk/ for bases containing only high vowels. Drawing the transducer in a legible manner is tricky, so I have opted to represent it as a collection of tables where each row corresponds to a transition. Rows are organized according to their origin state, and a label /a, b/ corresponds to having /a/ as the suffix on H (the tier of harmonic vowels) and /b/ as the suffix on V (the tier of all vowels). Notice how the /u,u/ and /u,i/ states both enforce harmony, but only the latter has a forgetful transition labelled 'i:i' (i.e. it has two rows for input /i/ in the table). Being in the /u,u/ state means that we have not read any transparent vowels after reading the most recent harmony-triggering vowel, while being in the /u,i/ state means that we have read *at least one* transparent vowel after reading the most recent harmony-triggering vowel. The transition

labelled ‘i:i’ leaving /u,u/ and landing in /u,i/ does not have a forgetful counterpart, and so harmony remains obligatory across this one transparent vowel. The transition labelled ‘i:i’ leaving /u,i/ and looping back to /u,i/ does, however, have a forgetful counterpart. There is thus a chance that reading a second transparent vowel (and third, and fourth, etc.) will cause us to forget having read /u/. Forgetting that we read /u/ will bring us to the /λ,i/ state, which corresponds to thinking that we have not read a harmony trigger yet (or at the very least, not remembering the identity of the most recent harmony trigger). The decaying probability of harmony then results from the fact that it is increasingly unlikely to remain in the harmony-enforcing /u,i/ state as we read more and more transparent vowels.

Origin	Input	Output	Landing	Origin	Input	Output	Landing
/λ,λ/	/C/	[C]	/λ,λ/	/λ,i/	/C/	[C]	/λ,i/
/λ,λ/	/i/	[i]	/λ,i/	/λ,i/	/i/	[i]	/λ,i/
/λ,λ/	/y/	[y]	/y,y/	/λ,i/	/y/	[y]	/y,y/
/λ,λ/	/u/	[u]	/u,u/	/λ,i/	/u/	[u]	/u,u/
/λ,λ/	/$\ltimes$/	[nɛk]	NA	/λ,i/	/$\ltimes$/	[nɛk]	NA
Origin	Input	Output	Landing	Origin	Input	Output	Landing
/y,y/	/C/	[C]	/y,y/	/u,u/	/C/	[C]	/u,u/
/y,y/	/i/	[i]	/y,i/	/u,u/	/i/	[i]	/u,i/
/y,y/	/y/	[y]	/y,y/	/u,u/	/y/	[y]	/y,y/
/y,y/	/u/	[u]	/u,u/	/u,u/	/u/	[u]	/u,u/
/y,y/	/$\ltimes$/	[nɛk]	NA	/u,u/	/$\ltimes$/	[nɔk]	NA
Origin	Input	Output	Landing	Origin	Input	Output	Landing
/y,i/	/C/	[C]	/y,i/	/u,i/	/C/	[C]	/u,i/
/y,i/	/i/	[i]	/y,i/	/u,i/	/i/	[i]	/u,i/
/y,i/	/i/	[i]	/λ,i/	/u,i/	/i/	[i]	/λ,i/
/y,i/	/y/	[y]	/y,y/	/u,i/	/y/	[y]	/y,y/
/y,i/	/u/	[u]	/u,u/	/u,i/	/u/	[u]	/u,u/
/y,i/	/$\ltimes$/	[nɛk]	NA	/u,i/	/$\ltimes$/	[nɔk]	NA

Table 6.1: A quasi-IMTSL$_2$ transducer fragment that enforces Hungarian suffixal harmony

Hayes & Londe (2006) and Hayes et al. (2009) collected a corpus of Hungarian noun bases (available at `https://linguistics.ucla.edu/people/hayes/HungarianVH/index.htm`), marking the percentage of times each base appears with the back versus front allomorph of the dative suffix (determined using Google search results). To ascertain whether and how much an MTSL model of decay outperforms a TSL model, I optimized a TSL-

like transducer and an MTSL-like transducer against this corpus using a modified Baum-Welch algorithm. Unlike for the Malagasy tests above, we are not trying to maximize the probability of each datum in the sample, but trying to replicate the indicated probability of each datum as closely as possible. Each 'base + allomorph' combination with greater than 0 probability was thus treated as a separate datum, and the amount contributed by a reading step in that datum to a transition's estimated traversal count was multiplied by the datum's probability. Essentially, this would treat a training datum with an indicated probability of, for example, 0.75 as being 75% of a datum (i.e., a datum that was observed 0.75 times). Consequently, the optimization procedure is maximizing a weighted version of model log-likelihood rather than the regular log-likelihood. Where $P(o \mid i)$ is the probability of the input-output pair (i, o) as indicated in the sample S, and where $\hat{P}(o \mid i)$ is the probability of the input-output pair (i, o) predicted by the model M, regular and weighted log-likelihood can be expressed as in (70). There were only ever up to two output possibilities o_1 and o_2 for a given input string i,[6] and their observed probabilities $P(o_1 \mid i)$ and $P(o_2 \mid i)$ always summed to 1, as did their predicted probabilities $\hat{P}(o_1 \mid i)$ and $\hat{P}(o_2 \mid i)$. Maximizing the weighted log-likelihood ensures that we are on average minimizing the distance between the points $\langle \hat{P}(o_1 \mid i),\ \hat{P}(o_2 \mid i) \rangle$ and $\langle P(o_1 \mid i),\ P(o_2 \mid i) \rangle$ for the input strings in the sample.

(70) Regular model log-likelihood:

$$L(M \mid S) = \sum_{(i,o) \in S} \log(\hat{P}(o \mid i))$$

Weighted model log-likelihood:

$$L_W(M \mid S) = \sum_{(i,o) \in S} \log(\hat{P}(o \mid i)) \cdot P(o \mid i)$$

The TSL-like transducer had three states: /λ/ when there was no known preceding harmonic vowel, /F/ when the most recent harmonic vowel was front, and /B/ when the most recent harmonic vowel was back. It had forgetful transitions for /iː/, /i/, /e/, and /ɛ/ leading to the /λ/ state out of the /B/ state. The /F/ state had no forgetful transitions since Hayes & Londe (2006) found that transparent vowels never block front harmonic vowels from imposing a front allomorph. The MTSL-like transducer had the 7 states listed in (71). The states /Biː/, /Bi/, /Be/, and /Bɛ/ were kept separate, rather than having a single 'back + transparent' state since Hayes & Londe (2006) found that the height of the most recent transparent vowel has a significant effect on the probability of a back allomorph. Each of the states /Biː/, /Bi/, /Be/, and /Bɛ/ had forgetful transitions for /iː/, /i/, /e/, and /ɛ/ leading to the state /N/.

(71) States in the MTSL-like Hungarian transducer

[6] The one with the front allomorph of the dative suffix and the one with the back allomorph of the dative suffix.

- /λ/ = no preceding vowel OR the most recent vowel is transparent with no known preceding harmonic vowel
- /F/ = the most recent harmonic vowel is front
- /B/ = the most recent vowel is back
- /Biː/ = the most recent harmonic vowel is back but the most recent vowel is iː
- /Bi/ = the most recent harmonic vowel is back but the most recent vowel is i
- /Beː/ = the most recent harmonic vowel is back but the most recent vowel is e
- /Bɛ/ = the most recent harmonic vowel is back but the most recent vowel is ɛ

Unfortunately, a log-likelihood ratio test cannot compare the two models because their parameters are not strictly nested; none of their forgetful transitions originate from equivalent states. Accordingly, their relative performance was assessed using AIC. Lower AIC values are preferred, and a model's AIC is equal to 2 times its number of free parameters minus 2 times its log-likelihood. The TSL-like model had a weighted log-likelihood of -266.90 and had 7 free parameters, so its AIC is 547.8. For its part, the MTSL-like model had a weighted log-likelihood of -249.73 and had 23 free parameters, so its AIC is 545.46. Going from the TSL model to the MTSL model reduces AIC by 2.34, which is not substantial, but nonetheless favours the MTSL model.

Visual inspection of the weights in the optimized transducers suggests that this is because the data warrant a division between spans of 1 versus $2+$ transparent vowels, a distinction that is possible in the MTSL model, but not in the TSL model. For example, the MTSL model assigns a probability of about $0.97 * 0.55 = 0.53$ to the form [ɔspirin-nɔk] 'aspirin-DAT' since the probability of harmony while in the state /Bi/ (i.e., across a single instance of /i/) is about 0.97 and the probability of /i/ not causing forgetfulness out of the state /Bi/ is about 0.55. Compare this to the TSL transducer which assigns the same form a probability of about $0.99 * 0.93^2 = 0.86$ since the probability of harmony while in the state /B/ is about 0.99 and the probability of /i/ not causing forgetfulness out of the state /B/ is about 0.93. The actually observed frequency of harmony for this noun is 0.21 and so the MTSL transducer, while still a fair ways off, is much closer than the TSL transducer.

Consistent with this interpretation of the models' differing performance, Table 6.2 compares the observed average probability of harmony against the average probabilities predicted by the TSL and MTSL models, broken down by the number of intervening transparent syllables. Taking every noun for which a back allomorph is possible (i.e., whose rightmost harmonic vowel is back), we find that adjacent harmony has an average probability of 0.99 (5317 eligible nouns), harmony across a single transparent vowel has an average probability of 0.67 (370 eligible nouns), harmony across two transparent vowels has an average probability of 0.18 (63 eligible nouns) and harmony

across three transparent vowels has an average probability of 0.00 (8 eligible nouns). In particular, we see that the TSL model overestimates the probability of harmony across two transparent vowels, whereas the MTSL model closely matches the observed probabilities in all cases.

Transparent syllables	Observed average	TSL average	MTSL average
0	0.99	0.99	0.99
1	0.67	0.64	0.67
2	0.18	0.33	0.19
3	0.00	0.03	0.01

Table 6.2: Average probability of Hungarian harmony by number of transparent syllables

Interestingly, both the trained TSL model and the trained MTSL model reproduce an additional aspect of Hungarian harmony whereby vowel height gradiently affects the degree to which a front unrounded vowel is transparent. Specifically, lower front unrounded vowels are "less transparent" than higher front unrounded vowels (Hayes & Londe, 2006; Hayes et al., 2009; Kimper, 2011; Rebrus & Törkenczy, 2016; Ozburn, 2019). In the optimized TSL model, the forgetfulness parameters out of state /B/ for /i/, /iː/, /eː/, and /ɛ/ are weighted 0.065, 0.11, 0.23, and 0.91 respectively, so lower vowels cause more forgetfulness than higher vowels. In the optimized MTSL model, the probability of appending the back allomorph upon ending in the /Bi/, /Biː/, /Beː/ and /Bɛ/ states is respectively 0.97, 0.99, 0.80, and 0.11, so backness harmony is more likely across a higher vowel than a lower vowel. The same height effect is also somewhat apparent in the MTSL model's forgetfulness parameters: the parameters for /iː/, /eː/, and /ɛ/ have average weights of 0.51, 0.78, and 0.91 respectively, mimicking the trend in the TSL model's forgetfulness parameters. The mimicking is not perfect, however, as the average forgetful weight for the vowel /i/ is unexpectedly high at 0.8. This mismatch could perhaps be because back vowels are only uncommonly followed by a chain of multiple front unrounded vowels, making the weights of the MTSL model's forgetfulness parameters a less reliable reflection of the height effect. Indeed, there were cases where the forgetful and non-forgetful transitions for the same vowel out of the same state both had a weight of 0, meaning that neither transition was ever crossed by the sample. These transitions effectively do not exist and were not considered when calculating the average weights.

6.3 Chapter summary

Prior to this chapter, I was working under the convenient assumption that phonological processes were an all or nothing affair, but long-distance processes are often optional in reality. I showed

here that probabilistic transducers with the right structure can capture probabilistic application while maintaining the relative computational simplicity afforded to us by tier-based strict locality. In particular, I demonstrated that distance-based decay can be modeled by strategically adding duplicate non-tier transitions that make the transducer forget the identity of the most recent tier-element, in a sense causing the machine's memory to deteriorate over time. Using established techniques for optimizing the weights of probabilistic automata, I found that these models performed well relative to two real-language data sets. The Malagasy case study suggested that the presence and strength of forgetful transitions might be tied to similarity. Finally, the Hungarian case study showed that an MTSL model can distinguish between spans of 1 versus $2+$ transparent elements, which may be a useful ability for some patterns.

Chapter 7

Learning tier-based functions

A key component of any linguistic theory is how it relates to language acquisition. On a basic level, theories address this question of learnability by placing limits on the grammars that learners can and cannot possibly hypothesize. Any pattern that is inexpressible using the theory's formalisms is inaccessible to a learner *a priori*. For instance, we could adopt the hypothesis that human language phonology is contained within the regular region of the Chomsky hierarchy, which would prevent a learner from acquiring anything more complex than a regular language or function (such as a pattern of "majority rules" harmony). Now, while it is true that knowing the range of possible hypotheses is important, it does little to explain acquisition. We must go a step further and ask whether a learner can efficiently search the hypothesis space created by the theory. The hypothesis that human language phonology approximates the regular region now encounters a serious hurdle. Namely, it was formally proven by Gold (1967) that the regular languages are not in general learnable from positive data alone. I am not aware of a parallel proof for the unlearnability of the regular relations, but the consensus in the field is that the relations are also not learnable from positive data alone (Chandlee, 2014, p. 109).

These last facts are a major inspiration behind much of the work on subregular classes of formal languages and functions. Indeed, several of the subregular classes have been shown to be learnable efficiently from just positive data. This has been shown for the Strictly Local languages (Garcia et al., 1990), the Strictly Piecewise languages (Heinz, 2009, 2010a),[1] the Tier-based Strictly Local languages (Jardine & Heinz, 2016; Jardine & McMullin, 2017; McMullin et al., 2019), the subsequential functions (Oncina et al., 1993; Oncina & Varó, 1996; Castellanos et al., 1998), and the Strictly Local functions (Chandlee, 2014; Chandlee et al., 2014, 2015). This chapter of my thesis will explore the learning of TSL functions. Preliminary work in this regard by Burness &

[1] These are languages that care about ordering but not contiguity. They can model many of the same things as TSL languages, although they cannot model blocking effects (McMullin & Hansson, 2016)

McMullin (2019) showed that Chandlee et al.'s (2015) method for learning OSL_k functions can also learn the OTSL_k functions, provided that the tier is known in advance. Burness & McMullin (2019) also showed that the tier of an OTSL function can be inferred from positive data when $k = 2$. The contribution of this chapter will be to extend these learnability results to other types of tier-based functions proposed throughout this thesis.

This chapter is structured as follows. Section 7.1 defines the learning paradigm within which I will frame my results and presents some relevant results from previous work on learning subregular languages and functions. Section 7.2 then illustrates how a slight modification of Chandlee et al.'s (2015) learning algorithm for OSL_k functions can learn the ITSL_k and OTSL_k functions when their tiers are known in advance. Following that, Section 7.3 presents a refined, much more efficient version of Burness & McMullin's (2019) method for learning the tier of any OTSL_2 function (and, with slight modification, the tier of any ITSL_2 function). Section 7.4 extends this result to target-specified MTSL_2 functions. Subsection 7.4.1 shows how successful tier learning is guaranteed for any *strongly* target-specified MTSL_2 functions, whereas Subsection 7.4.2 outlines how at least some *weakly* target-specified MTSL_2 functions could be learned. Unlike the other results in this chapter, that latter method does not (yet) have any proofs of convergence or data/time complexity.

7.1 Learning paradigm and relevant prior results

I adopt the criterion for successful learning that requires exact identification in the limit from positive data (Gold, 1967). This means an algorithm attempting to learn a target language/function from the desired class must eventually converge upon the correct grammar/transducer after some finite number of examples drawn from the target, and will not alter its hypothesis past this point in time (i.e., feeding it additional examples will just reinforce what it already knows to be true about the target). Additionally, the algorithm must succeed in this way on *all* members of the desired class. For instance it is not enough that an algorithm can learn a particular Strictly k-Local language, it must be able to learn *all* SL_k languages. Of course, guaranteed success is only part of the problem. A learning algorithm should also be *efficient* in that it does not require a prohibitively large number of examples (proportional to the size of the target grammar/transducer) before it can succeed and does not run for an excessively long time (proportional to the amount of data it is given). I accordingly impose the additional requirement of polynomial bounds on time and data (de la Higuera, 1997). Formal definitions of the paradigm are provided below in the context of functions, adapted from Chandlee et al. (2015).

Definition 7.1 Representation**.** *A class $\mathbb{T}$ of functions is represented by a class $\mathbb{R}$ of representations if every $r \in \mathbb{R}$ is of finite size and there is a total and surjective naming function $\mathcal{X} : \mathbb{R} \to \mathbb{T}$ such*

that $\mathcal{X}(r) = t$ if and only if for all inputs $w \in \Sigma^*$:

- *When the output $t(w)$ is defined, $r(w) = t(w)$*
- *When the output $t(w)$ is not defined, $r(w)$ is not defined*

Definition 7.2 Sample**.**
A sample S for a function $t \in \mathbb{T}$ is a finite set of data consistent with t, that is to say $(w, u) \in S \implies t(w) = u$. The size of a sample is the sum of the length of the strings it is composed of: $|S| = \sum_{(w,u)\in S} |w| + |u|$.

Definition 7.3 Learning algorithm**.**
A $(\mathbb{T}, \mathbb{R})$-learning algorithm $\mathbb{A}$ is a program that takes as input a sample for a function $t \in \mathbb{T}$ and outputs a representation $r \in \mathbb{R}$.

Definition 7.4 Characteristic sample**.**
For a $(\mathbb{T}, \mathbb{R})$-learning algorithm $\mathbb{A}$, a sample CS is a characteristic sample of a function $t \in \mathbb{T}$ if for all samples $S \supseteq CS$, the algorithm $\mathbb{A}$ returns a representation r such that $\mathcal{X}(r) = t$.

Definition 7.5 Identification in polynomial time and data**.**
A class $\mathbb{T}$ of functions is identifiable in polynomial time and data if there exists a $(\mathbb{T}, \mathbb{R})$-learning algorithm $\mathbb{A}$ and two polynomial equations $p()$ and $q()$ such that:

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

In the introduction to this chapter, I mentioned that the regular languages were proven to not be learnable from just positive data. This lack of guaranteed learning (let alone efficient learning) is a major motivation for research into *subregular* languages. For many such language classes, an intuitive learning method is available which Heinz (2010b) calls *string extension learning*. As an illustration of what this entails, consider the Strictly k-Local languages. These languages decide a word's grammaticality according to the contiguous sequences it contains that have a length of up to k: if the word contains any penalized substring, it is rejected. What a learner can do to learn such a language, then, is keep track of every sequence of up to length k that it has encountered thus far, and hypothesize that its grammar is exactly this list of observed sequences.[2] Contiguous substrings are easy to extract (the amount of time this takes is linearly proportional to the length of the string) and the learner will converge on the correct grammar so long as it sees each permissible sequence

[2]This would result in a *positive* grammar, which is very easily converted into a *negative* grammar, should one desire to do so. See Section 2.1.1 for more on positive/negative SL grammars.

at least once (so the amount of necessary data is linearly proportional to the size of the target grammar). Generally speaking, string extension learning projects narrowly-defined information of a certain type (e.g., contiguous substrings) from an example word into its hypothesized grammar, and cannot remove information once it is added.

When the tier of a TSL_k language is known in advance, one can learn it in almost the same way as an SL_k language. All one must change is that we extract substrings from an example word after erasing all of its non-tier elements. The issue, then, is how to learn the tier efficiently from positive data alone. This was shown by Jardine & Heinz (2016) to be possible when $k = 2$, and later shown by Jardine & McMullin (2017) to be possible for any value of k. Jardine & McMullin's (2017) learning strategy was to compare observed substrings of length $k - 1$, length k, and length $k + 1$ to see which elements can be freely inserted and/or freely deleted. Elements that must be included on the tier either *cannot* be freely inserted and/or *cannot* be freely deleted. A similar method for learning probabilistic tier-based phonotactics is adopted by Gouskova & Gallagher (2019) in the context of a Maximum Entropy constraint-based grammar. Their learner first induces locally-evaluated bigram and trigram constraints that penalize certain contiguous sequences of phonological feature-value matrices. If a trigram constraint induced on this step has an empty feature matrix as its middle member (i.e., a certain pair of sounds cannot appear across *any* single intervening sound), then it is likely that the constraint holds across unbounded distances. Accordingly, the learner will hypothesize a tier equal to the smallest natural class containing the two flanking sounds, and then induce bigram constraints over this tier. More recently, McMullin et al. (2019) devised a method for efficiently learning the multiple co-existing tiers of a Multi-tiered Strictly 2-Local language (i.e., a TSL_2 language that operates over multiple tiers), so long as a given bigram is penalized relative to at most one tier.

Tier-based *languages* thus enjoy strong learnability results, but the focus of this dissertation is on tier-based *functions*. Chandlee et al. (2014) and Chandlee et al. (2015) showed that the SL_k functions are efficiently learnable from positive data. As I will show in the next section, it turns out that the TSL_k functions are easily learned using the methods contained in those papers when the tier is known in advance, much in the same way that a TSL_k language is easily learned in the same way as an SL_k language once its tier is known. Following that, the remainder of this chapter then discusses how the single tier of a TSL_2 function is efficiently learnable, and how this method of tier learning can be extended to at least some subsets of the MTSL_2 functions. In the interest of space, I will only show the results pertaining to output-oriented functions, since the analogous results for input-oriented functions are identical once all references to an output tier are swapped for references to an input tier.

7.2 Learning functions when the tier is known

Algorithm 1 gives pseudo-code for modifications of Chandlee et al.'s (2015) Output Strictly Local Function Inference Algorithm (OSLFIA) that can learn the OTSL$_k$ functions. The only changes made to their algorithm are (i) that it now considers a tier as part of its input and (ii) that it uses different criteria when determining state labels.

Data: A sample S, a tier $T \subseteq \Delta$, and $k \in \mathbb{N}$
Result: An OTSL$_k$ transducer $\mathcal{M} = (\texttt{checked}, q_0, q_f, \Sigma, \Delta, \delta)$
Function `build.fst(`*S, T, k*`)`**:**
  $\texttt{checked} \leftarrow \{q_0, q_f\}$ with $q_0, q_f \notin T^{\leq k-1}$;
  $s \leftarrow \texttt{lcp}(\{y \mid (x, y) \in S\})$;
  $q \leftarrow \texttt{suff}_T^{k-1}(s)$;
  $\texttt{earliest}(q) \leftarrow \rtimes$;
  $\texttt{out}(q) \leftarrow s$;
  $\delta \leftarrow \{(q_0, \rtimes, s, q)\}$;
  $\texttt{reached} \leftarrow \{q\}$;
  while *reached* $\neq \emptyset$ **do**
    $q \leftarrow \texttt{first}(\texttt{reached})$;
    $s \leftarrow \texttt{earliest}(q)$;
    for *all* $a \in \Sigma$ **do**
      if $\exists (w, u) \in S$ *and* $\exists x \in \Sigma^*$ *such that* $w = sax\ltimes$ **then**
        $v \leftarrow \texttt{lcp}(\{y \mid \exists x$ such that $(sax\ltimes, y) \in S\})$;
        $r \leftarrow \texttt{suff}_T^{k-1}(v)$;
        $\delta \leftarrow \delta \cup \{(q, a, \texttt{out}(q)^{-1} \cdot v, r)\}$;
        if $r \notin$ *reached* $\cup$ *checked* **then**
          $\texttt{reached} \leftarrow \texttt{reached} \cup \{r\}$;
          $\texttt{earliest}(r) \leftarrow sa$;
          $\texttt{out}(r) \leftarrow v$;
    if $\exists u$ *such that* $(s\ltimes, u) \in S$ **then**
      $\delta \leftarrow \delta \cup \{(q, \ltimes, \texttt{out}(q)^{-1} \cdot u, q_f)\}$
    $\texttt{reached} \leftarrow \texttt{reached} - \{q\}$;
    $\texttt{checked} \leftarrow \texttt{checked} \cup \{q\}$;
  return $\mathcal{M} = (\texttt{checked}, q_0, q_f, \Sigma, \Delta, \delta)$

Algorithm 1: Modified OSLFIA (Chandlee et al., 2015) for learning OTSL$_k$ functions.

I will begin by generally describing this learning process and then provide a concrete demonstration. The algorithm conducts a breadth-first search of the possible state space, calculating all

possible transitions out of one state before moving to the next, treating the states in the order in which they were discovered. At each step of its main loop, it has a set of discovered states that it has checked (the set `checked`) and a queue of discovered states that can be reached from a checked state but have not been checked themselves (the set `reached`). Each discovered state q is associated with the string $\texttt{earliest}(q)$ equal to the alphabetically earliest input string that makes it reachable, as well as with the string $\texttt{out}(q)$ equal to longest common prefix of the outputs of all input strings in the training data that have $\texttt{earliest}(q)$ as a prefix (this will be $f^p(\texttt{earliest}(q) \cdot a)$ if the data is sufficient). On each iteration of its main loop, the algorithm takes the first unchecked state q and attempts to compute an outgoing transition for each $a \in \Sigma$. To do so, it gathers all inputs in the training data with $\texttt{earliest}(q) \cdot a$ as a prefix, produces a string v equal to the longest common prefix of their outputs, then adds the transition $(q, a, \texttt{out}^{-1}(q) \cdot v, \texttt{suff}_T^{k-1}(v))$. If the landing state $r = \texttt{suff}_T^{k-1}(v)$ of the new transition does not match the label of any known state, the algorithm adds r to the queue, associating it with $\texttt{earliest}(r) = \texttt{earliest}(q) \cdot a$ and $\texttt{out}(r) = v$. Since the algorithm treats each state only once and attempts to compute each possible transition only once, it is guaranteed to produce some FST that is deterministic and onward.

Note that the transducer constructed by this algorithm is a *delimited* transducer, so input strings are obligatorily flanked by special word-start ($\rtimes$) and word-end ($\ltimes$) symbols that are not part of the input alphabet. The dedicated initial state (q_0) of such a transducer has only one outgoing transition (which is for $\rtimes$) and each non-initial/non-final state has a transition for $\ltimes$ that leads to the dedicated final state (q_f).[3] Figure 7.1 shows an example delimited OTSL_2 transducer that computes an asymmetric pattern of sibilant harmony over the alphabets $\Sigma = \Delta = \{\text{s}, \text{ʃ}, \text{o}\}$ and the tier $T = \{\text{s}, \text{ʃ}\}$. The pattern is asymmetric in that preceding [s] will cause /ʃ/ to harmonize, but not vice versa. In the paragraphs that follow, I will demonstrate how the algorithm constructs this transducer given an appropriate sample. Formal proofs of the algorithm's guaranteed convergence and time/data efficiency are provided in Appendix B for the interested reader.[4]

Assume that the sample contains every input $w \in \Sigma^*$ of up to length 3, that the tier is set to $T = \{\text{s}, \text{ʃ}\}$, and that k is set to 2. The algorithm starts with the initial and final states in the set `checked` and goes to calculate the lone transition leaving the initial state. To do so, the algorithm calculates the longest common prefix (LCP) of all outputs in the sample (which here is 'λ'), and since we are set to learn an OTSL_2 function, the algorithm calculates the tier suffix of this LCP (which is also 'λ'). Together these tell the algorithm to add the transition $(q_0, \rtimes, \lambda, \lambda)$ to the machine. This transition leads to a state that is not in `checked` and so the set `reached` is

[3] This transition produces the state's *final string* from the non-delimited version of the same transducer.

[4] Burness & McMullin (2019) actually left these proofs implied in their paper since they are identical to the analogous proofs in Chandlee et al. (2015) aside from different criteria for state labelling.

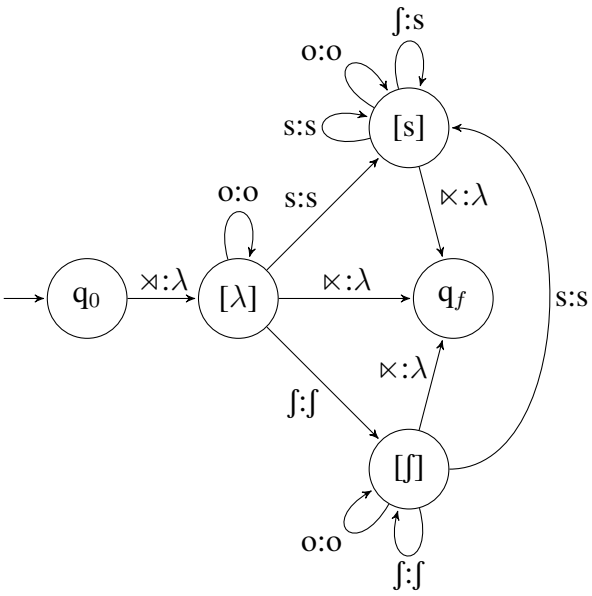


Figure 7.1: A delimited transducer computing asymmetric sibilant harmony.

initialized to $\texttt{reached} = \{\lambda\}$. Furthermore, $\texttt{earliest}(\lambda)$ is set to '$\rtimes$' and $\texttt{out}(\lambda)$ is set to 'λ'. The algorithm will now enter the main learning loop.

The first and only state in `reached` is 'λ' so its earliest string and associated output are accessed, and we go to calculate transitions for each input element. Suppose we start by calculating the transition for 'o'. We will collect all pairs in the sample that have '$\rtimes$o' as a prefix[5] and calculate the LCP of their outputs which will be 'o'. The tier suffix of this string is 'λ' since 'o' is off the tier, so we add the transition $(\lambda,\ \text{o},\ \texttt{out}(\lambda)^{-1}\cdot\text{o},\ \lambda) = (\lambda,\ \text{o},\ \lambda^{-1}\cdot\text{o},\ \lambda) = (\lambda,\ \text{o},\ \text{o},\ \lambda)$ to the machine. No new states have been discovered so we move to the next input element. Suppose we treat 's' next. After gathering all inputs that have '$\rtimes$s' as a prefix we will get 's' as their LCP, and the tier suffix of this string is 's' so we add $(\lambda,\ \text{s},\ \texttt{out}(\lambda)^{-1}\cdot\text{s},\ \text{s}) = (\lambda,\ \text{s},\ \lambda^{-1}\cdot\text{s},\ \text{s}) = (\lambda,\ \text{s},\ \text{s},\ \text{s})$ to the machine which leads to a new state 's'. This state gets added to `reached` and is associated with $\texttt{earliest}(\text{s}) = \rtimes\text{s}$ and $\texttt{out}(\text{s}) = \text{s}$. A similar sequence of steps will add the transition $(\lambda,\ ʃ,\ ʃ,\ ʃ)$ to the machine and add the state 'ʃ' to `reached` which is associated with $\texttt{earliest}(ʃ) = \rtimes ʃ$ and $\texttt{out}(\text{s}) = ʃ$. Finally we add the final transition leaving the state 'λ' by using the pair $(\rtimes\lambda\ltimes, \lambda)$ in the sample to calculate $(\lambda, \ltimes, \texttt{out}(\lambda)^{-1}\cdot\lambda, q_f) = (\lambda, \ltimes, \lambda^{-1}\cdot\lambda, q_f) = (\lambda, \ltimes, \lambda, q_f)$. This completes the treatment of the state 'λ' so it is moved to `checked` and we go to treat the remaining states in $\texttt{reached} = \{\text{s}, ʃ\}$. Repeating the same procedures for these two states will add transitions without adding any new states, and our final result is the transducer in Figure 7.1.

[5]Which includes the pairs ($\rtimes$osʃ$\ltimes$, oss) and ($\rtimes$oʃs$\ltimes$, oʃs), among others.

7.3 Learning a single tier

Above I showed how a transducer computing an OTSL$_k$ function can be constructed when the tier is known. Learners presumably do not have *a priori* knowledge of the tier contents, however, so how should we go about obtaining this knowledge? A gross simplification of what is accomplished by using a tier would be to say that we successfully ignore irrelevant elements. Under this perspective, learning a tier boils down to asking which elements are irrelevant and can thus safely be ignored. But what does it mean to be irrelevant to a phonological process? Burness & McMullin (2019) highlighted two important properties exhibited by OTSL$_2$ functions that let us flag such irrelevant elements. First, it is often possible to describe the same OTSL$_2$ function using any of a number of different tiers (e.g., the function represented in Figure 7.1 can be computed relative to $T = \{\text{s}, \int\}$ or $T = \{\text{s}\}$) and we can freely combine any two or more of these potential tiers to yield a "master" tier that can also be used to describe the function. A consequence of this property is that there will always be a unique "largest" tier that can describe a given OTSL$_2$ function. I follow Burness & McMullin (2019) by designating a function's largest possible tier as its *canonical* tier, and note that the canonical tier is necessarily a strict superset of any other possible tier.

This first important property of OTSL$_2$ functions unfortunately does not extend to higher values of k. To see why, consider the function represented by the transducer in Figure 7.2. To avoid visual clutter, I do not show the initial and final states, nor do I show the transitions to and from them. For this function, $\Sigma = \{a, b, c\}$ and $\Delta = \{a, b, d\}$. In this function, $\texttt{cont}_f(a, w)$ is always a and $\texttt{cont}_f(b, w)$ is always bd, but $\texttt{cont}_f(c, w) = \lambda$ when $\texttt{suff}^3(f^p(w)) = abd$, and $\texttt{cont}_f(c, w) = bd$ otherwise. The transitions representing the former and latter contributions of c are marked with dotted and dashed lines respectively. This function is OTSL$_3$ relative to the tier $T_1 = \{a, b\}$, in which case $\texttt{cont}_f(c, w) = \lambda$ when $\texttt{suff}^2_{T_1}(f^p(w)) = ab$. It is also OTSL$_3$ relative to the tier $T_2 = \{a, d\}$, in which case $\texttt{cont}_f(c, w) = \lambda$ when $\texttt{suff}^2_{T_2}(f^p(w)) = ad$. Note, however, that the function is not OTSL$_3$ relative to the tier $\Omega = T_1 \cup T_2 = \{a, b, d\}$. We have $\texttt{cont}_f(c, ab) = \lambda$ and $\texttt{cont}_f(c, bb) = bd$ even though $\texttt{suff}^2_\Omega(f^p(ab)) = \texttt{suff}^2_\Omega(abd) = bd = \texttt{suff}^2_\Omega(bdbd) = \texttt{suff}^2_\Omega(f^p(bb))$. Including both b and d on the tier inadvertently "pads" the tier window and we lose sight of crucial information (i.e. whether or not the most recent tier symbol prior to bd was a). As for OTSL$_2$ functions, their window cannot be so padded. Uniqueness of the target tier is immensely crucial to successful learning, so I will henceforth only consider functions where $k = 2$.

The second important property of OTSL$_2$ functions relates to their canonical tiers: an element that is not a member of the canonical tier can in fact *never* be a member of *any* tier capable of describing the function. Importantly, if we attempt to describe the function using a strict superset of the canonical tier, there will be at least one pair of mappings telling us that a given element

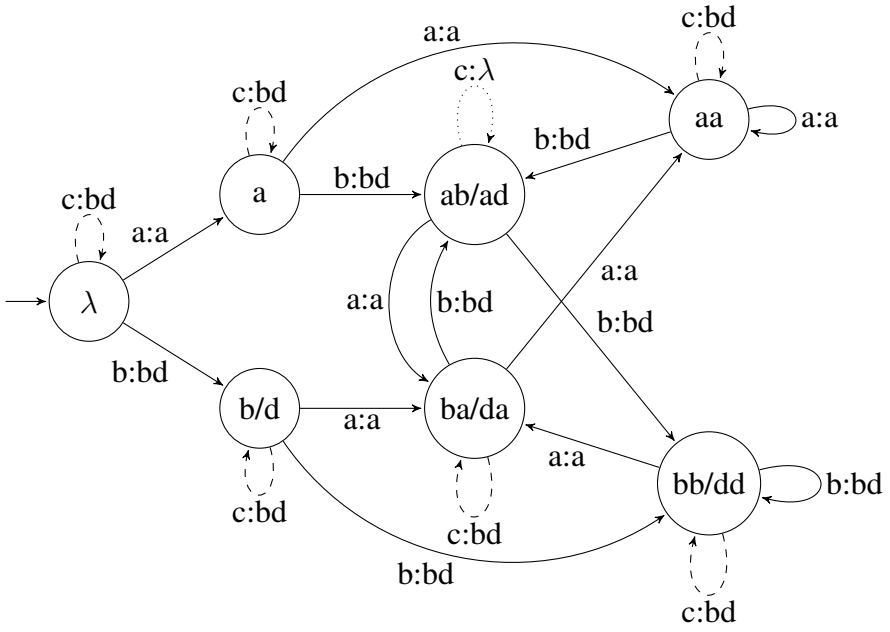


Figure 7.2: A function that is OTSL$_3$ for $T_1 = \{\text{a, b}\}$ and $T_2 = \{\text{a, d}\}$ but not $\Omega = T_1 \cup T_2$.

in our incorrect tier hypothesis must be removed therefrom. Specifically, we can find two input strings whose outputs share a common 1-length suffix on the hypothesized tier, but which have different tails relative to the function (because they have different suffixes on the canonical tier). For example, in the function represented by Figure 7.1, the input '⋊so' has the tail (ʃ, s) while the input '⋊ʃo' has the tail (ʃ, ʃ), despite the fact that the outputs of these two inputs have the same 1-suffix 'o'.

Putting the two properties together implies a method for discovering the canonical tier of an OTSL$_2$ function: start by hypothesizing that the entire output alphabet is on the tier, and remove elements as you find evidence that they need to be removed. At any given point in time following this strategy, we either have the canonical tier (in which case none of our training data favour the removal of a tier element), or have a strict superset thereof (in which case our training data contains evidence in favour of removing at least one tier element). When we've whittled our hypothesis down to the canonical tier, we will be able to mark all of its contents as "safe".

The first step in implementing this strategy is to gain as much information as possible about the prefix function f^p corresponding to f, based only on the evidence provided in the training sample. Burness & McMullin (2019) devised a method of doing so whose worst-case run time is in $\mathcal{O}(m^4)$, where m is the size of the training sample; I present an alternative method here whose worst-case run time is in $\mathcal{O}(m^2)$ and which also allows us to greatly improve the efficiency of later learning steps. We start with what is known as a Prefix Tree Transducer (PTT), defined in Definition 7.6 which is adapted from Chandlee et al. (2014). Note that $\texttt{pref}^*(w)$ denotes all prefixes of w. The

PTT corresponding to a sample of input-output pairs effectively generates all and only the pairs in the sample, writing the entire output in one fell swoop after reading the entire input. For example, the PTT corresponding to {(s, s), (ss, ss), (sʃ, ss), (so, so), (sos, sos), (soʃo, soso), (sooS, soos)} is shown in Figure 7.3. To avoid visual clutter, all transitions landing in q_f are incorporated into the label of their origin state.

Definition 7.6 Prefix Tree Transducer (adapted from Chandlee et al. 2014)**.**
A Prefix Tree Transducer (PTT) for the finite set D of pairs (w, w') from some function f is $PTT(D) = (Q, q_0, q_f, \Sigma, \Delta, \delta)$ where:

- $Q = \bigcup_{(w,w')\in D}\{\texttt{pref}^*(w)\}$
- $(\forall u \in \Sigma^*)(\forall a \in \Sigma)$
  $[u, ua \in Q \iff (u, a, \lambda, ua) \in \delta]$
- $(w, w') \in D \iff (w, \ltimes, w', q_f) \in \delta$
- $(q_0, \rtimes, \lambda, \lambda) \in \delta$

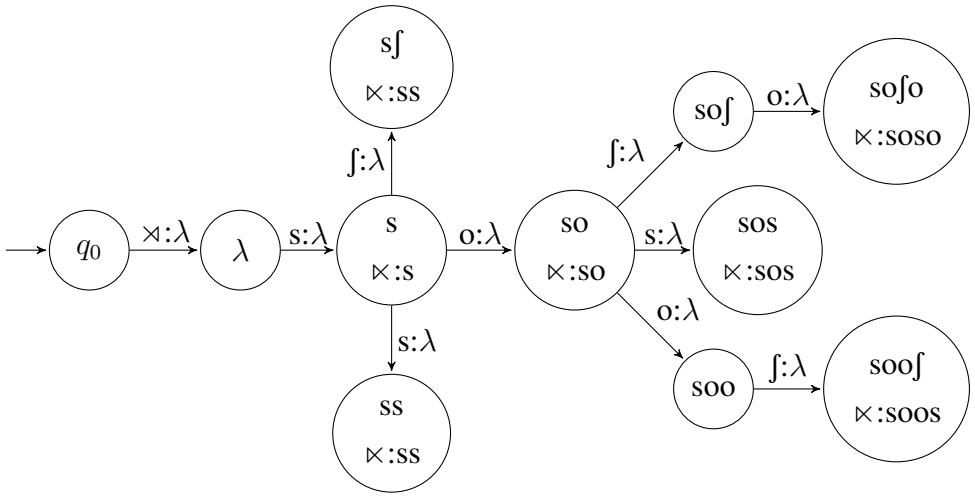


Figure 7.3: The Prefix Tree Transducer for the sample {(s, s), (ss, ss), (sʃ, ss), (so, so), (sos, sos), (soʃo, soso), (sooS, soos)}

In this initial form, a PTT is maximally lazy, waiting as late as possible before writing any output. For tier learning, we need to modify the PTT so that it is *minimally* lazy, producing as much output as it can as early as it can. To perform such a conversion, we can perform a depth-first parse of the sample working backwards from the leaves of the prefix tree towards the root. For each state along the way, we calculate the longest common prefix of the output edges on its outgoing transitions, pushing that string onto the output edge of its lone incoming transition (de la Higuera,

2010, pp. 377-379). Since they act analogously to onward FSTs, I use the term onward PTT to refer to such a converted PTT and write $\texttt{onward}(M)$ to denote the process being applied to M. For the full details of how to build a PTT and make it onward, see chapter 18 of de la Higuera (2010). Figure 7.4 shows the result of converting the PTT in Figure 7.3. Note especially that the very first transition produes 's' right away since the longest common prefix of all outputs is [s]. Similarly, the machine will anticipate reading /o/ if it reads /ʃ/ while in the /so/ state since the only input in the sample with /soʃ/ as a prefix is /soʃo/.

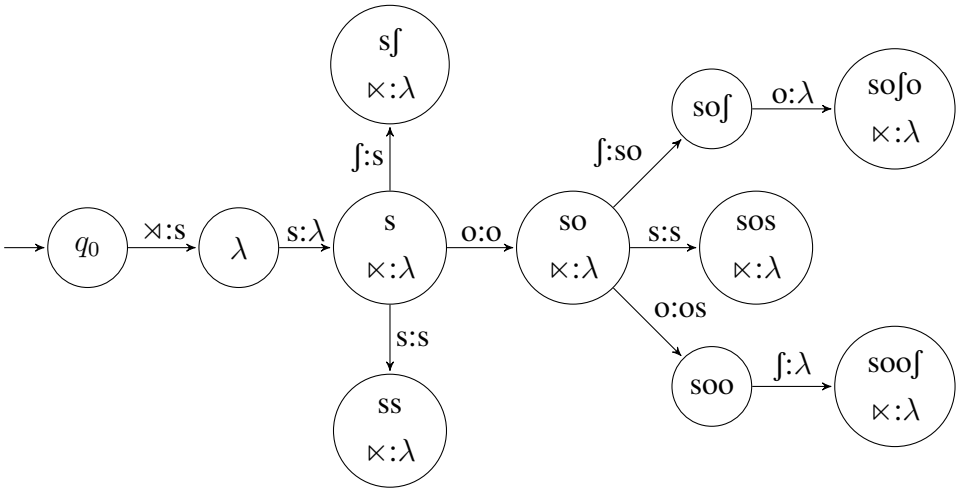


Figure 7.4: The onward version of Figure 7.3

A PTT that has been made onward exhibits some important properties that will be exploited below. First, given a state $q \in Q$ that has an outgoing transition for all $x \in \Sigma \cup \{\ltimes\}$, we will have produced exactly $f^p(q)$ so far when we enter the state q. I call such a state a *supported* state. Assuming that $\Sigma = \{\text{s, o, ʃ }\}$, the supported states in Figure 7.4 are 's' and 'so'. The procedure `estimate_fp` shown in Algorithm 2 sends a sample S once through its onward PTT and returns all pairs $(q, f^p(q))$ such that q is a supported state. The other crucial property of an onward PTT follows from the first: given a transition $(q, x, y, q') \in \delta$ such that q is a supported state and q' is either another supported state or q_f, the string y is guaranteed to be equal to $\texttt{cont}_f(x, q)$ provided that f is a subsequential function. The transitions meeting these criteria in Figure 7.4 are $(\text{s}, \ltimes, \lambda, q_f)$, (s, o, o, so) and $(\text{so}, \ltimes, \lambda, q_f)$. Important to note is that when there is no transition with the shape (q, x, y, q'), we do not know whether this is accidental (i.e., due to the finite nature of the training data) or because the function is undefined for all inputs starting with qx (i.e., because the function is *partial*). While the ability to accommodate partial functions would have practical applications for learning from natural language data, at present I leave the task of learning partial (M)TSL functions to future research, focusing only on total (M)TSL functions henceforth.

The full tier induction process is shown in Algorithm 3. This algorithm adapts the overall

Data: A sample $S \subset \Sigma^* \times \Delta^*$
Result: An onward PTT P and the set A containing $(q, f^p(q))$ for each supported state q
Function `estimate_fp(S)` **:**

$P \leftarrow PTT(S) = (Q, q_0, q_f, \Sigma, \Delta, \delta)$;
$P \leftarrow \texttt{onward}(P)$;
$A \leftarrow \emptyset$;
$B \leftarrow \emptyset$;
for $q \in Q$ **do**
  if $(\forall x \in \Sigma \cup \{\ltimes\})[\exists (q, x, y, q') \in \delta]$ **then**
    $B \leftarrow B \cup \{q\}$
for *each* $(x, y) \in S$ **do**
  $\texttt{out} \leftarrow z$ such that $(q_0, \rtimes, z, \lambda) \in \delta$;
  $\texttt{leave} \leftarrow \lambda$;
  if $\texttt{leave} \in B$ **then**
    $A \leftarrow A \cup \{(\texttt{leave}, \texttt{out})\}$;
  for n *from* 1 *to* $|x|$ **do**
    $\texttt{read} \leftarrow$ the n-th letter of x;
    $\texttt{write} \leftarrow u$ such that $(\texttt{leave}, \texttt{read}, u, q) \in \delta$;
    $\texttt{land} \leftarrow q$ such that$(\texttt{leave}, \texttt{read}, \texttt{write}, q) \in \delta$;
    $\texttt{out} \leftarrow \texttt{out} \cdot \texttt{write}$;
    $\texttt{leave} \leftarrow \texttt{land}$;
    if $\texttt{leave} \in B$ **then**
      $A \leftarrow A \cup \{(\texttt{leave}, \texttt{out})\}$;
return P, A

Algorithm 2: Prefix function estimation

strategy from Burness & McMullin (2019) so that it can be used with the PTT objects described above. Adapting the strategy in this way drastically improves time efficiency; I formally prove in Appendix B that the run time of the modified tier induction process is in $\mathcal{O}(m^2)$ relative to the size of the sample. The run time of the original implementation from Burness & McMullin (2019) is in $\mathcal{O}(m^5)$ relative to the size of the sample, which is efficient by theoretical standards, but not necessarily by practical standards. Indeed, it is worse than the $\mathcal{O}(m^3)$ complexity of the Onward Subsequential Transducer Inference Algorithm (OSTIA: Oncina et al., 1993) which can learn the superset class of subsequential functions. Reducing the time complexity of tier-learning this much is a substantial contribution in its own right, since it gives even more reason to prefer a TSL function over a subsequential function when modeling a long-distance phonological pattern.

Given a finite sample of training data, the tier-learning algorithm first estimates the relevant

prefix function using an onward PTT as described above. Following that, it hypothesizes that $T = \Delta$ (i.e., that all members of the output alphabet are on the tier). Then, for each transition (q, x, y, r) in the onward PTT, the algorithm checks whether q is a supported state and whether r is a supported state or q_f. If both of these conditions hold, then the output edge y of the transition is equal to $\texttt{cont}_f(x, q)$. Accordingly, the algorithm takes $(q, f^p(q)) \in A$, calculates $t = \texttt{suff}^1_T(f^p(q))$, and adds y to the set $C_{x,t}$ if it is not already there. If t is on the function's canonical tier, the cardinality of this set should always be equal to or less than 1 since the target function is OTSL_2 and so the contribution should be the same whenever the output tier suffix is equal to t.

After scanning through all transitions in the onward PTT, the algorithm looks for any constructed set of contributions $C_{x,t}$ with cardinality greater than 1. If none of the sets associated with a specific $t \in T$ have cardinality greater than 1, the element t will get added to the auxiliary set `keep` containing elements that are safe to keep on the tier for now. If any of the sets associated to $t \in T$ have cardinality greater than 1, the element t is removed from the tier hypothesis since it cannot possibly be a member of the function's canonical tier. If at any point some symbol gets removed from T, the set `keep` is immediately emptied. The algorithm repeatedly alternates between scanning the PTT transitions and checking the cardinality of contribution sets until every t in the current hypothesis for T gets added to the set `keep`, in which case it knows it has found the canonical tier of the target function. The sample and this tier can then be fed to the transducer building algorithm from Section 7.2.

Like I did for the transducer-building algorithm in Section 7.2, I will demonstrate how Algorithm 3 learns the tier of the function represented in Figure 7.1. It would be cumbersome and not particularly informative to present the core data from this function required for discovering the tier, so I simply note that such a characteristic sample is contained in the set $S = \{(w, f(w)) \mid w \in \Sigma^{\leq 4}\}$, or the set of all input-output mappings for input strings of up to length 4. This training set allows us to confidently determine $f^p(w)$ for all w of up to length 3, which is sufficient to discover the tier of this function.

We begin tier learning by hypothesizing that T is equal to the set {o, s, ʃ}. Now we cycle through the transitions in the onward PTT, and whenever we find one that starts in a supported state and ends in a supported state or q_f, we place its output edge into the contribution set simultaneously associated to its input edge and the suffix of its origin's label on the current tier hypothesis. Going through this process will create the 9 sets $C_{\text{s, s}} = \{\text{s}\}$, $C_{\text{o, s}} = \{\text{o}\}$, $C_{\text{ʃ, s}} = \{\text{s}\}$, $C_{\text{s, ʃ}} = \{\text{s}\}$, $C_{\text{o, ʃ}} = \{\text{o}\}$, $C_{\text{ʃ, ʃ}} = \{\text{ʃ}\}$, $C_{\text{s, o}} = \{\text{s}\}$, $C_{\text{o, o}} = \{\text{o}\}$, $C_{\text{ʃ, o}} = \{\text{s, ʃ}\}$. All sets associated with the hypothesized tier elements 's' and 'ʃ' have a cardinality of 1, so 's' and 'ʃ' can be placed in `keep`. The last set $C_{\text{ʃ, o}}$, however, has a cardinality of 2 and is associated with the hypothesized tier element 'o', telling us that we need to remove 'o' from the tier.

Upon removing 'o', it just so happens that we have the correct tier, but since there is no means

Data: A sample $S \subset \Sigma^* \times \Delta^*$
Result: A tier $T \subseteq \Delta$
Function `get_tier(`S`)`**:**
    $P, A \leftarrow$ `estimate_fp`(S);
    $T \leftarrow \Delta$;
    `keep` $\leftarrow \emptyset$;
    while `keep` $\neq T$ **do**
        for *each* $t \in T$ **do**
            for *each* $x \in \Sigma \cup \{\ltimes\}$ **do**
                $C_{x,t} = \emptyset$;
        for *each* $(q, x, y, r) \in \delta$ *from* P **do**
            if $[\exists (q, a) \in A] \wedge [[r = q_f] \vee [\exists (r, b) \in A]]$ **then**
                $t \leftarrow \mathtt{suff}^1_T(a)$;
                $C_{x,t} \leftarrow C_{x,t} \cup \{y\}$;
        for *each* $t \in T$ **do**
            for *each* $x \in \Sigma \cup \{\ltimes\}$ **do**
                if $|C_{x,t}| > 1$ **then**
                    $T \leftarrow T - \{t\}$;
                    `keep` $\leftarrow \emptyset$;
            if $t \in T$ **then**
                `keep` $\leftarrow$ `keep` $\cup \{t\}$
    return T

Algorithm 3: Single tier induction

of knowing exactly how many times we must verify the contents of T, we must reconsider the safe status of all the elements we have verified up until this point. Cycling again through the transitions of the onward PTT will construct the 6 sets $C_{\mathrm{s},\,\mathrm{s}} = \{\mathrm{s}\}$, $C_{\mathrm{o},\,\mathrm{s}} = \{\mathrm{o}\}$, $C_{\int,\,\mathrm{s}} = \{\mathrm{s}\}$, $C_{\mathrm{s},\,\int} = \{\mathrm{s}\}$, $C_{\mathrm{o},\,\int} = \{\mathrm{o}\}$, and $C_{\int,\,\int} = \{\int\}$. All of these have a cardinality of 1 meaning that both of the current hypothesized tier elements get added to `keep`. As a result, `keep` becomes equal to T and we stop.

It is not necessarily true that a single run through the contents of Δ will be able to find and flag all elements that are not in the canonical tier. This is precisely why `get_tier` is designed to reconsider the the safe status of elements as soon as anything is removed from the tier hypothesis. Burness & McMullin (2019) show that their implementation of the contribution-checking strategy is guaranteed to learn the canonical tier of any total OTSL_2 function, provided that the learner sees at least a core sample of representative input-output pairs. They furthermore show that their implementation has a polynomial bound on run time and a polynomial bound on the amount of required data. The same statements hold for the implementation shown in this section; in fact, the present

implementation represents a marked improvement in time efficiency over Burness & McMullin's (2019) original implementation. Proofs of the current algorithm's convergence and efficiency are included in Appendix B for interested readers. The remainder of this chapter will show how the method can be modified to work for the strongly and weakly target-specified OMTSL_2 functions defined and explored in Chapter 4.

7.4 Learning multiple tiers

The difficulty posed by multi-tiered functions is the fact that their multiple tiers *jointly* determine the function's tails. In order to ascertain whether a symbol belongs to one tier, then, we need to make sure the content on all other tiers remains constant. This aspect of the OMTSL_2 class traps us in a sort of chicken-and-egg problem, and consequently, I do not believe that it is possible to learn any arbitrary collection of tiers efficiently from positive data alone. Rather than give up, however, we might ask whether there are any structured subclasses of the OMTSL_2 functions that escape the problematic interlocking uncertainties. I will now show that the notion of target-specification from Chapter 4 imparts a structure onto the hypothesis space that a learner can use to great effect. Section 7.4.1 discusses how the strong version of target specification permits (with minor modifications) the same learning method as above with guaranteed results. Section 7.4.2 considers the weak version of target specification and proposes an alternative learning method which does not yet have accompanying formal proofs, but succeeds on some test cases and can interact with phonological substance in interesting ways.

7.4.1 When they are independent (strongly target-specified tiers)

Consider the crucial properties that permitted efficient learning of a tier for any OTSL_2 function. These were (i) that each OTSL_2 function has a unique *canonical* tier which contains all of the function's other possible tiers and (ii) that there will always be positive evidence from an OTSL_2 function in favour of removing at least one superfluous element if we try using a superset of that function's canonical tier. Now recall that when a function is strongly target specified, each input element $x \in \Sigma \cup \{\ltimes\}$ can be associated to a single tier which on its own can always correctly determine the contribution of x. Let us call such a tier an x-tier. Importantly, a y-tier for $y \in \Sigma \cup \{\ltimes\}$ will operate independently from a z-tier for a different $z \in \Sigma \cup \{\ltimes\}$. As a consequence of this independence, strongly target-specified OMTSL_2 functions have properties directly parallel to the ones that were exploited when learning a tier for a given OTSL_2 function. First, each strongly target-specified OMTSL_2 function has a unique *canonical* x-tier for $x \in \Sigma \cup \{\ltimes\}$ which contains all of the function's other possible x-tiers. Second, there will always be positive evidence from a

strongly target-specified OMTSL$_2$ function in favour of removing at least one superfluous element if we try using a superset of that function's canonical x-tier. Formal statements and proofs of these properties are in Appendix B.

The fact that the basis for the OTSL$_2$ tier learning strategy transfers neatly over to the strongly target specified OMTSL$_2$ functions implies that the tier learning strategy itself can also be transferred over. When looking for the canonical tier of an OTSL$_2$ function, we decide whether or not to remove some element a from the current tier hypothesis H by checking whether or not the contributions of all input elements are stable wherever a is the suffix on the projection imposed by H. If we start by hypothesizing that everything is on the canonical tier, this strategy will eventually whittle the hypothesis down to the actual canonical tier. Learning all the canonical tiers of a strongly target-specified OMTSL$_2$ requires only two relatively minor modifications. First, when looking for the canonical x-tier associated to $x \in \Sigma \cup \{\ltimes\}$, we check the stability of only the contributions of x rather than the stability of *all* input elements. Second, we implement the "whittling down" procedure once per input element. Algorithm 4 presents a modification of Algorithm 3 that learns the lone tier specified by a given input element x in a strongly target-specified OMTSL$_2$ function. Running this algorithm once for each member of $\Sigma \cup \{\ltimes\}$ would give us the information (i.e., the tiers) necessary for building a transducer that computes the target function. Proofs of the algorithm's efficiency and guaranteed success are given in Appendix B.

7.4.2 When they interact (weakly target-specified tiers)

We face some serious difficulties if we want to extend our coverage from the *strongly* target-specified OMTSL$_2$ functions to the *weakly* target-speficied OMTSL$_2$ functions. In order for the above method to succeed we need a unique target tier, but our learning target now consists of potentially multiple tiers per input element. Crucially, weak target specification requires that the multiple tiers associated to a given input element fall into a strict subset-superset hierarchy. In light of this definitional requirement, I see a potential approach to learning the required superset-subset hierarchy of tiers, although I cannot yet provide any proof of its convergence.

To simplify exposition in this section, I have the learner presented with a ready-made list of what I call *contribution triples*. Each of these consists of (1) the contributing input element, (2) the contributed output string, and (3) the preceding output string. Deriving such information from the result A of `estimate_fp` is not too difficult, and a process for doing so is shown in Algorithm 5. We take every transition (q, x, y, r) in the onward PTT $M = (Q, q_0, q_f, \Sigma, \Delta, \delta)$, checking whether q is supported and whether r is supported or equal to q_f. If so, we store the triple $(/x/, [y],$ 'z'$)$ where z is such that $(q, z) \in A$. For example, given (sa, ʃ, s, saʃ) $\in \delta$, given (sa, sa) $\in A$ and given (saʃ, sas) $\in A$ we can create a triple (/ʃ/, [s], 'sa') which records the fact that an instance of /ʃ/ was

Data: A sample $S \subset \Sigma^* \times \Delta^*$ and an element $x \in \Sigma \cup \{\ltimes\}$
Result: A tier $T_x \subseteq \Delta$
Function `get_x_tier(S, x)`**:**

```
P, A ← estimate_fp(S);
T_x ← Δ;
keep ← ∅;
while keep ≠ T_x do
    for each t ∈ T_x do
        C_t ← ∅;
    for each (q, y, z, r) ∈ δ from P do
        if [y = x] ∧ [∃(q, a) ∈ A] ∧ [[r = q_f] ∨ [∃(r, b) ∈ A]] then
            t ← suff^1_{T_x}(a);
            C_t ← C_t ∪ {z};
    for each t ∈ T_x do
        if |C_t| > 1 then
            T_x ← T_x − {t};
            keep ← ∅;
        if t ∈ T_x then
            keep ← keep ∪ {t}
return T_x
```

Algorithm 4: Strongly target-specified tier induction

mapped to [s] when the preceding output was 'sa'.

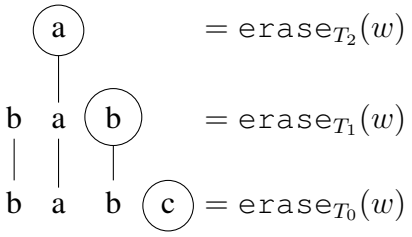
Let the term "x-tierset" denote a superset-subset hierarchy of tiers associated to the input element $x \in \Sigma \cup \{\ltimes\}$ and let the term "x-member" denote a single tier from an x-tierset. The approach I have in mind is based on the intuition that if the superset-subset hierarchy is optimally constructed, then the suffix on each x-member should alter the output behaviour of x significantly *after controlling for the suffixes on the larger x-members in the same x-tierset*. To see how output behaviour can be quantified, consider the fact that each input element $x \in \Sigma \cup \{\ltimes\}$ has a finite number of mappings that it can undergo. For example in a basic sibilant harmony system the mappings available to /s/ are /s/→[s] and /s/→[ʃ]. When viewed at the level of a sample, this finite list of possible outcomes will define a multinomial distribution where the probability of $/x/ \rightarrow [y]$ is the number of times that mapping occurs in the sample divided by the total number of times $/x/$ appears in the sample's input strings. By seeing how the distribution changes under different sampling criteria, we can progressively build a set of tiers that act together to predict what would happen to an instance of $/x/$ at a given step during a transduction.

Data: A sample $S \subset \Sigma^* \times \Delta^*$
Result: A set $B \subset \Sigma \cup \{\ltimes\} \times \Delta^* \times \Delta^*$ of contribution triples
Function `extract_conts(`S`)`**:**
    $P, A \leftarrow$ `estimate_fp`(S);
    $B \leftarrow \emptyset$;
    for *each* $(q, x, y, r) \in \delta$ *from* P **do**
        if $[\exists(q, a) \in A] \wedge [[r = q_f] \vee [\exists(r, b) \in A]]$ **then**
            $B \leftarrow B \cup \{(/x/, [y], \text{`a'})\}$;
    return B

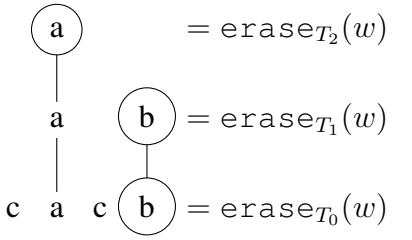
Algorithm 5: Contribution extraction

Before describing the sampling criteria I have in mind, some additional terminology is necessary. The following are all defined relative to an x-tierset such that $T_i \subset T_{i+1}$ and where T_0 is the full alphabet. A string w exhibits the *context* $\langle o_0 \in T_0, ..., o_n \in T_n \rangle$ if and only if $\texttt{suff}^1_{T_i}(w) = o_i$ for each o_i in the context. For example, given the tiers $T_0 = \{a, b, c\}$, $T_1 = \{a, b\}$, and $T_2 = \{a\}$, the string $babc$ exhibits the context $\langle c, b, a \rangle$ and the string $cacb$ exhibits the context $\langle b, b, a \rangle$. These contexts are depicted as the circled nodes in (72) and (73). Furthermore, let $\texttt{peak}_T(w)$ be the position in string w that is $\texttt{suff}^1_{T_i}$ (its *peak* relative to the tier T_i). For example, once again given the tiers $T_0 = \{a, b, c\}$, $T_1 = \{a, b\}$, and $T_2 = \{a\}$ we have $\texttt{peak}_{T_0}(babc) = 4$, $\texttt{peak}_{T_1}(babc) = 3$, and $\texttt{peak}_{T_2}(babc) = 2$. Finally, where $\texttt{peak}_T(w) = n$, let $\texttt{pre\_peak}_T(w) = \texttt{pref}^{n-1}(w)$ be the *pre-peak string* of w relative of to T. For example, once again given the tiers $T_0 = \{a, b, c\}$, $T_1 = \{a, b\}$, and $T_2 = \{a\}$ we have $\texttt{pre\_peak}_{T_0}(babc) = bab$, $\texttt{pre\_peak}_{T_1}(babc) = ba$, and $\texttt{pre\_peak}_{T_2}(babc) = b$.

(72) Context $\langle c, b, a \rangle$ exhibited by $w = babc$ given $T_0 = \{a, b, c\}$, $T_1 = \{a, b\}$, and $T_2 = \{a\}$



(73) Context $\langle b, b, a \rangle$ exhibited by $w = cacb$ given $T_0 = \{a, b, c\}$, $T_1 = \{a, b\}$, and $T_2 = \{a\}$



Given the above terminology, I propose the following procedure for learning the tiers of a weakly target-restricted OMTSL$_2$ function. First, when attempting to build the j-th x-member T_j (with $j \geq 1$), we start by hypothesizing that it is equal to the next-biggest x-member T_{j-1}, with T_0 being the full alphabet. Now we pick one element a from the current hypothesis for T_j and begin testing whether it needs to be removed. To do so we take a context C built from the x-tierset T_0 through T_{j-1}. Next we collect each triple $(/x/, [y], \text{`}z\text{'})$ such that '$z$' exhibits the context $C$, placing these into a set `reference`. Then we build a separate set `inspection` by collecting each triple `reference` such that the element $a$ being tested is $\texttt{suff}^1_{T_j}(\texttt{pre\_peak}_{T_{j-1}}(\text{`z'}))$. Following that, we calculate and compare the distributions of $/x/ \rightarrow [y]$ mappings in `reference` and `inspection`. If the distributions significantly differ (e.g., $p < 0.01$ on a chi-squared goodness of fit test), then a significantly alters the output behaviour of x beyond the effects of the tiers T_0 through T_{j-1}, and therefore should be included on T_j. If however for all possible contexts C, the distribution in `inspection` does *not* differ from the distribution in `reference`, then a does not significantly alter the output behaviour of x beyond the effects of the tiers T_0 through T_{j-1}, and therefore should not be included on T_j. We can stop attempting to build further x-members if T_j is reduced to the empty set, whereupon we conclude that T_0 through T_{j-1} are the necessary tiers. Pseudo-code for this procedure is given in Algorithm 6. The pseudo-code also stops if the same tier is built twice in a row, since in testing (discussed below) there were instances where the algorithm would keep building and adding the same tier forever.

To test the suitability of this learning strategy, I created 5 test cases that are each based on a pattern analyzed in Chapter 4. These are (i) the overriding of local palatalization by non-local sibilant harmony in Samala, (ii) the overriding of non-local labial dissimilation by local place assimilation in Tamashek Tuareg, (iii) the blocking of sibilant voicing but not sibilant anteriority harmony by voiceless consonants in Imdlawn Tashlilhiyt, (iv) split rounding and backness vowel harmony in Turkish, and (v) trigger preference in Uyghur backness vowel harmony. A learning sample of contribution triples was constructed for each pattern by building a subsequential transducer computing that pattern, running every possible input of up to length 3 through that transducer to obtain its associated output string 'z', and identifying the contribution for each input element out of its landing state (i.e., the input edge /x/ and output edge [y] of the corresponding transition). The statistical comparison step of the algorithm performed a chi-squared goodness of fit test and the significance threshold was initially set to $p < 0.01$. All of these learning simulations were implemented using custom Python code.

The toy Samala pattern operated over the alphabets $\Sigma = \Delta = \{$i, e, a, o, u, p, t, k, m, n, ŋ, s, ʃ$\}$ and applied progressively (i.e., it is a mirror image of the real Samala pattern). According to the toy pattern's sibilant harmony rule, input /s/ was mapped to [ʃ] if it was preceded by [ʃ] at any distance and input /ʃ/ was mapped to [s] if it was preceded by [s] at any distance. According to the

Data: A sample S of contribution triples $(/u/, [v], \text{`}w\text{'}) \in \Sigma \cup \{\ltimes\} \times \Delta^* \times \Delta^*$
Result: A superset-subset hierarchy Θ_x of tiers for a given $x \in \Sigma \cup \{\ltimes\}$
Function `get_x_tierset` (*S, x*) **:**

$\Theta_x \leftarrow \{T_0 = \Delta\}$
`stop` $\leftarrow 0$
$j \leftarrow 0$
while $stop = 0$ **do**
  $j \leftarrow j + 1$
  $T_j \leftarrow T_{j-1}$
  `keep` $\leftarrow \emptyset$
  while $keep \neq T_j$ **do**
    for *each* $a \in T_j$ **do**
      for *each context* $C = \langle c_0, ..., c_{j-1} \rangle \in T_0 \times ... \times T_{j-1}$ **do**
        `reference` $\leftarrow \{(/u/, [v], \text{`}w\text{'}) \in S \mid u = x \wedge$
          $(\forall n \leq j-1)[\texttt{suff}^1_{T_n}(w) = c_n]\}$
        `inspection` $\leftarrow \{(/u/, [v], \text{`}w\text{'}) \in$ `reference` $\mid$
          $\texttt{suff}^1_{T_j}(\texttt{pre\_peak}_{T_{j-1}}(w)) = a\}$
        if $reference$ *and* $inspection$ *significantly differ* **then**
          `keep` $\leftarrow$ `keep` $\cup \{a\}$
      if $a \notin keep$ **then**
        $T_j \leftarrow T_j - \{a\}$
        `keep` $\leftarrow \emptyset$
  if $[T_j = \emptyset] \vee [keep = T_{j-1}]$ **then**
    `stop` $= 1$
  else
    $\Theta_x \leftarrow \Theta_x \cup \{T_j\}$
return Θ_x

Algorithm 6: Weakly target-specified tierset induction

toy pattern's rule of local palatalization, input /s/ was mapped to [ʃ] when immediately preceded by [t] or by [n] unless this would violate the rule of sibilant harmony. Input elements surfaced faithfully wherever neither of these rules apply. With the significance threshold at $p < 0.01$, the algorithm returned the tiers $T_0 = \Delta$ and $T_1 = \{\text{s}, \text{ʃ}\}$ for the input element /s/. The algorithm thus successfully discovered that sibilant consonants exert an influence on /s/ that goes above and beyond other purely local effects. For instance, of the 183 triples in the sample whose preceding output ended in 'a', 27 triples or 14.75% give [ʃ] for /s/ and 156 triples or 85.25% give [s] for /s/, but 16 out of 16 triples whose preceding output ended in 'ʃa' give [ʃ] for /s/. Comparing the observed values of 16 [ʃ] and 0 [s] to the expected values of 2.36 [ʃ] and 13.64 [s] gives us $\chi^2 = 92.475$ on

1 degree of freedom, which is highly significant ($p = 6.818 \times 10^{-22}$).

The algorithm equally succeeded in learning a toy version of split rounding and backness harmony in Turkish. For this case, the alphabets were $\Delta = \{$i, y, ɯ, u e, ø, a, o, p, t, k, kʲ, b, d, g, gʲ$\}$ and $\Sigma = \Delta \cup \{$I$\}$ where /I/ is a high vowel that is unspecified for rounding and backness. Input /I/ could take on any combination of values for [±round] and [±back] to become one of [i], [y], [ɯ], or [u]. The value for rounding was taken from the closest preceding element from the set $\{$i, y, ɯ, u e, ø, a, o$\}$ and the value for backness was taken from the closest preceding element from the set $\{$i, y, ɯ, u e, ø, a, o, k, kʲ, g, gʲ$\}$. With the significance threshold at $p < 0.01$, the algorithm returned the tiers $T_0 = \Delta$, $T_1 = \{$i, y, ɯ, u e, ø, a, o, k, kʲ, g, gʲ$\}$, and $T_2 = \{$i, y, ɯ, u e, ø, a, o$\}$ for the input element /I/. This result suggests that the algorithm discovered that all vowels plus the velar consonants could exert a non-local influence on /I/, and furthermore discovered that vowels could exert an even less local influence on /I/ across velar consonants.

In the remaining test cases, the algorithm did not succeed given a significance threshold of $p < 0.01$ and the failures can be divided into two types. On the one hand are those where the algorithm failed to keep something necessary on a tier. Imdlawn Tashlhiyt and Uyghur fall into this category. On the other hand are those where the algorithm failed to remove something unnecessary from a tier. Tamashek Tuareg falls into this category. Interestingly, the algorithm could be rendered successful (or at the very least *more* successful) in each case by adjusting the significance threshold. I will individually discuss each failure below, but summarizing the results of these case studies, it would seem that an overly stringent significance threshold can lead to elements getting incorrectly removed from a tier and an insufficiently stringent significance threshold can lead to elements getting incorrectly kept on a tier. A possible means to prevent these issues would be to vary the significance threshold according to the similarity between the element specifying the tier and the element whose tier membership is being scrutinized. Tiers seem generally to be made of similar elements, so with an appropriate means of quantifying similarity, one could loosen the significance threshold as similarity becomes stronger and tighten the significance threshold as similarity becomes weaker. In the unmarked case, this would result in tiers that most phonologists would deem *natural*, though more exotic tiers would be possible given appropriate data. Support for the idea that similarity factors into tierset learning comes from psycholinguistic experiments showing that dependencies between non-adjacent sounds are easier to learn when the interacting sounds are more similar to one another (e.g., Creel et al., 2004; Newport & Aslin, 2004; Gebhart et al., 2009). Similarity is not essential for tierset learning, however: experiments by Koo & Callahan (2012) and Koo & Oh (2013) show that humans are able to passively learn patterns that operate over arbitrary tiers.

The toy Uyghur pattern operated over the alphabets $\Delta = \{$i, y, u, e, ø, o, æ, a, p, t, k, q, b, d, g, ʁ$\}$ and $\Sigma = \Delta \cup \{$A$\}$ where /A/ is a low vowel that is unspecified for backness. This

unspecified vowel takes on the backness of the most recent vowel from the set {y, u, ø, o, æ, a} if one is available, else it will surface as front if the most recent dorsal consonant is velar, else it will surface as back by default. With the significance threshold at $p < 0.01$, the algorithm returned the tiers $T_0 = \Delta$, $T_1 = \{$y, u, ø, o, æ, a, k, g, ʁ$\}$, and $T_2 = \{$y, u, ø, o, æ, a$\}$ for the input element /A/. The smallest tier is correct, but the intermediate tier is missing the voiceless uvular consonant [q]. Loosening the threshold to $p < 0.05$ we instead get the tiers $T_0 = \Delta$, $T_1 = \{$y, u, ø, o, æ, a, k, q, g, ʁ$\}$, $T_2 = \{$y, u, ø, o, æ, a, ʁ$\}$, and $T_3 = \{$y, u, ø, o, æ, a$\}$. Combining the influence of tiers T_1 and T_3 would correctly partition inputs into tail-equivalence classes, and while T_2 is not necessary, it is also not detrimental. The fact that we succeed only under a weaker significance threshold suggests that the influence of dorsal consonants on unspecified /A/ is not particularly robust statistically, even in an idealized data set. This is unsurprising considering that dorsal consonants only affect /A/ in the complete absence of any non-transparent vowel.

As I stated earlier, a similar state of affairs held for the toy Imdlawn Tashlhiyt pattern which operated over the alphabets $\Delta = \Sigma = \{$i, u, a, e, o, p, t, k, b, d, g, s, ʃ, z, ʒ$\}$. Like for the toy Samala pattern, the toy Imdlawn Tashlhiyt pattern is a progressive mirror image of the real regressive pattern. Input sibilants took on the anteriority and voicing of the most recent preceding sibilant unless a voiceless consonant intervened, in which case they took on the preceding sibilant's anteriority but maintained their own voicing. With the significance threshold at $p < 0.01$, the algorithm returned $T_0 = \Delta$ and $T_1 = \{$s, ʃ, z, ʒ$\}$ for /s/. Once again, the smallest tier is correct, but we are missing an intermediate tier that also contains the voiceless obstruents. Loosening the threshold to $p < 0.05$ we instead get $T_0 = \Delta$, $T_1 = \{$s, ʃ, z, ʒ, k$\}$, and $T_2 = \{$s, ʃ, z, ʒ$\}$, which is still incorrect but is closer to the desired hierarchy wherein T_1 would additionally contain [p] and [t]. The fact that we only partially succeed under a weaker significance threshold suggests that the influence of voiceless non-sibilants on sibilants is statistically very weak, even in an idealized data set. For its part, the influence of sibilants on other sibilants seems particularly robust and easy to discover.

Our final test case is the toy Tamashek Tuareg pattern which operated over the alphabets $\Delta = \Sigma = \{$i, u, a, e, o, p, t, k, b, d, g, m, n, ŋ$\}$. Here as well, the toy Tamashek Tuareg pattern is a progressive mirror image of the real regressive pattern. In this pattern, input /m/ would dissimilate to [n] if preceded non-locally by either [p], [b], or [m]. The pattern contained an additional requirement of local place assimilation between a nasal consonant and a preceding stop, and this local assimilation took precedence over dissimilation. With the significance threshold at $p < 0.01$, the algorithm returned $T_0 = \Delta$ and $T_1 = \{$p, b, m, n, ŋ$\}$ for /m/. As I mentioned earlier, this is a failure opposite in flavour to the previous two: we should have eliminated [n] and [ŋ] from the tier but didn't. Tightening the threshold to $p < 0.001$ we instead get $T_0 = \Delta$ and $T_1 = \{$p, b, m, ŋ$\}$ which is closer to the desired $T_1 = \{$p, b, m$\}$. The fact that we only partially succeed under

a stronger significance threshold suggests that the non-local influence of labial consonants on /m/ is statistically strong, but nonetheless hard to tease apart from other factors.

We have seen that my proposed method's reliance on statistical comparisons makes it very susceptible to the quality of the data set it is working off of. Despite these issues, however, it succeeds reasonably well for 3 out of the 5 test cases, and comes close to the correct solution in the remaining 2. Whether or not we decide to adopt and refine this proposed learning strategy, then, the above case studies go to show that the superset-subset nature of tiersets in a weakly target-specified MTSL function can be exploited to great effect to facilitate learning.

7.5 Chapter summary

Learnability is an important consideration for many linguistic theories, and is especially important for computational work such as that pursued in this dissertation. In this chapter, I showed that the various function classes defined in previous chapters impart a structure onto a learner's hypothesis space that can allow for efficient learning from positive data alone. I first recapitulated and refined the results of Burness & McMullin (2019), who showed that any OTSL_k function can be learned when the tier is known in advance and who furthermore showed that the tier itself can be learned when $k = 2$. The run time of their tier learner was in $\mathcal{O}(m^5)$ where m is the size of the sample; the run time of the alternative tier learner presented here is in $\mathcal{O}(m^2)$, a significant improvement. Expanding upon these results, I then demonstrated that the multiple tiers of a strongly target-specified OMTSL_2 function can equally be learned using a slight modification of the method used for single tiers. Formal proofs of the above claims were not shown in this chapter, but are available in Appendix B. As for the weakly target-specified OMTSL_2 functions, I do not yet have a formally proven method for learning their multiple sets of multiple tiers. I did, however, propose a strategy, and illustrated some of its desirable properties through a series of small test cases. To close off this chapter, a caveat that applies to all of the above learning results is that they assume the learner is given input-output pairs, when we would ideally assume that learners have access to output forms only. In other frameworks such as Optimality Theory (Prince & Smolensky, 2004), there has been some progress made on the simultaneous learning of a grammar and inputs from just a set of output forms (e.g., Tesar et al., 2003; McCarthy, 2005; Tesar & Prince, 2007; Merchant, 2008; Merchant & Tesar, 2008; Akers, 2012; Tesar, 2012, 2014a,b, 2016). I am optimistic that similar results can and will be developed within the subregular program to which this dissertation belongs.

Chapter 8

Conclusion

Autosegmental tiers capture the intuition that long-distance processes behave as if they were local once we narrow our focus to the relevant segments only, and I have shown how function classes that operate according to tier-based strict locality present an intuitive model of long-distance phonology. The Tier-based Strictly Local (TSL) and Multi-Tiered Strictly Local (MTSL) function classes that I defined are natural extensions of the Strictly Local (SL) functions developped for phonology by Chandlee (2014). SL functions could not capture non-local processes, and the TSL/MTSL functions remedy this limitation without resorting to the full power of the subsequential functions, which were previously offered as a characterization of long-distance phonology (Heinz & Lai, 2013; Luo, 2017; Payne, 2017). Several pathological patterns are amenable to a subsequential analysis but lie outside the capabilities of the TSL and MTSL functions, suggesting that tier-based strict locality is a more accurate characterization of non-local processes than subsequentiality. Probabilistic variants of TSL and MTSL functions can furthermore model optional application and provide a cognitively plausible model of distance-based decay.

The main innovation contained in this thesis is the proposal that attested processes are *target specified*, meaning that tiersets are linked to input elements in a specific way. Namely, each input element designates a superset-subset hierarchy of tiers to which it pays exclusive attention. This added restriction has beneficial consequences for learnability, such that tiers can be discovered efficiently from positive data. Such learning is guaranteed when the tiers are independent (i.e., when each input element specifies just one tier) and the function cares only about the most recent element on each tier. Furthermore, the (M)TSL learning algorithms in this thesis offer a substantial improvement in terms of runtime over previous efforts by Burness & McMullin (2019). With the current algorithms, (M)TSL functions can be learned faster than subsequential functions, at speeds comparable to those estalished for SL functions by Chandlee et al. (2014, 2015).

Taken together, the results collected in this thesis deepen our understanding of the characteristics shared by most long-distance processes and solidify our idea of what is and is not possible in a

long-distance phonological process. By virtue of being formalized in a maximally theory-neutral manner, these outcomes can inform analyses couched within a diverse range of theoretical frameworks, provided only that those analyses assume some sort of transformation between strings (e.g., a mapping between underlying representations and surface pronunciations).

Where do we go from here? As I disclosed in the introduction to this thesis, I largely ignored questions of substance throughout, focusing on the computational properties of long-distance phonology. For their part, theories with a focus on the substantive properties of patterns such as the constraint-based Optimality Theory (Prince & Smolensky, 2004) have made valuable contributions to our understanding of the articulatory and perceptual factors that shape the diachronic and synchronic profile of phonological patterns. We would ideally combine these insights with the computational results, but it is not immediately obvious how to reconcile the two approaches. A promising venue is to investigate tier-based functions from a logical perspective. Bhaskar et al. (2020) showed that the subsequential functions can be logically characterized using Boolean Monadic Recursive Schemes (BMRS), and Chandlee & Jardine (2021) discuss how BMRS transductions can intuitively express substantive generalizations and act much like an ordered hierarchy of constraints. BMRS are an interesting middle ground since we can at once discuss computational and substantive restrictions over them. An example of a computational restriction would be that, at least as conjectured by Bhaskar et al. (2020), a BMRS system describes an Input Strictly Local function if predicates cannot apply recursively to themselves. Substantive restrictions would limit the structures that predicates can identify; for example, we might demand that predicates be formulated using a particular feature theory, making other logically possible predicates inexpressible. No matter how they are to be combined, computational and substantive perspectives are both fundamental to our understanding of long-distance phonological processes.

Bibliography

Akers, Crystal. 2012. *Commitment-based learning of hidden linguistic structures*. New Brunswick. Rutgers University Doctoral dissertation.

Aksënova, Alëna & Deskmukh, Sanket. 2018. Formal restrictions on multiple tiers. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2018*. 64–73.

Andersen, Torben. 1988. Consonant alternation in the verbal morphology of Päri. *Afrika und Übersee* 71. 63–113.

Andersen, Torben. 1999. Consonant alternation and verbal morphology. *Afrika und Übersee* 82. 65–97.

Anderson, Stephen R. 1972. Icelandic u-umlaut and breaking in generative grammar. In Firchow, Evelyn S. & Grimstad, Kaaren & Hasselmo, Nils & O'Neil, Wayne A. (eds.), *Studies for Einar Haugen presented by friends and colleagues* 13–30. The Hague: Mouton. doi: 10.1515/9783110879131-003.

Anderson, Stephen R. 1974. *The organization of phonology*. New York, San Francisco, London: Academic Press. doi:10.1525/aa.1977.79.3.02a00720.

Andersson, Samuel & Dolatian, Hossep & Hao, Yiding. 2020. Computing vowel harmony: The generative capacity of search and copy. In Baek, Hyunah & Takahashi, Chikako & Hong-Lun Yeung, Alex (eds.), *Proceedings of the 2019 Annual Meeting on Phonology*. doi:10.3765/amp.v8i0.4752.

Ao, Benjamin. 1991. Kikongo nasal harmony and context-sensitive underspecification. *Linguistic Inquiry* 22. 193–196.

Applegate, Richard B. 1972. *Ineseño Chumash grammar*. Berkeley. University of California Doctoral dissertation.

Archangeli, Diana & Pulleyblank, Douglas. 1994. *Grounded phonology*. Cambridge, MA: MIT Press.

Archangeli, Diana & Pulleyblank, Douglas. 2007. Harmony. In de Lacy, Paul (ed.), *The Cambridge handbook of phonology* 353–378. Cambridge University Press. doi:10.1017/cbo9780511486371.016.

Árnason, Kristján. 1992. Problems in the lexical phonology of Icelandic. In Dressler, Wolfgang & Luschützky, Hans & Pfeiffer, Oskar & Rennison, John (eds.), *Phonologica 1988* 5–14. Cambridge: Cambridge University Press.

Baković, Eric. 2000. *Harmony, dominance and control*. New Brunswick, NJ. Rutgers University Doctoral dissertation.

Balakaev, N. A. 1962. *Sovremennyi kazaxskij jazyk: Fonetika i morfologija [contemporary Kazakh language: Phonetics and morphology]*. Nauka.

Baum, Leonard E. 1972. An inequality and associated maximization technique occurring in the statistical estimation for probabilistic functions of Markov processes. *Inequalities* 3. 1–8.

Baum, Leonard E. & Petrie, Ted & Soules, George & Weiss, Norman. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics* 41. 164–171. doi:10.1214/aoms/1177697196.

Beddor, Patrice S. & Harnsberger, James D. & Lindemann, Stephanie. 2002. Language-specific patterns of vowel-to-vowel coarticulation. *Journal of Phonetics* 30. 591–627. doi:10.1006/jpho.2002.0177.

Beddor, Patrice S. & Krakow, Rena A. & Lindemann, Stephanie. 2001. Patterns of perceptual compensation and their phonological consequences. In Hume, Elizabeth V. & Johnson, Keith (eds.), *The role of speech perception in phonology* 55–78. San Diego, CA: Academic Press.

Bennett, William G. 2015. *The phonology of consonants: Harmony, dissimilation, and correspondence*. Cambridge: Cambridge University Press. doi:10.1017/cbo9781139683586.

Benus, Stefan & Gafos, Adamantios. 2007. Articulatory characteristics of Hungarian "transparent" vowels. *Journal of Phonetics* 35. 271–300. doi:10.1016/j.wocn.2006.11.002.

Benus, Stefan & Gafos, Adamantios & Goldstein, Louis. 2004. Phonetics and phonology of transparent vowels in Hungarian. In *Proceedings of the 29th Annual Meeting of the Berkeley Linguistics Society*. 485–497. doi:10.3765/bls.v29i1.1000.

Bhaskar, Siddharth & Chandlee, Jane & Jardine, Adam & Oakden, Christopher. 2020. Boolean monadic recursive schemes as a logical characterization of the subsequential functions. In *Language and automata theory and applications* (Lecture Notes in Computer Science, Vol. 12308). 157–169. Springer. doi:10.1007/978-3-030-40608-0_10.

Bickmore, Lee S. & Kula, Nancy C. 2013. Ternary spreading and the OCP in Copperbelt Bemba. *Studies in African Linguistics* 42. 101–132.

Blevins, Juliette. 2004. *Evolutionary phonology: The emergence of sound patterns*. Cambridge University Press. doi:10.1017/cbo9780511486357.

Blevins, Juliette & Garrett, Andrew. 1998. The origins of consonant-vowel metathesis. *Language* 74. 508–556. doi:10.2307/417792.

Blevins, Juliette & Garrett, Andrew. 2004. The evolution of metathesis. In Hayes, Bruce & Kirchner, Robert & Steriade, Donca (eds.), *Phonetically based phonology* 117–156. Cambridge: Cambridge University Press. doi:10.1017/cbo9780511486401.005.

Bobuafor, Mercy. 2013. *A grammar of Tafi*. University of Leiden Doctoral dissertation.

Boyce, Suzanne E. 1988. *The influence of phonological structure on articulatory organization in Turkish and in English: Vowel harmony and coarticulation*. Yale Doctoral dissertation.

Bright, William. 1957. *The Karok language*. Berkeley and Los Angeles: University of California Press.

Burness, Phillip & McMullin, Kevin. 2019. Efficient learning of Output Tier-Based Strictly 2-Local functions. In *Proceedings of the 16th Meeting on the Mathematics of Language*. 78–90. Association for Computational Linguistics. doi:10.18653/v1/w19-5707.

Casali, Roderic F. 2002. Nawuri ATR harmony in typological perspective. *Journal of West African Languages* 29. 3–43.

Castellanos, Antonio & Vidal, Enrique & Varó, Miguel A. & Oncina, José. 1998. Language understanding and subsequential transducer learning. *Computer Speech and Language* 12. 193–228. doi:10.1006/csla.1998.0045.

Chandlee, Jane. 2014. *Strictly Local phonological processes*. University of Delaware Doctoral dissertation.

Chandlee, Jane & Eyraud, Rémi & Heinz, Jeffrey. 2014. Learning Strictly Local subsequential functions. *Transactions of the Association for Computational Linguistics* 2. 491–503. doi: 10.1162/tacl_a_00198.

Chandlee, Jane & Eyraud, Rémi & Heinz, Jeffrey. 2015. Output Strictly Local functions. In *Proceedings of the 14th Meeting on the Mathematics of Language (MOL 2015)*. 112–125. doi: 10.3115/v1/w15-2310.

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

Chandlee, Jane & Heinz, Jeffrey. 2018. Strict Locality and phonological maps. *Linguistic Inquiry* 49. 23–60. doi:10.1162/ling_a_00265.

Chandlee, Jane & Heinz, Jeffrey & Jardine, Adam. 2018. Input strictly local opaque maps. *Phonology* 35. 171–205. doi:10.1017/s0952675718000027.

Chandlee, Jane & Jardine, Adam. 2021. Resursive schemes for phonological analysis. *Language* TBD. TBD.

Chandlee, Jane & Lindell, Steven. under review. A logical characterization of Strictly Local functions. Ms., Haverford College.

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 2. 113–124. doi:10.1109/tit.1956.1056813.

Chomsky, Noam & Halle, Morris. 1968. *The sound pattern of English*. New York: Harper and row.

Clements, George N. 1985. The geometry of phonological features. *Phonology Yearbook* 2. 225–252. doi:10.1017/s0952675700000440.

Clements, George N. & Hume, Elizabeth. 1995. The internal organization of speech sounds. In Goldsmith, John (ed.), *The handbook of phonological theory* 245–306. Cambridge, MA and Oxford, UK: Blackwell.

Clements, George N. & Sezer, Engin. 1982. Vowel and consonant disharmony in Turkish. In van der Hulst, Harry & Smith, Norval (eds.), *The structure of phonological representations (Part II)* 213–255. Dordrecht: Foris.

Coetzee, Andries & Pater, Joe. 2011. The place of variation in phonological theory. In Goldsmith, John & Riggle, Jason & Yu, Jason (eds.), *The handbook of phonological theory* 401–431. Blackwell 2nd edn. doi:10.1002/9781444343069.ch13.

Cole, Jennifer & Kisseberth, Charles. 1994. An optimal domains theory of harmony. In Yoon, James H. (ed.), *Proceedings of the Formal Linguistics Society of Mid-America 5*. 101–114. University of Illinois.

Coupez, André. 1980. *Abrège de grammaire Rwanda*. Butare: Institut National de Recherche Scientifique.

Creel, Sarah C. & Newport, Elissa L. & Aslin, Richard N. 2004. Distant melodies: statistical learning of nonadjacent dependencies in tone sequences. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 30. 1119–1130. doi:10.1037/0278-7393.30.5.1119.

Crowhurst, Megan & Hewitt, Mark. 1995. Prosodic overlay and headless feet in Yidiɲ. *Phonology* 12. 39–84. doi:10.1017/s0952675700002384.

Davis, Stuart. 1995. Emphasis spreading in Arabic and Grounded Phonology. *Linguistic Inquiry* 26. 465–498.

de Blois, Kornelis F. 1975. *Bukusu generative phonology an aspects of Bantu structure* (Annales 85). Tervuren: Musée Royal de l'Afrique Centrale. doi:10.2307/1159247.

de la Beaujardière, Jean-Marie. 2004. Malagasy dictionary and encyclopedia of Madagascar. `http://malagasyword.org/bins/homePage`.

de la Higuera, Colin. 1997. Characteristic sets for polynomial grammatical inference. *Machine Learning Journal* 27. 125–138. doi:10.1023/A:1007353007695.

de la Higuera, Colin. 2010. *Grammatical inference: Learning automata and grammars*. New York: Cambridge University Press. doi:10.1017/cbo9781139194655.

De Santo, Aniello & Graf, Thomas. 2019. Structure sensitive tier projection: Applications and formal properties. In Bernardi, R. & Kobele, G. & Pogodalla, S. (eds.), *Formal grammar 2019* (Lecture Notes in Computer Science, vol. 11668). 35–50. Springer. doi:10.1007/978-3-662-59648-7_3.

de Wit, Gerrit. 2015. *Liko phonology and grammar: A Bantu language of the Democratic Republic of the Congo*. University of Leiden Doctoral dissertation.

Dettweiler, Stephen H. 2000. Vowel harmony and neutral vowels in c'Lela. *Journal of West African Languages* 18. 3–18.

Dimmendaal, Gerrit J. 1983. *The Turkana language*. Dordrecht: Foris. doi:10.1515/9783110869149.

Dinnsen, Daniel & Eckman, Fred R. 1977. Some substantive universals in atomic phonology. *Lingua* 45. 1–14. doi:10.1016/0024-3841(78)90017-7.

Dixon, Robert M. W. 1977. *A grammar of Yidiñ*. Cambridge: Cambridge University Press. doi:10.1017/cbo9781139085045.

Dresher, B. Elan & Nevins, Andrew. 2017. Conditions on iterative rounding harmony in Oroqen. *Transactions of the Philological Society* 115. 365–394. doi:10.1111/1467-968x.12104.

Elgot, Calvin C. & Mezei, Jorge E. 1965. On relations defined by generalized finite automata. *IBM Journal of Research and Development* 9. 47–68. doi:10.1147/rd.91.0047.

Elmedlaoui, Mohamed. 1995. *Aspects de représentations phonolgiques dans certains langues chamito-sémitiques [Aspects of phonological representations in certain Chamito-Semitic languages]*. Rabat, Morocco. Université Mohammed V Doctoral dissertation.

Engelfriet, Joost & Hoogeboom, Hendrik Jan. 2001. MSO definable string transductions and two-way finite state transducers. *ACM Transactions on Computational Logic* 2. 216–254. doi:10.1145/371316.371512.

Fallon, P. D. 2001. Liquid dissimilation in Georgian. In Katholl, A. & Bernstein, M. (eds.), *Proceedings of the 10th eastern states conference on linguistics*. 105–116. Ithaca, NY: DMLL Publications.

Filiot, Emmanuel & Reynier, Pierre-Alain. 2016. Transducers, logic, and algebra for functions of finite words. *ACM SIGLOG News* 3. 4–19. doi:10.1145/2984450.2984453.

Finley, Sara. 2017. Locality and harmony: Perspectives from artificial grammar learning. *Language and Linguistics Compass* 11. doi:10.1111/lnc3.12233.

Gafos, Adamantios & Benus, Stefan. 2006. Dynamics of phonological cognition. *Cognitive Science* 30. 1–39. doi:10.1207/s15516709cog0000_80.

Gafos, Adamantios & Dye, Amanda. 2011. Vowel harmony: Transparent and opaque vowels. In van Oostendorp, Marc (ed.), *The Blackwell companion to phonology*, vol. 4. Wiley-Blackwell. doi:10.1002/9781444335262.wbctp0091.

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

Garcia, Pedro & Vidal, Enrique & Oncina, José. 1990. Learning Locally Testable languages in the strict sense. In *Proceedings of the Workshop on Algorithmic Learning Theory*. 325–338. Japanese Society for Artificial Intelligence.

Gebhart, Andrea L. & Newport, Elissa L. & Aslin, Richard N. 2009. Statistical learning of adjacent and non-adjacent dependencies among non-linguistic sounds. *Psychonomic Bulletin and Review* 16. 486–490. doi:10.3758/pbr.16.3.486.

Gick, Bryan & Pulleyblank, Douglas & Campbell, Fiona & Mutaka, Ngessimo. 2006. Low vowels and transparency in Kinande vowel harmony. *Phonology* 23. 1–20. doi:10.1017/S0952675706000741.

Gold, E. Mark. 1967. Language identification in the limit. *Information and Control* 10. 447–474. doi:10.1016/s0019-9958(67)91165-5.

Goldsmith, John. 1976. *Autosegmental phonology*. MIT dissertation.

Goldwater, Sharon & Johnson, Mark. 2003. Learning OT constraint rankings using a maximum entropy model. In *Proceedings of the Workshop on Variation within Optimality Theory*. 111–120.

Gouskova, Maria & Gallagher, Gillian. 2019. Inducing nonlocal constraints from baseline phonotactics. *Natural Language and Linguistic Theory* 38. 77–116. doi:10.1007/s11049-019-09446-x.

Graf, Thomas & Mayer, Connor. 2018. Sanskrit n-retroflexion is Input-Output Tier-Based Strictly Local. In *Proceedings of SIGMORPHON 2018*. 151–160. doi:10.18653/v1/w18-5817.

Gurevich, Naomi. 2011. Lenition. In van Oostendorp, Marc & Ewen, Colin J. & Hume, Elizabeth V. & Rice, Keren (eds.), *The Blackwell companion to phonology*, vol. 3. Wiley Blackwell. doi:10.1002/9781444335262.wbctp0066.

Hansson, Gunnar Ólafur. 2010. *Consonant harmony: Long-distance interaction in phonology* (University of California Publications in Linguistics 145). Berkeley, CA: University of California Press.

Hao, Yiding & Andersson, Samuel. 2019. Unbounded stress in subregular phonology. In *Proceedings of the 16th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology*

and Morphology. 135–143. Florence, Italy: Association for Computational Linguistics. doi: 10.18653/v1/w19-4216.

Hao, Yiding & Bowers, Dustin. 2019. Action-sensitive phonological dependencies. In *Proceedings of the 16th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology and Morphology*. 218–228. Florence, Italy: Association for Computational Linguistics. doi: 10.18653/v1/w19-4225.

Harms, Phillip. 1985. Epena Pedee (Saija) nasalization. In Brend, R. (ed.), *From phonology to discourse: Studies in six Columbian languages* 13–18. Dallas, TX: Summer Institute of Linguistics.

Harris, John. 2011. Deletion. In van Oostendorp, Marc & Ewen, Colin J. & Hume, Elizabeth V. & Rice, Keren (eds.), *The Blackwell companion to phonology*, vol. 3. Wiley Blackwell. doi: 10.1002/9781444335262.wbctp0068.

Hayes, Bruce & Londe, Zsuzsa. 2006. Stochastic phonological knowledge: The case of Hungarian vowel harmony. *Phonology* 23. 59–104. doi:10.1017/s0952675706000765.

Hayes, Bruce & Wilson, Colin. 2008. A maximum entropy model of phonotactics and phonological learning. *Linguistic Inquiry* 39. 379–440. doi:10.1162/ling.2008.39.3.379.

Hayes, Bruce & Zuraw, Kie & Siptar, Peter & Londe, Zsuzsa. 2009. Natural and unnatural constraints in Hungarian vowel harmony. *Language* 85. 822–863. doi:10.1353/lan.0.0169.

Hayward, Richard J. 1990. Notes on the Aari language. In Hayward, Richard J. (ed.), *Omotic language studies* 425–493. London: School of Oriental and African Studies. doi: 10.4324/9780203045954-14.

Heath, Jeffrey. 2005. *A grammar of Tamashek (Tuareg of Mali)*. Berlin: Mouton de Gruyter. doi:10.1515/9783110909586.

Heinz, Jeffrey. 2009. On the role of locality in learning stress patterns. *Phonology* 26. 303–351. doi:10.1017/s0952675709990145.

Heinz, Jeffrey. 2010a. Learning long-distance phonotactics. *Linguistic Inquiry* 41. 623–661. doi: 10.1162/ling_a_00015.

Heinz, Jeffrey. 2010b. String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. 897–906. Association for Computational Linguistics.

Heinz, Jeffrey & Idsardi, William J. 2010. Learning opaque generalizations: the case of Samala. Unpublished Manuscript.

Heinz, Jeffrey & Lai, Regine. 2013. Vowel harmony and subsequentiality. In Kornai, Andras & Kuhlmann, Marco (eds.), *Proceedings of the 13th Meeting on the Mathematics of Language (MOL 13)*. 52–63. Sofia, Bulgaria: Association for Computational Linguistics.

Heinz, Jeffrey & Rawal, Chetan & Tanner, Herbert G. 2011. Tier-based strictly local constraints for phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*. 58–64. Portland, OR: Association for Computational Linguistics.

Hyman, Larry. 1995. Nasal consonant harmony at a distance: The case of Yaka. *Studies in African Linguistics* 24. 5–30.

Hyman, Larry. 1998. Positional prominence and the 'prosodic trough' in Yaka. *Phonology* 15. 41–75. doi:10.1017/s0952675798003522.

Hyman, Larry. 2011. Tone: Is it different? In Goldsmith, John & Riggle, Jason & Yu, Alan C. (eds.), *The handbook of phonological theory* 197–239. Oxford: Wiley-Blackwell 2nd edn.

Itō, Junko & Mester, Armin. 1999. The phonological lexicon. In Tsujimura, Natsuko (ed.), *The handbook of Japanese linguistics* 62–100. Oxford: Blackwell. doi:10.1002/9781405166225.ch3.

Jardine, Adam. 2016. Computationally, tone is different. *Phonology* 33. 247–283. doi:10.1017/s0952675716000129.

Jardine, Adam & Heinz, Jeffrey. 2016. Learning Tier-Based Strictly 2-Local languages. *Transactions of the Association for Computational Linguistics* 4. 87–98. doi:10.1162/tacl_a_00085.

Jardine, Adam & McMullin, Kevin. 2017. Efficient learning of Tier-Based Strictly k-Local languages. In *International Conference on Language and Automata Theory and Applications (LATA 2017)*. 64–76. doi:10.1007/978-3-319-53733-7_4.

Johnson, C. Douglas. 1972. *Formal aspects of phonological description*. The Hague: Mouton. doi:10.1515/9783110876000.

Jurgec, Peter. 2011. *Feature spreading 2.0: A unified theory of assimilation*. University of Tromso Doctoral dissertation.

Kaplan, Aaron. 2008. *Noniterativity is an emergent property of grammar*. University of California Santa Cruz Doctoral dissertation.

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

Kari, Ethelbert E. 2007. Vowel harmony in Degema, Nigeria. *African Study Monographs* 28. 87–97.

Kaun, Abigail. 1995. *The typology of rounding harmony: An Optimality Theoretic approach*. Los Angeles. University of California Doctoral dissertation.

Kaun, Abigail. 2004. The typology of rounding harmony. In Hayes, Bruce & Kirchner, Robert & Steriade, Donca (eds.), *Phonetically based phonology* 87–116. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511486401.004.

Kimenyi, Alexandre. 1979. *Studies in Kinyarwanda and Bantu phonology*. Carbondale, IL: Linguistic Research.

Kimper, Wendell A. 2011. *Competing triggers: Transparency and opacity in vowel harmony*. University of Massachusetts Amherst Doctoral dissertation.

Kimper, Wendell A. 2017. Not crazy after all these years? Perceptual grounding for long-distance vowel harmony. *Laboratory Phonology* 8. 19. doi:10.5334/labphon.47.

Koo, Hahn & Callahan, Lydia. 2012. Tier-adjacency is not a necessary condition for learning phonotactic dependencies. *Language and Cognitive Processes* 27. 1425–1432. doi:10.1080/01690965.2011.603933.

Koo, Hahn & Oh, Young-il. 2013. Beyond tier-based bigrams: An artificial grammar learning study. *Language Sciences* 38. 53–58. doi:10.1016/j.langsci.2013.01.004.

Korn, David. 1969. Types of labial vowel harmony in the Turkic languages. *Anthropological Linguistics* 11. 98–106.

Kula, Nancy & Bickmore, Lee. 2015. Phrasal phonology in Copperbelt Bemba. *Phonology* 32. 147–176. doi:10.1017/s095267571500007x.

Lambert, Dakotah & Rogers, James. 2020. Tier-Based Strictly Local stringsets: Perspectives from model and automata theory. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2020*. 330–337. New Orleans, Louisianna.

Lamont, Andrew & O'Hara, Charlie & Smith, Caitlin. 2019. Weakly deterministic transformations are subregular. In *Proceedings of the 16th workshop on computational research in phonetics, phonology and morphology*. 196–205. Association for Computational Linguistics. doi: 10.18653/v1/w19-4223.

Lesley-Neuman, Diane. 2012. Morpho-phonological levels and grammaticalization in Karimojong: A review of the evidence. *Studies in African Linguistics* 41. 99–169.

Li, Bing. 1996. *Tungusic vowel harmony*. The Hague: Holland Academic Graphics.

Lindblad, Vern M. 1990. *Neutralization in Uyghur*. University of Washington Master's thesis.

Linebaugh, Gary. 2007. *Phonetic grounding and phonology: Vowel backness harmony and vowel height harmony*. University of Illinois Doctoral dissertation.

Loos, Eugene. 1969. *The phonology of Capanahua and its grammatical basis* (Summer Institute of Linguistics publications in linguistics and related fields 20). University of Oklahoma: Summer Institute of Linguistics.

Luo, Huan. 2017. Long-distance consonant agreement and subsequentiality. *Glossa: A Journal of General Linguistics* 2. 1–25. doi:10.5334/gjgl.42.

Manuel, Sharon. 1999. Cross-language studies: Relating language-particular coarticulation patterns to other language-particular facts. In Hardcastle, W. J. (ed.), *Coarticulation: Theory, data and techniques* 179–198. Cambridge: Cambridge University Press. doi:10.1017/cbo9780511486395.009.

Martin, Andrew. 2005. *The effects of distance on lexical bias: Sibilant harmony in Navajo compounds*. Los Angeles. University of California Master's thesis.

Mayer, Connor & Major, Travis. 2018. A challenge for Tier-Based Strict Locality from Uyghur backness harmony. In *Formal Grammar 2018* (Lecture Notes in Computer Science 10950) 62–83. Berlin: Springer. doi:10.1007/978-3-662-57784-4_4.

McCarthy, John. 1988. Feature geometry and dependence: A review. *Phonetica* 43. 84–108. doi:10.1159/000261820.

McCarthy, John. 2005. Taking a free ride in morphophonemic learning. *Catalan Journal of Linguistics* 4. 19–56. doi:10.5565/rev/catjl.112.

McCarthy, John. 2007. Consonant harmony via correspondence: Evidence from Chumash. In Bateman, Leah & O'Keefe, Michael & Reilly, Ehren & Werle, Adam (eds.), *Papers in Optimality Theory III* 223–237. BookSurge Publishing.

McCollum, Adam G. 2018. Vowel dispersion and Kazakh labial harmony. *Phonology* 35. 287–326. doi:10.1017/s0952675718000052.

McCollum, Adam G. & Baković, Eric & Mai, Anna & Meinhardt, Eric. 2020. Unbounded circumambient patterns in segmental phonology. *Phonology* 37. 215–255. doi:10.1017/s095267572000010x.

McCollum, Adam G. & Essegbey, James. 2018. Unbounded harmony is not always myopic: Evidence from Tutrugbu. In Bennett, William G. & Hracs, Lindsay & Storoshenko, Dennis R. (eds.), *Proceedings of the 35th West Coast Conference on Formal Linguistics*. 251–258. Sommerville, MA: Cascadilla Proceedings Project.

McCollum, Adam G. & Kavitskaya, Darya. 2018. Non-iterative vowel harmony in Crimean Tatar. In Bennett, William G. & Hracs, Lindsay & Storoshenko, Dennis Ryan (eds.), *Proceedings of the 35th west coast conference on formal linguistics*. 259–268.

McMullin, Kevin. 2016. *Tier-based locality in long-distance phonotactics: Learnability and typology*. Vancouver, BC. University of British Columbia Doctoral dissertation.

McMullin, Kevin & Aksënova, Alëna & De Santo, Aniello. 2019. Learning phonotactic restrictions on multiple tiers. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, vol. 2. 377–378.

McMullin, Kevin & Hansson, Gunnar Ólafur. 2016. Long-distance phonotactics as Tier-Based Strictly 2-Local Languages. In Albright, Adam & Fullwood, Michelle A. (eds.), *Proceedings of the 2014 Annual Meeting on Phonology*. Washington, DC: Linguistic Society of America. doi:10.3765/amp.v2i0.3750.

McMullin, Kevin & Hansson, Gunnar Ólafur. 2019. Inductive learning of locality relations in segmental phonology. *Laboratory Phonology* 10. doi:10.5334/labphon.150.

McNaughton, Robert & Papert, Seymour A. 1971. *Counter-free automata*. Cambridge, MA: MIT Press.

Meinhardt, Eric & Mai, Anna & Baković, Eric & McCollum, Adam G. 2020. On the proper treatment of weak determinism: Subsequentiality and simultaneous application in phonological maps. Unpublished manuscript.

Merchant, Nazarré. 2008. *Discovering underlying forms: Contrast pairs and ranking*. New Brunswick. Rutgers University Doctoral dissertation.

Merchant, Nazarré & Tesar, Bruce. 2008. Learning underlying forms by searching restricted lexical subspaces. In *Proceedings of the Forty-First Conference of the Chicago Linguistics Society (2005), vol. II: The Panels*. 33–47.

Michel, Daniel. 2009. Positional transparency in c'Lela. In *Chicago linguistic society (cls) 45*. Chicago: Chicago Linguistic Society.

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

Nevins, Andrew. 2010. *Locality in vowel harmony* (Linguistic Inquiry Monographs 55). MIT Press. doi:10.7551/mitpress/9780262140973.001.0001.

Newport, Elissa L. & Aslin, Richard N. 2004. Learning at a distance i: statistical learning of non-adjacent dependencies. *Cognitive Psychology* 48. 127–162. doi:10.1016/s0010-0285(03)00128-2.

Ní Chiosáin, Máire & Padgett, Jaye. 2001. Markedness, segment realization, and locality in spreading. In *Segmental phonology in Optimality Theory: Constraints and representations* 118–156. Cambridge: Cambridge University Press. doi:10.1017/cbo9780511570582.005.

Noske, Manuela. 1996. [ATR] harmony in Turkana. *Studies in African Linguistics* 25. 61–99.

Noske, Manuela. 2000. ATR harmony in Turkana: A case of Faith Suffix $>$ Faith Root. *Natural Language and Linguistic Theory* 18. doi:10.1023/A:1006474124675.

Novelli, Bruno. 1985. *A grammar of the Karimojong language*. Berlin: Reimer.

Odden, David. 1994. Adjacency parameters in phonology. *Language* 70. 289–330. doi:10.2307/415830.

Ohala, John J. 1994. Towards a universal, phonetically-based theory of vowel harmony. In *Proceedings of the 3rd International Conference on Spoken Language Processing*. 491–494.

O'Hara, Charlie & Smith, Caitlin. 2019. Computational complexity and sour-grapes-like patterns. In *Supplemental proceedings of the annual meeting on phonology 2018*. doi:10.3765/amp.v7i0.4502.

Oncina, José & Garcia, Pedro. 1991. Inductive learning of subsequential functions. Tech. rep. University Politecnia de Valencia. Technical Report DSIC II-34.

Oncina, José & Garcia, Pedro & Vidal, Enrique. 1993. Learning subsequential transducers for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15. 448–458. doi:10.1109/34.211465.

Oncina, José & Varó, Miguel A. 1996. Using domain information during the learning of a subsequential transducer. In Miclet, Laurent & de la Higuera, Colin (eds.), *Grammatical interference: Learning syntax from sentences* (Lecture Notes in Artificial Intelligence 1147) 301–312. Berlin: Springer. doi:10.1007/bfb0033364.

Onn, Farid M. 1980. *Aspects of Malay phonology and morphology: A generative approach*. Kuala Lumpur: Universiti Kebangsaan Malaysia.

Orešnik, Janez. 1975. The modern Icelandic u-umlaut rule. In Dahlstedt, Karl-Hampus (ed.), *The Nordic languages and modern linguistics 2: Proceedings of the second international conference of Nordic and general linguistics*. 621–633. Stockholm: Almqvist and Wiskell.

Orešnik, Janez. 1977. Modern Icelandic u-umlaut from the descriptive point of view. *Gripla* 1. 151–182.

Orr, Carolyn. 1962. Ecuador Quichua phonology. In Elson, Benjamin (ed.), *Studies in ecuadorial indian languages* 60–77. Norman, Oklahoma: Summer Institute in Linguistics.

Ozburn, Avery. 2019. A target-oriented approach to neutrality in vowel harmony: Evidence from Hungarian. *Glossa* 4. 1–36. doi:10.5334/gjgl.681.

Padgett, Jaye. 1995. Partial class behaviour and nasal place assimilation. In *Proceedings of the arizona phonology conference: Workshop on features in Optimality Theory*. 145–183. doi:10.1002/9780470756171.ch19.

Payne, Amanda. 2017. All dissimilation is computationally subsequential. *Language* 93. 353–371. doi:10.1353/lan.2017.0076.

Poser, William. 1982. Phonological representations and action-at-a-distance. In van der Hulst, Harry & Smith, Norval (eds.), *The structure of phonological representations*, vol. 2 121–158. Dordrecht: Foris.

Poser, William. 1993. Are strict cycle effects derivable? In Hargus, Sharon & Kaisse, Ellen (eds.), *Studies in lexical phonology* 315–321. New York: Academic Press. doi:10.1016/b978-0-12-325071-1.50017-0.

Prince, Alan & Smolensky, Paul. 2004. Optimality Theory: Constraint interaction in generative grammar. In McCarthy, John J. (ed.), *Optimality Theory in phonology: A reader*. Malden, MA: Blackwell. doi:10.1002/9780470756171.ch1.

Pulleyblank, Douglas. 2002. Harmony drivers: No disagreement allowed. In Larson, Julie & Paster, Mary (eds.), *Proceedings of the 28th Annual Meeting of the Berkeley Linguistics Society*. 249–297. University of California, Berkeley. doi:10.3765/bls.v28i1.3841.

Rebrus, Péter & Törkenczy, Miklós. 2016. A non-cumulative pattern in vowel harmony: A frequency-based account. In Ólafur Hansson, Gunnar & Farris-Trimble, Ashley & McMullin, Kevin & Pulleyblank, Douglas (eds.), *Proceedings of the 2015 annual meeting on phonology*. doi:10.3765/amp.v3i0.3692.

Ribeiro, Eduardo R. 2003. Directionality in vowel harmony: The case of Karajá (Macro-Jê). In Larson, Julie & Paster, Mary (eds.), *Proceedings of the Berkeley Linguistics Society 28*. 475–485. Berkeley, CA: Berkeley Linguistics Society. doi:10.3765/bls.v28i1.3859.

Rice, Keren. 1993. A reexamination of the feature [sonorant]: The status of 'sonorant obstruents'. *Language* 69. 308–344. doi:10.2307/416536.

Ringen, Catherine O. 1975. *Vowel harmony: Theoretical implications*. Bloomington, IN. Indiana University Doctoral dissertation.

Ringen, Catherine O. 1979. Vowel harmony in Igbo and Diola-Fogny. *Studies in African Linguistics* 3. 247–259.

Rogers, James & Heinz, Jeffrey & Fero, Margaret & Hurst, Jeremy & Lambert, Dakotah & Wibel, Sean. 2013. Cognitive and sub-regular complexity. In *Formal Grammar* (Lecture Notes in Artificial Intelligence 8036) 90–108. Springer. doi:10.1007/978-3-642-39998-5_6.

Rogers, James & Pullum, Geoffrey K. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information* 20. 329–342. doi:10.1007/s10849-011-9140-2.

Rose, Sharon. 2004. Long-distance vowel-consonant agreement in Harari. *Journal of African Languages and Linguistics* 25. 41–87. doi:10.1515/jall.2004.003.

Rose, Sharon & Walker, Rachel. 2004. A typology of consonant agreement as correspondence. *Language* 80. 475–531. doi:10.1353/lan.2004.0144.

Rose, Sharon & Walker, Rachel. 2011. Harmony systems. In Goldsmith, John & Riggle, Jason & Yu, Alan C. (eds.), *The handbook of phonological theory*. Blackwell 2nd edn. doi:10.1002/9781444343069.ch8.

Ryan, Kevin. 2017. Attenuated spreading in Sanskrit retroflex harmony. *Linguistic Inquiry* 48. 299–340. doi:10.1162/ling_a_00244.

Sagey, Elizabeth C. 1986. *The representation of features and relations in non-linear phonology*. MIT Doctoral dissertation.

Sapir, J. David. 1965. *A grammar of Diola Fogny: A language spoken in the Basse-Casamance region of Senegal*. London, UK: Cambridge University Press.

Smith, Caitlin & O'Hara, Charlie. 2019. Formal characterizations of true and false sour grapes. In *Proceedings of the society for computation in linguistics 2019*, vol. 2. 338–341.

Suomi, Kari. 1983. Palatal harmony: A perceptually motivated phenomenon? *Nordic Journal of Linguistics* 6. 1–35. doi:10.1017/s0332586500000949.

Svantesson, Jan-Olof & Tsendina, Anna & Karlsson, Anastasia & Franzén, Vivian. 2005. *The phonology of Mongolian*. New York: Oxford University Press.

Tesar, Bruce. 2012. Learning phonological grammars for output-driven maps. In *Proceedings of NELS 39*.

Tesar, Bruce. 2014a. *Output-driven phonology*. Cambridge University Press. doi:10.1017/cbo9780511740039.

Tesar, Bruce. 2014b. When worst is best: Grammar construction in phonological learning. In *Proceedings of NELS 43*, vol. 2. 167–180.

Tesar, Bruce. 2016. Phonological learning with output-driven maps. *Language Acquisition* 24. 148–167. doi:10.1080/10489223.2016.1223079.

Tesar, Bruce & Alderete, John & Hornwood, Graham & Merchant, Nazarré & Nishitani, Koichi & Prince, Alan. 2003. Surgery in language learning. In Garding, Gina & Tsujimura, Mimu (eds.), *Proceedings of the 22nd West Coast Conference on Formal Linguistics*. 477–490. Sommerville, MA: Cascadilla Press.

Tesar, Bruce & Prince, Alan. 2007. Using phonotactics to learn phonological alternations. In *Proceedings of the Thirty-Ninth Conference of the Chicago Linguistics Society (2003)*. 209–237.

Tucker, Archibald N. & Mpaayei, John T. O. 1955. *A Maasai grammar, with vocabulary*. London: Longmans, Green and Co.

van Oostendorp, Marc & Revithiadou, Anthi. 2005. Quasi-opacity and headed spans in Silly and Megisti Greek. Paper presented at the 13th Manchester Phonology Meeting.

Vaux, Bert. 2000. Disharmony and derived transparency in Uyghur vowel harmony. In *Proceedings of NELS 30*. 671–698.

Vaysman, Olga. 2009. *Segmental alternations and metrical theory*. MIT Doctoral dissertation.

Vidal, Enrique & Tollard, Franck & de la Higuera, Colin & Casacuberta, Francisco & Carrasco, Rafael. 2005a. Probabilistic finite-state machines - Part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(7). 1013–1025. doi:10.1109/tpami.2005.147.

Vidal, Enrique & Tollard, Franck & de la Higuera, Colin & Casacuberta, Francisco & Carrasco, Rafael. 2005b. Probabilistic finite-state machines - Part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(7). 1026–1039. doi:10.1109/tpami.2005.148.

Walker, Rachel. 1998. *Nasalization, neutral segments, and opacity effects*. University of California, Santa Cruz Doctoral dissertation. doi:10.4324/9781315054544. Published by Garland Publishing, New York, 2000.

Walker, Rachel. 2000. Yaka nasal harmony: Spreading or segmental correspondence. In *Annual Meeting of the Berkeley Linguistics Society (BLS 26)*. 323–332. doi:10.3765/bls.v26i1.1164.

Walker, Rachel. 2001. Round licensing and bisyllabic triggers in Altaic. *Natural Language and Linguistic Theory* 19. 827–878.

Walker, Rachel. 2005. Weak triggers in vowel harmony. *Natural Language and Linguistic Theory* 23. 917–989. doi:10.1007/s11049-004-4562-z.

Walker, Rachel. 2011. *Vowel patterns in language*. New York: Cambridge University Press. doi:10.1017/cbo9780511973710.

Walker, Rachel. 2012. Vowel harmony in Optimality Theory. *Language and Linguistics Compass* 6. 575–592. doi:10.1002/lnc3.340.

Walker, Rachel. 2014. Nonlocal trigger-target relations. *Linguistic Inquiry* 45. 501–523. doi:10.1162/ling_a_00165.

Walker, Rachel & Byrd, Dani & Mpiranya, Fidèle. 2008. An articulatory view of Kinyarwanda coronal harmony. *Phonology* 25. 499–535. doi:10.1017/s0952675708001619.

Walker, Rachel & Mpiranya, Fidèle. 2006. On triggers and opacity in coronal harmony. In Cover, Rebecca T. & Kim, Yuni (eds.), *Proceedings of the 31stth Annual Meeting of the Berkeley Linguistics Society*. University of California, Berkeley. doi:10.3765/bls.v31i1.880.

Wilson, Colin. 2003. Experimental investigation of phonological naturalness. In Tsujimura, Mimu & Garding, Gina (eds.), *Proceedings of the 22nd West Coast Conference on Formal Linguistics*. 533–546. Sommerville, MA: Cascadilla Press.

Wilson, Colin. 2006. Unbounded spreading is myopic. In *Phonology Fest Workshop on Current Perspectives on Phonology*. Indinana University.

Zhang, Xi. 1996. *Vowel systems of the Manchu-Tungus languages of China*. University of Toronto Doctoral dissertation.

Zhang, Xi & Dresher, B. Elan. 1996. Labial harmony in written Manchu. *Saksaha: A Review of Manchu Studies* 1. 13–24.

Zhang, Yanchang & Li, Bing & Zhang, Xi. 1989. *The Oroqen language*. Changchun: Jilin University Press.

Zymet, Jesse. 2015. Distance-based decay in long-distance phonological processes. In Steindl, Ulrike & Borer, Thomas & Fang, Huilin & Pard, Alfredo G. & Guekguezian, Peter & Hsu, Brian & O'Hara, Charlie & Ouyang, Iris Chuoying (eds.), *Proceedings of the 32nd West Coast Conference on Formal Linguistics*. 72–81. Sommerville, MA: Cascadilla Press.

Zymet, Jesse. 2020. Malagasy ocp targets a single affix: Implications for morphosyntactic generalization in learning. *Linguistic Inquiry* 51. 624–634. doi:10.1162/ling_a_00356.

Appendix A

Additional unbounded circumambient patterns

In Section 5.3, I discussed how patterns reported to exhibit unbounded circumambience can be broadly divided into two types based on the relationship that holds between the two "halves" that it can be divided into. On the one hand are those processes that can be divided into two conflicting sub-processes, and on the other hand are those processes that can be divided into two cooperating sub-processes. As I discussed using data from Karimojong and Tutrugbu, such decompositions (to varying degrees) meet the looser definitions of Weak Determinism and Tier-based Weak Locality (see Section 5.1). This Appendix analyzes the other processes known to me that exhibit unbounded circumambience, classifying them into one of the broader types just described. Yidiɲ liquid dissimilation, Sanskrit retroflexion harmony/spreading, and Liko ATR harmony, are analyzed in Section A.1 as antagonistic pairs of component functions. Yaka height harmony is analyzed in Section A.2 as a cooperating pair of component functions.

A.1 Conflicting processes

A.1.1 Yidiɲ

In Yidiɲ, the suffix that marks 'going' aspect contains an /l/ that will dissimilate to [r] if it would be followed by another [l], unless that resulting [r] would be preceded by another [r] (Dixon, 1977; Crowhurst & Hewitt, 1995; Payne, 2017). Example (74a) shows a form where the /l/ is not under any pressure to dissimilate, example (74b) shows a form where dissimilation applies, and example (74c) shows a form where dissimilation is blocked. I abstract away from the truncation affecting the suffix, since it governed by root class membership and prosodic structure (Dixon, 1977; Crowhurst & Hewitt, 1995).

(74) Unbounded circumambience in Yidiɲ liquid dissimilation (Dixon, 1977, p. 99-100)

a.	/ɟuŋga-ŋalin-ɲu/ run-going-past	[ɟuŋga-ŋaliː-ɲ]	‘went running’
b.	/ɟuŋga-ŋalin-ŋal-ɲu/ run-going-with-past	[ɟuŋgaː-ri-ŋaː-l]	‘went running with’
c.	/burwa-ŋalin-ŋal-ɲu/ jump-going-with-past	[burwaː-li-ŋaː-l]	‘went jumping with’

Yidiɲ /l/ dissimilation exhibits unbounded circumambience since we need to know whether an instance of /l/ is preceded by an [r] and followed by an [l], and both conditioning segments (if they exist) can be an unbounded distance away. Fortunately, Dixon (1977) and Payne (2017) argue that the blocking of regressive /l/ dissimilation by a preceding [r] can be interpreted as the result of a secondary progressive process of [r] dissimilation. The net result of such an analysis is that the work of regressive [l] dissimilation gets undone by progressive [r] dissimilation in certain circumstances. Each component process is computationally very simple on its own (both are TSL_2), although the fact that [l] dissimilation feeds [r] dissimilation while also being counterfed by [r] dissimilation makes the overall pattern more complex than subsequential. There is nothing particularly interesting about the two functions aside from the complexity that results from their interaction, so in the interest of space I will not analyze them any further here.

A.1.2 Sanskrit

Sanskrit is well-known for its process of retroflexion harmony, usually referred to by its Sanskrit name *nati*, whereby non-lateral retroflex continuants /ɻ/, /ɻ̩/, /ɻ̩ː/, and /ʂ/ generally cause the closest following coronal /n/ to become retroflex [ɳ] (Hansson, 2010; Jardine, 2016; Ryan, 2017; Graf & Mayer, 2018). Data exemplifying the basic process are shown in (75). *Nati* is subject to a number of constraints that make it rather challenging to analyze, most relevantly the fact that it is blocked across a left root boundary (henceforth denoted with $\surd$) when the target is followed by a retroflex consonant (Hansson, 2010; Jardine, 2016; Ryan, 2017; Graf & Mayer, 2018). This particular blocking effect, exemplified in (76), creates a potential instance of unbounded circumambience. The trigger can be an unbounded distance to the left of /n/, but it is unclear from the data whether a following retroflex consonant can actually block from any distance (Ryan, 2017; Graf & Mayer, 2018).

(75) Basic instances of Sanskrit *nati* (Ryan, 2017, p. 305)

a.	káːm-eːna	‘by desire’
b.	ɻaːgʱaʊ-eːɳa	‘by the Rāghava’
c.	puʂpaugʱ-eːɳa	‘by the heap of flowers’

(76) Blocking of *nati* across $\sqrt{}$ by following retroflex consonants (Ryan, 2017, p. 325)

a. pɻa-$\sqrt{}$naɻd- *pɻa-$\sqrt{}$ŋaɻd- ‘roar’

b. pɻa-$\sqrt{}$nakʂ- *pɻa-$\sqrt{}$ŋakʂ- ‘approach’

If blocking by following retroflex consonants turns out to be locally bounded, *nati* can in fact be analyzed as a single progressive MTSL function, as I will now show. I then demonstrate that, under certain assumptions, what would appear to be unbounded blocking by following retroflex consonants can be achieved with the inclusion of a secondary regressive MTSL function before the progressive one. The unbounded circumambience exhibited by *nati* is thus TWL, at least with a particular interpretation of the data.

An assumption that is not crucial for the progressive portion of the analysis but that will prove important for the regressive portion is that *nati* results from successive local spreading of retroflexion starting at the trigger and continuing up until the target. Hansson (2010) and Ryan (2017) also base their analysis around this assumption, but Graf & Mayer (2018) caution against it since there is no clear evidence of spreading (e.g., retroflexion on vowels, if it occurs, is not reflected in Sanskrit orthography) and we run the risk of smuggling too much information into strings if our alphabet is expanded to include unpronounced material. Putting aside the uncertain validity of the spreading interpretation of *nati*, the progressive portion of the analysis enforces the basic process while obeying three additional restrictions on its application. First, as shown in (77), *nati* cannot occur across dental, retroflex, or palatal consonants with the exception of /j/ and the triggering segments /ɻ/, /ɻ̩/, /ɻ̩ː/, and /ʂ/ (Hansson, 2010; Ryan, 2017; Graf & Mayer, 2018). Second, as shown in (78), the target of *nati* must be immediately followed by a non-liquid sonorant (Hansson, 2010; Ryan, 2017; Graf & Mayer, 2018). Finally, as shown in (79), *nati* cannot cross a left root boundary ($\sqrt{}$) if the target is immediately preceded by a labial or velar plosive (Ryan, 2017; Graf & Mayer, 2018). All these restrictions can be enforced simultaneously using an ITSL_3 function that has access to three tiers: T_1 containing all input elements, T_2 containing all triggers/blockers plus $\sqrt{}$, and T_3 containing all triggers/blockers but excluding $\sqrt{}$.

(77) Blocking of *nati* by intervening dental, retroflex, and palatal non-triggers, excluding /j/ (Hansson 2010, p. 227; Ryan 2017, pp. 305-306)

a.	ɻat$^{\text{h}}$-eːna	*ɻat$^{\text{h}}$-eːŋa	‘by chariot’
b.	gaɻuɖ-eːna	*gaɻuɖ-eːŋa	‘by Garuda’
c.	pɻa-ɳinaːja	*pɻa-ɳiŋaːja	‘lead forth’
d.	*manuʂj-eːna	manuʂj-eːŋa	‘by human’
e.	*kʂiːɻ-eːna	kʂiːɻ-eːŋa	‘by milk’

(78) Target of *nati* must be immediately followed by a non-liquid sonorant (Hansson 2010, p. 229; Ryan 2017, p. 318)

a. bɻafiman *bɻafimaɳ ‘brahman’
b. tɻ̩=n=t-te *tɻ̩=ɳ=t-te ‘split (3 PL middle)’
c. caɻ-a-n-ti *caɻ-a-ɳ-ti ‘wander (3 PL)’

(79) Blocking of *nati* across $\sqrt{}$ by an immediately preceding labial or velar plosive (Ryan, 2017, pp. 318-320)
a. *pɻa-$\sqrt{}$ɦi-noː-ti pɻa-$\sqrt{}$ɦi-ɳoː-ti ‘incites (3 SG)’
b. *$\sqrt{}$ɻug-ná $\sqrt{}$ɻug-ɳá ‘break (passive participle)’
c. niɻ-$\sqrt{}$ʋig-na *niɻ-$\sqrt{}$ʋig-ɳá ‘unshaken’

To begin, Table A.1 displays a basic instance of *nati*. When the rightmost symbol of the suffix on T_3 is a trigger, all vowels, non-triggers, and non-blockers will become retroflex. For clarity of presentation, segments that have received retroflexion will be underlined. Retroflexion spread starts on step 2 and continues until step 7 when we read /n/. This instance of /n/ is preceded on T_3 by a trigger and so it may be subject to *nati*, but we need to wait and see if it is immediately followed by a sonorant, so nothing is output on this step. When we read /a/ on the next step, our local window tells us that we read /n/ on the previous step and the window on T_3 tells us this /n/ is eligible for *nati*; if this local /n/ were not the first /n/ after the triggering /ɻ/, the window would instead be /nn/. Since /a/ is a sonorant, we apply *nati* by outputting [ɳa]. Admittedly, we are mildly abusing the information afforded to us by the multiple tiers when $k = 3$. Were we to try using $k = 2$, we would not be able to account for the mandatory sonorant adjacency, since /n/ is an icy target (i.e., a simultaneous undergoer and blocker). All instances of /n/ must be projected to T_3 to account for the blocking of further retroflexion spread, but we need to remember which tier symbol (if any) preceded that /n/ on T_3, which requires at least $k = 3$. A different way around this issue would be to have T_3 and T_2 but not T_1 track the output string instead of the input string. The step where we postpone the output decision for /n/ would then not affect the contents of T_2 and T_3, making $k = 2$ viable. I leave the question of whether and how we can combine input sensitivity and output sensitivity into a single function for future research.

Moving on to the other aspects of the analysis, consider what occurs when we encounter a blocker like /t$^{\text{h}}$/ before we reach an instance of /n/. Going from step 3 to 4 in Table A.2, the trigger /ɻ/ gets closed off by the blocker /t$^{\text{h}}$/ on T_2 and T_3. Consequently, the spreading of retroflexion is halted, and the input /n/ on step 5 is immediately written to the output faithfully.

Turning now to the conditional blocking by immediately preceding velar and labial plosives, the fact that $\sqrt{}$ is a member of T_2 but not T_3 becomes relevant. On step 7 in Table A.3, the suffix on the most inclusive tier T_1 contains /g/ so we know we are immediately preceded by a velar stop. Looking at the suffix on T_3 we furthermore see that we are preceded by a trigger of *nati* that is not blocked by a dental, retroflex, or palatal consonant. Thanks to the superset-subset relationship

Step	Window 1 (all elements)	Window 2 (triggers, blockers, and $\surd$)	Window 3 (triggers and blockers)	Input	Output
1	/ /	/ /	/ /	/ɻ/	[ɻ]
2	/ɻ/	/ɻ/	/ɻ/	/aː/	[a̲ː]
3	/ɻaː/	/ɻ/	/ɻ/	/gʱ/	[g̲ʱ]
4	/aːgʱ/	/ɻ/	/ɻ/	/a/	[a̲]
5	/gʰ a/	/ɻ/	/ɻ/	/ʊ/	[ʊ̲]
6	/aʊ/	/ɻ/	/ɻ/	/eː/	[e̲ː]
7	/ʊeː/	/ɻ/	/ɻ/	/n/	[]
8	/eːn/	/ɻn/	/ ɻn/	/a/	[n̲ɑ]

Table A.1: Basic instance of *nati*

Step	Window 1 (all elements)	Window 2 (triggers, blockers, and $\surd$)	Window 3 (triggers and blockers)	Input	Output
1	/ /	/ /	/ /	/ɻ/	[ɻ]
2	/ɻ/	/ɻ/	/ɻ/	/a/	[a̲]
3	/ɻa/	/ɻ/	/ɻ/	/tʰ/	[tʰ]
4	/atʰ/	/ɻtʰ/	/ɻtʰ/	/eː/	[eː]
5	/tʰeː/	/ɻtʰ/	/ɻtʰ/	/n/	[n]
6	/eːn/	/tʰn/	/tʰn/	/a/	[a]

Table A.2: Blocking of *nati* by an intervening coronal

between the tiers, however, the suffix on T_2 tells us that we have crossed a left root boundary since encountering the trigger, turning the immediately preceding /g/ into blocker. We therefore cannot apply *nati* in this case.

But what of the blocking across $\surd$ by a following retroflex consonant? As I mentioned earlier, it is unclear whether following retroflex consonants block from any distance, and if the blocking turns out to bounded we can account for it by increasing the above progressive function's value of k. The analysis becomes trickier, but not impossible, should the blocking turn out to be unbounded. Ryan (2017) analyzes the blocking as being due to a dispreference for two colliding spans of retroflexion when the first crosses $\surd$. He proposes that retroflex consonants also cause regressive spreading of retroflexion, and that progressive spreading will halt immediately before an /n/ if spreading to that /n/ would create a collision between spans. Adapting his analysis into TWL terms, we can have the regressive spreading apply first as a regressive OSL function, letting it run up to /n/ without

Step	Window 1 (all elements)	Window 2 (triggers, blockers, and $\surd$)	Window 3 (triggers and blockers)	Input	Output
1	/ /	/ /	/ /	/n/	[n]
1	/n/	/n/	/n/	/i/	[i]
2	/ni/	/n/	/n/	/ɻ/	[ɻ]
3	/iɻ/	/nɻ/	/nɻ/	/$\surd$/	[]
4	/ɻ$\surd$/	/ɻ$\surd$/	/nɻ/	/ʊ/	[ʊ̲]
5	/$\surd$ʊ/	/ɻ$\surd$/	/nɻ/	/i/	[i̲]
6	/ʊi/	/ɻ$\surd$/	/nɻ/	/g/	[g̲]
7	/ig/	/ɻ$\surd$/	/nɻ/	/n/	[n]
8	/gn/	/$\surd$n/	/ɻn/	/a/	[a]

Table A.3: Blocking of *nati* across $\surd$ by an immediately preceding velar plosive

affecting it. The progressive function would then check not just whether the segment after an eligible /n/ is a sonorant, but also verify that the sonorant does not have secondary retroflexion. Encountering secondary retroflexion tells the progressive function that it has found the start of a retroflexion span. The function should then not apply *nati* if T_2 has the suffix /$\surd$n/ since doing so would create a par of colliding spans, the first of which crosses a left root boundary. A two part derivation along these lines is shown in Tables A.4 and A.5. Note that neither function is fully alphabet preserving, since they can both produce non-phonemic segments (e.g., retroflex vowels). That being said, neither process is length-increasing, and neither introduces "abstract" intermediate symbols that are not part of the ultimate output alphabet, so the analysis fits in the spirit Tier-based Weak Locality.

A.1.3 Liko

ATR harmony in Liko looks much like the harmony in Karimojong analyzed in Section 5.3.1. Like in Karimojong, underlying /a/ becomes [o] if it is preceded by a dominant [+ATR] vowel and not followed by a dominant [−ATR] vowel as shown in (80b). Liko harmony differs from the full Karimojong system, however, in three minor details. First, the low vowel /a/ remains [a] when between a dominant [+ATR] vowel and a dominant [-ATR] vowel, instead of becoming [ɔ] like it does in Karimojong (de Wit, 2015; McCollum et al., 2020) as shown in (80c). Second, dominant [−ATR] suffixes do not override the [+ATR] value of preceding dominant morphemes (de Wit, 2015; McCollum et al., 2020) as shown in (80c-d). Finally, regressive harmony terminates once it reaches the first prefix before a root (de Wit, 2015; McCollum et al., 2020) as shown in (80b-d).

Step	Window (all elements)	Input	Output
1	[ ]	/ʂ/	[ʂ]
2	[ʂ]	/k/	[k]
3	[k]	/a/	[a]
4	[a]	/n/	[n]
5	[n]	/√/	[√]
6	[√]	/a/	[a]
7	[a]	/ɻ/	[ɻ]
8	[ɻ]	/p/	[p]

Table A.4: Regressive retroflexion spread prior to *nati*

Step	Window 1 (all elements)	Window 2 (triggers, blockers, and √)	Window 3 (triggers and blockers)	Input	Output
1	/ /	/ /	/ /	/p/	[p]
2	/p/	/ /	/ /	/ɻ/	[ɻ]
3	/pɻ/	/ɻ/	/ɻ/	/a/	[a]
4	/ɻa/	/ɻ/	/ɻ/	/√/	[]
5	/a√/	/ɻ√/	/ɻ/	/n/	[]
6	/√n/	/√n/	/ɻn/	/a/	[na]
7	/na/	/√n/	/ɻn/	/k/	[k]
8	/ak/	/√n/	/ɻn/	/ʂ/	[ʂ]

Table A.5: Failure of *nati* in order to avoid a collision between spans

Like I did for the Karimojong data, I mark dominant [+ATR] in bold and dominant [−ATR] in italics.

(80) Conflicting dominance in Liko ATR harmony (McCollum et al., 2020, p. 245)

a. /ta-pók-a/ [ta-pók-a]
1PL-leave-FV

b. /ká-**lut**-án-ág-á/ [kó-**lut**-ón-óg-ó]
CL9B-pull-ASSOC-PL-FV

c. /ná-ká-**bi̍n**-ág-á=*gʊ*/ [ná-kó-**bi̍n**-ág-á=*gʊ*]
1SG.PST-NEG-dance-PL-FV.PST=NEG

d. /ná-ká-**bi̍n**-ɪ=**gʊ**/ [ná-kó-**bi̍n**-i=*gʊ*]
1SG.PST-NEG-dance-FV=NEG

To account for Liko harmony as a TWL function, we could say that root vowels and dominant suffix vowels are specified for [±ATR], but recessive affix vowels are unspecified for [±ATR]. Under this assumption, regressive harmony applies first, whereupon a dominant [−ATR] suffix fills in the missing [±ATR] values of unspecified low vowels between itself and the root. Progressive harmony applies afterwards, but cannot affect vowels that already have a specification for [±ATR]. We essentially have an "early bird gets the worm" scenario where the dominant [−ATR] vowel finds the unspecified low vowels before the dominant [+ATR] vowel does.

Somewhat worryingly, the mapping in (80c) looks a lot like sour grapes on the surface: the dominant [+ATR] vowel fails to spread its value rightward because a dominant [−ATR] vowel exists further down the line. Importantly, however, the regressive spreading of dominant [−ATR] affects only low vowels (de Wit, 2015; McCollum et al., 2020). A high vowel between a dominant [+ATR] vowel and a dominant [−ATR] vowel becomes [+ATR], as shown by the mapping in (80d). Liko harmony might not be a true sour grapes process, then, if it turns out that dominant [+ATR] vowels always spread their value rightward as far as they can. Unfortunately, I could not find any data with a string of non-low vowels between a [+ATR] root and a low vowel that would confirm this hypothesis. Were the hypothetical input /**bin**-ɪ-pɪ-a=*gʊ*/ to exist, the hypothesis would predict [**bin**-i-pi-a=*gʊ*] wherein the progressive [+ATR] harmony affecting /ɪ/ does not care that it will not affect /a/. If the output for this input were to be [**bin**-ɪ-pɪ-a=*gʊ*] instead because the progressive [+ATR] harmony cannot affect /a/, we would have an actual sour grapes pattern on our hands.

A.2 Complementary processes (Yaka)

Yaka contains a unique pattern of "vowel plateauing". Suffixal high vowels become mid if they are both preceded *and* followed by a mid vowel, with seemingly no limit on distance in either direction (Jardine, 2016). Consider the data in (81a-b), comparing forms with the applicative suffix /-ila/ to forms with the perfective suffix /-ile/. These show us that a preceding mid vowel on its own is not

enough to trigger height harmony. That a following mid vowel on its own is not enough to cause plateauing is shown by the forms in (81c-d). In fact, when there is no preceding mid vowel in the root, the mid vowel of the perfective /-ile/ becomes [i] in the output, in turn causing the underlying /l/ to become [d] (Hyman, 1998, 2011; Jardine, 2016). Hyman (1998, 2011) analyzes the Yaka pattern as a repair for the precarious status of final mid vowels, which must anchor themselves to a mid vowel in the root or else loose their mid status and become high.

(81) Mid vowel plateauing in Yaka (Hyman, 2011, p. 218)

	Root		Applicative	Perfective
a.	/keb/	'pay attention'	keb-ila	keb-ele
b.	/sol/	'clear brush'	sol-ila	sol-ele
c.	/kik/	'obstruct'	kik-ila	kik-idi
d.	/kas/	'bind'	kas-ila	kas-idi

We can translate Hyman's analysis into a WD or TWL analysis, since a progressive run through the string can always tell us unambiguously whether an affixal mid vowel gets to keep is mid specification. For the TWL analysis to work, root vowels must be distinguished from affix vowels with a special diacritic (which I will represent using subscript R), since root mid vowels do not become high in the absence of a preceding mid vowel, and root mid vowels do not ever spread their mid specification regressively (Hyman, 1998). Under this assumption, we can use an IMTSL$_2$ function with the following tiers as our first progressive function: T_1 consisting of everything and T_2 consisting of just root mid vowels. This first function's only job is to change affixal mid vowels to their high counterparts if they are not word final or if they are word final but not preceded by a root mid vowel. Following that, a second ITSL$_2$ or OTSL$_2$ function with a tier of root and affix mid vowels can enforce spreading to accomplish the desired plateau. An example derivation with plateauing is shown in Table A.6 and an example derivation without plateauing is shown in Table A.7. What is striking about this analysis is that we do not need any abstract intermediate symbols to tell the second function that suffix /e/ must spread its mid specification. The mere fact that suffix /e/ remained /e/ on the progressive run provides this information.

Step	Window 1 (all elements)	Window 2 (root mid vowels)	Input	Output
1	/ /	/ /	/k/	[k]
2	/k/	/ /	/e$_R$/	[e$_R$]
3	/e$_R$/	/e$_R$/	/b/	b
4	/b/	/e$_R$/	/i/	[i]
5	/i/	/e$_R$/	/l/	[l]
6	/l/	/e$_R$/	/e/	[]
7	/e/	/e$_R$/	/⋉/	[e]

Step	Window (root and affix mid vowels)		Input	Output
1	/ /		/e/	[e]
2	/e/		/l/	[l]
3	/e/		/i/	[e]
4	/e/		/b/	[b]
5	/e/		/e$_R$/	[e]
6	/e$_R$/		/k/	[k]

Table A.6: Yaka derivation with vowel plateauing.

Step	Window 1 (all elements)	Window 2 (root mid vowels)	Input	Output
1	/ /	/ /	/k/	[k]
2	/k/	/ /	/a/	[a]
3	/a/	/ /	/s/	[s]
4	/s/	/ /	/i/	[i]
5	/i/	/ /	/l/	[l]
6	/l/	/ /	/e/	[i]
7	/e/	/ /	/⋉/	[]

Step	Window (root and affix mid vowels)		Input	Output
1	/ /		/i/	[i]
2	/ /		/l/	[d]
3	/ /		/i/	[i]
4	/ /		/s/	[s]
5	/ /		/a/	[a]
6	/ /		/k/	[k]

Table A.7: Yaka derivation with no vowel plateauing.

Appendix B

Formal proofs

In Chapter 7, I discussed formal learning results pertaining to O(M)TSL functions, the proofs for which are all contained in this Appendix. Section B.1 provides the automata characterization of the OTSL$_k$ functions upon which the transducer learning results rely. Section B.2 proves that Algorithm 1 in Section 7.2 can efficiently construct a transducer for any OTSL$_k$ function using only positive data when the tier is known in advance. Section B.3.1 proves that combining Algorithm 2 and Algorithm 3 in Section 7.3 can learn the canonical tier of any OTSL$_2$ function efficiently from positive data. Finally, Section B.3.2 proves that combining Algorithm 2 and Algorithm 4 in Section 7.4.1 can learn the canonical tiers of any strongly target-specified OMTSL$_2$ function efficiently from positive data.

B.1 Automata characterization of OTSL$_k$ functions

I begin with the definition of a delimited subsequential finite state transducer (DSFST) borrowed from Chandlee et al. (2015). Recall that $\rtimes \notin \Sigma$ indicates the start of the input and $\ltimes \notin \Sigma$ indicates the end of the input.

Definition B.1 Delimited subsequential finite-state transducer**.**
A DSFST *is a 6-tuple* $(Q, q_0, q_f, \Sigma, \Delta, \delta)$ *where* Q *is a finite set of states,* $q_0 \in Q$ *is the unique initial state,* $q_f \in Q$ *is the unique final state,* Σ *is the finite input alphabet,* Δ *is the finite output alphabet,* $\delta \subseteq Q \times (\Sigma \cup \{\rtimes, \ltimes\}) \times \Delta^* \times Q$ *is the transition function, and the following hold:*

1. $(q, a, u, q') \in \delta \implies [q \neq q_f \ \wedge \ q' \neq q_0]$
2. $(q, a, u, q_f) \in \delta \implies [a = \ltimes \ \wedge \ q \neq q_0]$
3. $(q_0, a, u, q') \in \delta \implies a = \rtimes$

4. $(q, \rtimes, u, q') \in \delta \implies q = q_0$

5. $[(q, a, u, q') \in \delta \ \wedge \ (q, a, u', q'') \in \delta] \implies [q' = q'' \ \wedge \ u = u']$

I now slightly revise the definition of onwardness (Definition 2.10 in Section 2.1.3) so that it applies to DSFSTs, then borrow a lemma from Chandlee et al. (2015) related to onwardness that will be important later. Since the lemma is borrowed directly, I do not repeat its proof here.

Definition B.2 Onwardness (for DSFSTs)**.**
A DSFST is onward *if and only if:*

$$(q_0, \rtimes w, u, q) \in \delta^* \iff u = \mathtt{lcp}(f(w\Sigma^*))$$

Lemma B.1 From Chandlee et al. (2015)**.**
Let $\mathtt{outputs}(q) = \{u \mid (\exists a \in \Sigma \cup \{\rtimes, \ltimes\})(\exists q' \in Q)[(q, a, u, q') \in \delta]\}$*, that is the output edges of the transitions leaving state* q*. If* $\mathcal{M}$ *is an onward DSFST and recognizes* f *then:*

- $(\forall q \neq q_0)[\mathtt{lcp}(\mathtt{outputs}(q)) = \lambda]$
- $\mathtt{lcp}(\mathtt{outputs}(q_0)) = \mathtt{lcp}(f(\Sigma^*))$

The size of a DSFST $\mathcal{M} = (Q, q_0, q_f, \Sigma, \Delta, \delta)$ is $|\mathcal{M}| = |Q| + |\delta| + \sum_{(q,a,u,q') \in \delta} |u|$, and the relation defined by that DSFST is $\mathcal{R}(\mathcal{M}) = \{(x, y) \in \Sigma^* \times \Delta^* \mid (q_0, \rtimes x \ltimes, y, q_f) \in \delta^*\}$. When a DSFST satisfies the following definition, it will compute an OTSL$_k$ function, as shown by Lemmas B.2 and B.3 which are adapted from Chandlee et al. (2015) for tier-based cases.

Definition B.3 OTSL$_k$ DSFST**.**
An onward DSFST $\mathcal{M} = (Q, q_0, q_f, \Sigma, \Delta, \delta)$ *is OTSL*$_k$ *for the tier* $T \subseteq \Delta$ *if:*

1. $Q = S \cup \{q_0, q_f\}$ *with* $S \subseteq T^{\leq k-1}$
2. $(q_0, \rtimes, u, q') \in \delta \Rightarrow q' = \mathtt{suff}_T^{k-1}(u)$
3. $(\forall q \neq q_0)(\forall a \in \Sigma)[(q, a, u, q') \in \delta \Rightarrow q' = \mathtt{suff}_T^{k-1}(qu)]$

Lemma B.2.
If $\mathcal{M} = (Q, q_0, q_f, \Sigma, \Delta, \delta)$ *is an OTSL*$_k$ *transducer for the tier* T*, then for all* $w \in \Sigma^*$*:*

$$(q_0, \rtimes w, u, q) \in \delta^* \implies [q = \mathtt{suff}_T^{k-1}(u)]$$

Proof. By induction on the size of w. The initial case is valid for $|w| = 0$ since if $(q_0, \rtimes, u, q) \in \delta^*$ then $(q_0, \rtimes, u, q) \in \delta$ and $q = \mathtt{suff}_T^{k-1}(u)$ by point 2 in Definition B.3. Suppose now that for some $n > 0$, the lemma holds for inputs of size n. Let w be of size n such that $(q_0, \rtimes w, u, q) \in \delta^*$

and suppose $(q, a, v, q') \in \delta$ which implies $(q_0, \rtimes wa, uv, q') \in \delta^*$. By induction, we know that $q = \texttt{erase}_T(t_1x_1t_2x_2...t_{k-1}x_{k-1}) = \texttt{suff}_T^{k-1}(u)$ where $t_i \in T$ and $x_i \in (\Delta - T)^*$. Therefore $r = t_1x_1t_2x_2...t_{k-1}x_{k-1}$ is a suffix of u. By point 3 in Definition B.3 we get $q' = \texttt{suff}_T^{k-1}(rv) = \texttt{suff}_T^{k-1}(qv) = \texttt{suff}_T^{k-1}(\texttt{Suff}_T^{k-1}(u)v) = \texttt{suff}_T^{k-1}(uv)$. □

Lemma B.3.
Any OTSL$_k$ transducer computes an OTSL$_k$ function.

Proof. Let $\mathcal{M} = (Q, q_0, q_f, \Sigma, \Delta, \delta)$ be an OTSL$_k$ transducer for the tier T computing some function f and let $w_1, w_2 \in \Sigma^*$ be such that $\texttt{suff}_T^{k-1}(f^p(w_1)) = \texttt{suff}_T^{k-1}(f^p(w_2))$. Since $\mathcal{M}$ is onward, there exists $q, q' \in Q$ such that $(q_0, \rtimes w_1, f^p(w_1), q) \in \delta^*$ and $(q_0, \rtimes w_2, f^p(w_2), q') \in \delta^*$. By Lemma B.2 we get $q = \texttt{suff}_T^{k-1}(f^p(w_1)) = \texttt{suff}_T^{k-1}(f^p(w_2)) = q'$ which implies that $\texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$. Therefore f is an OTSL$_k$ function. □

I assume a fixed but arbitrary total order $\prec$ over the letters of Σ, an order which I extend to all strings in Σ^* by defining the *length-lexicographical order* (Oncina et al., 1993; Chandlee et al., 2015) as follows. String w_1 occurs length-lexicographically before w_2 (written as $w_1 \lhd w_2$) when $|w_1| < |w_2|$ or, if $|w_1| = |w_2|$, when $a_i \prec b_i$ where a_i is the i^{th} letter in w_1, where b_i is the i^{th} letter in w_2, and where i is the first position on which w_1 and w_2 differ. For example, given $\Sigma = \{a, b\}$ and $a \prec b$ we get $\lambda \lhd a \lhd b \lhd aa \lhd ab \lhd ba \lhd bb \lhd aaa$ and so on. Below I will frequently make reference to the length-lexicographically earliest input string that can lead to a state q in a given transducer $\mathcal{M}$, which I will denote as w_q.

Definition B.4 Earliest string**.**
Given a transducer $\mathcal{M} = \langle Q, q_0, q_f, \Sigma, \Delta, \delta \rangle$, the earliest string that leads to $q \in Q$ is:

$$w_q = min_{\lhd}\{w \in \Sigma^* \mid (\exists u)[(q_0, \rtimes w, u, q) \in \delta^*]\}$$

Theorem B.1 states the reverse of Lemma B.3—that every OTSL$_k$ function can be computed by some OTSL$_k$ transducer—and its proof comes in two parts, presented as Lemmas B.4 and B.5. Once again these are adaptations of nearly identical proofs in Chandlee et al. (2015).

Theorem B.1.
Given an OTSL$_k$ function f and one of its tiers T, the DSFST $\mathcal{M} = (Q, q_0, q_f, \Sigma, \Delta, \delta)$ defined as follows computes f:

1. $Q = S \cup \{q_0, q_f\}$ *with* $S \subseteq T^{\leq k-1}$
2. $(q_0, \rtimes, u, \texttt{suff}_T^{k-1}(u)) \in \delta \iff u = f^p(\lambda)$
3. $(q, a, u, \texttt{suff}_T^1(qu)) \in \delta \iff (\exists w)\,[f^p(w) = vr \wedge \texttt{suff}_T^{k-1}(vr) = q \wedge f^p(wa) = vru]$
   where $[r = t_1x_1t_2x_2...t_{k-1}x_{k-1}] \wedge [t_i \in T] \wedge [x_i \in (\Delta - T)^*] \wedge [v = f^p(w) \cdot r^{-1}]$

4. $(q, \ltimes, u, q_f) \in \delta \iff u = f^p(w_q)^{-1} \cdot f(w_q)$

Proof. Immediate from Lemmas B.4 and B.5 □

Lemma B.4.

Let $\mathcal{M}$ be the transducer defined in Theorem B.1. For all $w \in \Sigma^*$:

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

Proof. ($\Rightarrow$) By induction on the length of w. Suppose that $(q_0, \rtimes w, u, q) \in \delta^*$ and that $|w| = 0$. This means $(q_0, \rtimes, u, q) \in \delta$. By point 2 in Theorem B.1 we get $q = \texttt{suff}_T^{k-1}(u)$ and $f^p(\lambda) = u$, which validates the initial case. Now suppose the result holds for all w of size $n > 0$ and pick such a w. Suppose then that $(q_0, \rtimes wa, u, q) \in \delta^*$. By the definition of δ^* there exists u_1, u_2, q' such that $u = u_1u_2$, such that $(q_0, \rtimes w, u_1, q') \in \delta^*$ and such that $(q', a, u_2, q) \in \delta$. By induction we have $f^p(w) = u_1$, and thus by Lemma B.2 we have $q' = \texttt{suff}_T^{k-1}(f^p(w))$. Now divide the string u_1 into two parts v and r such that $v = u_1 \cdot r^{-1}$ where $r = t_1x_1t_2x_2...t_{k-1}x_{k-1}$ with $t_i \in T$ and $x_i \in (\Delta - T)^*$. This gives us $\texttt{suff}_T^{k-1}(vr) = q' = t_1t_2...t_{k-1}$. By point 3 in Theorem B.1 we get $q = \texttt{suff}_T^{k-1}(ru_2) = \texttt{suff}_T^{k-1}(q'u_2)$. This means that $f^p(wa) = vru_2$ with $v = f^p(w) \cdot r^{-1}$. Therefore $f^p(wa) = vru_2 = f^p(w) \cdot r^{-1} \cdot ru_2 = f^p(w)u_2 = u_1u_2 = u$.

($\Leftarrow$) Also by induction on the length of w. If $|w| = 0$ then $f^p(\lambda) = u$. By point 2 of Theorem B.1 we get $(q_0, \rtimes, u, \texttt{suff}_T^{k-1}(u)) \in \delta$ which validates the base case. Now fix $n > 0$ and suppose the result holds for all w of size n. Pick such a w and let $f^p(wa) = u$. As f is subsequential, there exists u_1 such that $f^p(w) = u_1$. By induction, there exists q such that $(q_0, \rtimes w, u_1, q) \in \delta^*$. The construction of $\mathcal{M}$ according to Theorem B.1 ensures that it is an OTSL$_k$ DSFST so by Lemma B.2 we know that $q = \texttt{suff}_T^{k-1}(u_1) = \texttt{suff}_T^{k-1}(f^p(w))$. Therefore there is a string $r = t_1x_1t_2x_2...t_{k-1}x_{k-1}$ that is a suffix of u_1 where $q = t_1t_2...t_{k-1}$ with $t_i \in T$ and $x_i \in (\Delta - T)^*$. By definition of f^p it is the case that $f^p(wa) = u_1 \cdot u_1^{-1} \cdot u$ which equals $u_1 \cdot r^{-1} \cdot r \cdot u^{-1} \cdot u$. Hence $f^p(wa) = vru_2$, with $u_2 = u_1^{-1} \cdot u$ and $v = u_1 \cdot r^{-1} = f^p(w) \cdot r^{-1}$. Thus, by point 3 in Theorem B.1 we get $(q, a, u_2, \texttt{suff}_T^{k-1}(ru_2)) \in \delta$. Since $u_1 \cdot u_2 = u_1 \cdot u_1^{-1} \cdot u = u$ we have $(q_0, \rtimes wa, u, \texttt{suff}_T^{k-1}(ru_2)) \in \delta^*$. □

Lemma B.5.

Let $\mathcal{M}$ be the transducer defined in Theorem B.1. For all $w \in \Sigma^*$:

$$(w, u) \in \mathcal{R}(\mathcal{M}) \iff f(w) = u$$

Proof. By the definition of $\mathcal{R}(\mathcal{M})$ we know that $(q_0, \rtimes w \ltimes, u, q_f) \in \delta^*$. By definition of δ^* there exists $u_1, u_2 \in \Delta^*$ and $q \in (Q - \{q_0, q_f\})$ such that $(q_0, \rtimes w, u_1, q) \in \delta^*$, such that $(q, \ltimes, u_2, q_f) \in \delta$ and such that $u = u_1u_2$. By Lemma B.4 we know that $f^p(w) = u_1$. By point 4 of Theorem B.1

we have $u_2 = f^p(w_q)^{-1} \cdot f(w_q)$. Therefore $(w_q, u'u_2) \in R(T)$. Now by Lemma B.4 we also have $f^p(w_q) = u'$ and so $u'u_2 = f^p(w_q) \cdot u_2 = f^p(w_q) \cdot f^p(w_q)^{-1} \cdot f(w_q) = f(w_q)$. We furthermore have $\texttt{suff}_T^{k-1}(u_1) = \texttt{suff}_T^{k-1}(f^p(w_q)) = \texttt{suff}_T^{k-1}(f^p(w)) = q$. Since the function f is OTSL$_k$ we therefore know that $\texttt{tails}_f(w_q) = \texttt{tails}_f(w)$, which implies that $(\lambda, f^p(w_q)^{-1} \cdot f(w_q)) \in \texttt{tails}_f(w)$. Thus $f(w) = f^p(w) \cdot f^p(w_q)^{-1} \cdot f(w_q) = u_1u_2 = u$. □

I do not do so here, but one could take the definitions/proofs in this section and define/prove similar automata characterizations for the ITSL$_k$, IMTSL$_k$ functions, and OMTSL$_k$ functions by appropriately altering all references to the state labels.

B.2 Learning OTSL$_k$ functions given the tier

I begin by establishing that the run-time of `build_fst` (Algorithm 1 from Section 7.2) does not grow prohibitively quickly with larger samples.

Lemma B.6 Polynomial time**.**
For any input sample S, the function $\texttt{build\_fst}(S)$ returns a DSFST $\mathcal{M}$ in $\mathcal{O}(|S|^2)$ time.

Proof. The maximum number of non-initial and non-final states to be created is $|\Delta^{\leq k-1}|$ which means that the "while" loop will be called at most $|\Delta^{\leq k-1}|$ times, a constant. Now let $l = \sum_{(w,u)\in S} |w|$ be the sum of the lengths of the input strings in the sample and let $o = max\{|u| : (w, u) \in S\}$ be the length of the longest output string in the sample.

The "for" loop within the main "while" loop is executed $|\Sigma|$ times which is a constant. At each execution, the first conditional of the "for" loop searches the sample for an input string that has a certain input prefix wa, which can be done in at most l steps. The loop then collects all pairs whose input strings have wa as a prefix, which can also be done in at most l steps. Next, the algorithm calculates the longest common prefix of the collected outputs, which can be done in lo steps with an appropriate data structure such as a prefix tree (as noted by Chandlee et al. 2015). Following that the algorithm determines $\texttt{suff}_T^{k-1}(f^p(wa))$, which can be done in at most o steps. Determining the contribution for the transition being calculated (i.e. subtracting one output string from the front of another) can be done in at most o steps.

Following this, the conditional of the "while" loop checks whether the w of wa above exists as an input in the sample, which can be done in at most l steps. It then calculates a final transition, which like above can be done in at most o steps. All the other instructions can be done in constant time. The overall time complexity (factoring out the constants) is thus in $\mathcal{O}(l + o + lo)$, which is quadratic in the size of the sample. □

To simplify the remaining proofs in this section, I will limit myself to total functions (i.e., there are no transitions missing from any state for any input element in the corresponding transducer). I now define the core sample of data necessary for `build_fst` to successfully induce a transducer once an appropriate tier T is known and show that it does not get prohibitively large as the target transducer grows. As defined here, the seed will contain more data than is necessary for learning the transducer, but the excess data will be necessary for learning the tier (see the next section).

Definition B.5 Seed**.**
Given a DSFST $\mathcal{M} = (Q, q_0, q_f, \Sigma, \Delta, \delta)$ computing a subsequential function f, a sample S is a seed for that DSFST if for each $q \in (Q - \{q_0, q_f\})$:

1. $(\rtimes w_q \ltimes, f(w_q)) \in S$.
2. *For all triples $a, b, c \in \Sigma$:*
   a. $(\rtimes w_q a \ltimes, f(w_q a)) \in S$.
   b. $(\rtimes w_q ab \ltimes, f(w_q ab)) \in S$
   c. $(\rtimes w_q abcx \ltimes, f(w_q abcx)) \in S$ *for some* $x \in \Sigma^*$

Lemma B.7 Polynomial data**.**
Given a DSFST $\mathcal{M} = (Q, q_0, q_f, \Sigma, \Delta, \delta)$, there exists a seed whose size is in $\mathcal{O}(|\mathcal{M}|^2)$.

Proof. The number of states in the target transducer is finite and is treated a constant. For item 1 in Definition B.5 there are $|Q|$ input-output pairs $(\rtimes w_q \ltimes, f(w_q))$ in a seed.. For each of these pairs, it is the case that $| \rtimes w_q \ltimes | \leq |Q|$ and it is the case that $|f(w_q)| \leq \sum_{(q,a,u,q') \in \delta_\diamond} |u|$. I write this latter quantity with $x_\diamond$ and note that it is linear in the size of the transducer. The overall length of the inputs in the portion of the seed contributed by item 1 is thus in $\mathcal{O}(|Q|^2)$ and the overall length of the outputs in the portion of the seed contributed by item 1 is thus in $\mathcal{O}(|Q| \cdot x_\diamond)$. Both of these are quadratic in the size of $\mathcal{M}$.

For items 2a, 2b, and 2c in Definition B.5, there are respectively $|\Sigma|$, $|\Sigma|^2$ and $|\Sigma|^3$ corresponding input-output pairs per state $q \in Q$. Factoring out the constants gives us $|Q|$ pairs. For the pairs contributed by item 2c, I restrict myself without loss of generality to pairs $(\rtimes w_q abc \ltimes, f(w_q abc))$. For each pair from item 2, the length of the input is less than or equal to $|Q| + 3$. For each pair from item 2, the length of the output is less than or equal to $(\sum_{(q,a,u,q') \in \delta_\diamond} |u|) + (3 \cdot max\{|u| : (q, a, u, q') \in \delta_\diamond\})$. I write this quantity as $y_\diamond$ and note that $y_\diamond$ is linear in the size of the transducer. The overall length of the inputs in the portion of the seed contributed by item 2 is in $\mathcal{O}(|Q|^2)$ and the overall length of the outputs in the portion of the seed contributed by item 2 is in $\mathcal{O}(|Q| \cdot y_\diamond)$. Both of these are quadratic in the size of $\mathcal{M}$. □

In what follows, I let f be the target OTSL$_k$ function, let T be one of its tiers, let $\mathcal{M}^\diamond = (Q_\diamond, q_0, q_f, \Sigma, \Delta, \delta_\diamond)$ be the target DSFST constructed relative to T according to Theorem B.1, and let $\mathcal{M} = (Q, q_0, q_f, \Sigma, \Delta, \delta)$ be the transducer that `buil_fst` constructs given a sample S, the tier T, and k.

Lemma B.8 Transducer convergence**.**
If the learning sample S contains a seed then:

$$(q_0, \rtimes w, u, r) \in \delta^* \iff (q_0, \rtimes w, u, r) \in \delta_\diamond^*$$

Proof. ($\Rightarrow$) By induction on the length of w. If $|w| = 0$ then we get $(q_0, \rtimes, u, r) \in \delta$ such that $u = \texttt{lcp}(\{y \mid (x, y) \in S\})$ and $r = \texttt{suff}_T^{k-1}(u)$ as a consequence of the initial steps of `build_fst`. As S is a seed, $(\rtimes \ltimes, f(\lambda)) \in S$ and there is an element $(\rtimes a \ltimes, f(a)) \in S$ for all possible $a \in \Sigma$ which implies that $u = \texttt{lcp}(f(\lambda\Sigma^*))$. Since $\mathcal{M}^\diamond$ is onward, we have $(q_0, \rtimes, \texttt{lcp}(f(\lambda\Sigma^*)), r') \in \delta_\diamond$ and since the target is an OTSL$_k$ DSFST, $r' = \texttt{suff}_T^{k-1}(\texttt{lcp}(f(\lambda\Sigma^*)))$ according to Lemma B.2, giving us $r' = \texttt{suff}_T^{k-1}(u) = r$. This validates the base case.

Suppose the lemma is true for strings of length less than or equal to n. I refer to this as the First Inductive Hypothesis (IH1). Let wa be of size $n + 1$ such that $(q_0, \rtimes wa, u, r) \in \delta^*$. By the definition of δ^*, there exist u_1, u_2, q such that $(q_0, \rtimes w, u_1, q) \in \delta^*$, $(q, a, u_2, r) \in \delta$, and $u = u_1 u_2$. By IH1, $(q_0, \rtimes w, u_1, q) \in \delta_\diamond^*$. It needs to be shown that $(q_0, \rtimes wa, u, r) \in \delta_\diamond^*$, or more specifically that the transition $(q, a, u_2, r) \in \delta$ built by the algorithm has an exactly matching transition $(q, a, u_2, r) \in \delta_\diamond$.

First I show that IH1 implies that the string s assigned by the algorithm to be $\texttt{earliest}(q)$ is equal to $\rtimes w_q$. Since the algorithm searches breadth-first, s is the first input that reaches q in $\mathcal{M}$ according to the length-lexicographical order. This gives us $(q_0, s, u, q) \in \delta^*$ and also $(q_0, s, u, q) \in \delta_\diamond^*$ by IH1 since $|s| \leq n$. If $\rtimes w_q \lhd s$ then $\exists q' \neq q$ such that $(q_0, \rtimes w_q, u', q') \in \delta^*$ because $\rtimes w_q$ is a prefix of an input string of the sample S and S contains a seed. Since $\rtimes w_q \lhd s$ and $|s| \leq n$, the existence of $(q_0, \rtimes w_q, u', q') \in \delta^*$ implies the existence of $(q_0, \rtimes w_q, u', q') \in \delta_\diamond^*$ by IH1. Having both $(q_0, \rtimes w_q, u', q') \in \delta_\diamond^*$ and $(q_0, s, u, q) \in \delta_\diamond^*$ implies $q' = q$ since $\mathcal{M}_\diamond$ is OTSL$_k$ relative to T and therefore $q = \texttt{suff}_T^{k-1}(f^p(\rtimes w_q))$ by the definition of w_q, contradicting the initial consequence of supposing $\rtimes w_q \lhd s$. If $s \lhd w_q$, the fact that implies that $(q_0, s, u, q) \in \delta_\diamond^*$ by IH1 since $|s| \leq n$, contradicting the fact that w_q is by definition the shortest string leading to $q \in \mathcal{M}^\diamond$. Therefore $s = \rtimes w_q$.

Next I show that the string o assigned to be $\texttt{out}(q)$ by the algorithm is equal to $f^p(w_q)$. By construction of the seed, $(\rtimes w_q \ltimes, f(w_q)) \in S$ and $(\rtimes w_q a \ltimes, f(w_q a)) \in S$ for all transitions (q, a, x, q') leaving q in $\mathcal{M}^\diamond$. As the target is onward, $\texttt{lcp}(\{x \mid (q, c, x, q') \in \delta_\diamond \wedge c \in \Sigma \cup \{\ltimes\}\}) = \lambda$ by Lemma B.1. This implies $o = \texttt{lcp}(\{y \mid \exists a \in \Sigma \wedge \exists x \in \Sigma^* \; s.t. \; (\rtimes s a x \ltimes, y) \in S\}) = \texttt{lcp}(\{y \mid \exists a \in \Sigma \wedge \exists x \in \Sigma^* \; s.t. \; (\rtimes w_q a x \ltimes, y) \in S\}) = \texttt{lcp}(f(w_q\Sigma^*)) = f^p(w_q)$.

Recalling that $(q, a, u_2, r) \in \delta$, I now characterize u_2 to help establish $(q, a, u_2, r) \in \delta_\diamond$. By construction of a seed, there exists an element $(\rtimes w_q a \ltimes, f(w_q a))$ in S and there exist elements $(\rtimes w_q abx \ltimes, f(w_q abx))$ in S for all possible $b \in \Sigma$. By the onwardness of the target, this implies that $v = \texttt{lcp}(\{y \mid \exists b \in \Sigma \wedge x \in \Sigma^* \; s.t. \; (\rtimes sabx \ltimes, y) \in S)\} \cup \{f(sa)\}) = \texttt{lcp}(f(sa\Sigma^*)) = f^p(sa)$. Therefore $u_2 = o^{-1} \cdot v = f^p(s)^{-1} \cdot f^p(sa) = f^p(w_q)^{-1} \cdot f^p(w_q a)$.

Finally I identify r to complete this part of the proof. By Lemma B.4 we have $(q_0, \rtimes w_q a, f^p(w_q a), r') \in \delta_\diamond^*$ since $\mathcal{T}^\diamond$ is OTSL$_k$ and onward. The fact that $(q_0, \rtimes w_q, f^p(w_q), q) \in \delta_\diamond^*$ by IH1, together with the fact that the target is OTSL$_k$ and onward implies that $(q, a, f^p(w_q)^{-1} \cdot f^p(w_q a), r') = (q, a, o^{-1} \cdot v, r') \in \delta_\diamond$. As the target is OTSL$_k$, $r' = \texttt{suff}_T^{k-1}(o^{-1} \cdot v)$ which is r by construction of the transition in the algorithm. Therefore, given that $(q_0, \rtimes w, u_1, q) \in \delta_\diamond^*$ by IH1, we have $(q_0, \rtimes w_q a, u_1 \cdot o^{-1} \cdot v, r) = (q_0, \rtimes wa, u_1 u_2, r) = (q_0, \rtimes wa, u, r) \in \delta_\diamond^*$.

($\Leftarrow$) This is also by induction on the length of w. If $|w| = 0$, we have $\texttt{lcp}(\{\texttt{outputs}(q_0)) = \texttt{lcp}(f(\Sigma^*))$ by Lemma B.1 since $\mathcal{T}^\diamond$ is onward, and thus $(q_0, \rtimes, \texttt{lcp}(f(\Sigma^*)), r) \in \delta_\diamond$ with $r = \texttt{suff}_T^{k-1}(\texttt{lcp}(f(\Sigma^*)))$ as $\mathcal{M}^\diamond$ is OTSLk for the tier T. By construction of the seed, there is at least one element in S using each transition leaving r. As $\texttt{lcp}(\texttt{outputs}(r)) = \lambda$ by Lemma B.1, this implies that $\texttt{lcp}(\{y|(x, y) \in S\}) = \texttt{lcp}(f(\Sigma^*))$. Therefore $(q_0, \rtimes, \texttt{lcp}(f(\Sigma^*)), r) \in \delta$, which validates the base case.

Suppose the lemma is true for all strings up to length n. I refer to this as the Second Inductive Hypothesis (IH2). Pick wa of legnth $n+1$ such that $(q_0, \rtimes wa, u_1, r) \in \delta_\diamond^*$. By definition of $\delta_\diamond^*$ there exist q, u_1, u_2 such that $(q_0, \rtimes w, u_1, q) \in \delta_\diamond^*$ and $(q, a, u_2, r) \in \delta_\diamond$, with $u_1 u_2 = u$. By IH2, we have $(q_0, \rtimes w, u_1, q) \in \delta^*$ and $(q_0, \rtimes w_q, u_1', q) \in \delta^*$ since it is impossible for $w \neq w_q$ to precede w_q in the length-lexicographical order. It needs to be shown that $(q_0, \rtimes wa, u, r) \in \delta^*$, or more specifically that the target transition $(q, a, u_2, r) \in \delta_\diamond$ has an exactly matching transition $(q, a, u_2, r) \in \delta$.

First I show that IH2 also implies that the string s assigned by the algorithm to be $\texttt{Shortest}(q)$ is equal to $\rtimes w_q$. Suppose $s \lhd \rtimes w_q$. The algorithm constructs its FST Suppose $s \lhd \rtimes w_q$. The algorithm constructs its FST such that s is a prefix of an element in S which means there exists q' such that $(q_0, s, f^p(s), q') \in \delta_\diamond^*$. The IH2 then implies that $(q_0, s, f^p(s), q') \in \delta^*$ but the fact that $s = \texttt{earliest}(q)$ implies that $q = q'$, and the definition of w_q then contradicts $s \lhd \rtimes w_q$. Suppose now that $\rtimes w_q \lhd s$. By construction of the seed, $\rtimes w_q$ is a prefix of an element of the sample, which implies it is considered by the algorithm. The existence of $(q_0, \rtimes w_q, u_1', q) \in \delta_\diamond^*$ implies that $(q_0, \rtimes w_q, u_1', q) \in \delta^*$ by IH2, meaning that $\rtimes w_q$ is a smaller prefix than s that reaches the same state, which is impossible as the construction of the algorithm ensures that s is the earliest prefix that makes the state q reachable. Therefore $\rtimes w_q = s$.

I have already shown above that the string o assigned to be $\texttt{out}(q)$ by the algorithm is equal to $f^p(w_q)$ which implies that $o = f^p(s)$. Now let $v = \texttt{lcp}(\{y \mid \exists b \in \Sigma, x \in \Sigma^* \; s.t. \; (sabx, y) \in S\})$. Since $s = \rtimes w_q$, it is the case that $(q_0, \rtimes w_q a, v, r) \in \delta_\diamond^*$ since, as before, the onwardness of the

target implies that λ is the longest common prefix of the outputs on the transitions out of r. This is because each possible outgoing transition from r is represented in S according to Definition B.5. Consequently $v = f^p(w_q a) = f^p(sa)$. Together these results imply that $u_2 = f^p(w_q)^{-1} \cdot f^p(w_q a) = f^p(s)^{-1} \cdot f^p(sa) = o^{-1} \cdot v$. As the target is an OTSL$_k$ transducer for the tier T and is thus deterministic, $r = \texttt{suff}_T^{k-1}(f^p(w_q a))$. Therefore the transition $(q, a, o^{-1} \cdot v, r')$ that is added to δ is the same as the transition $(q, a, u_2, r) \in \delta_\diamond$. This implies $(q_0, \rtimes wa, u, r) \in \delta^*$. □

Lemma B.9 Characteristic sample**.**
Any learning sample containing a seed is a characteristic sample for `build_fst`*.*

Proof. A corollary of Lemma B.8 is that if a seed is contained in a learning sample, then $(q_0, \rtimes w, u, q) \in \delta^*$ if and only if $f^p(w) = u$. We know this because of Lemma B.4. Now, for all states q we have $(\rtimes w_q \ltimes, f(w_q))$ in the seed, which implies that the algorithm will add $(q, \ltimes, f^p(w_q)^{-1} \cdot f(w_q), q_f)$ to the transition function δ. Since each state is tested only once, this holds for any learning sample containing a seed. Therefore, from any superset of a seed, for any w, the transducer built by the algorithm computes $f^p(w) \cdot f^p(w)^{-1} \cdot f(w) = f(w)$. □

Theorem B.2.
`build_fst` *identifies the OTSL$_k$ functions in polynomial time and data*

Proof. Immediate from Lemmas B.6, B.7, B.8, and B.9. □

I noted above that one could take the definitions/proofs from the automata characterization of the OTSL$_k$ functions and define/prove automata characterizations for the ITSL$_k$, IMTSL$_k$, and OMTSL$_k$ functions just by altering the references to state labels appropriately. Similarly, one could prove that the ITSL$_k$, IMTSL$_k$, and OMTSL$_k$ functions are efficiently learnable once their tier(s) are known if the references to state labelling in this section are appropriately altered.

B.3 Learning tiers

B.3.1 OTSL$_2$ functions

Before proving anything about `get_tier` (Algorithm 3 in Section 7.3), I provide formal statements of the crucial properties exhibited by OTSL$_2$ functions as discussed in Burness & McMullin (2019).

Lemma B.10 Free combination of tiers**.**
Given an OTSL$_2$ function $f : \Sigma^* \to \Delta^*$, if $A \subseteq \Delta$ and $B \subseteq \Delta$ are both tiers that can describe f, then $C = A \cup B$ is also a tier that can describe f.

Proof. If $\texttt{suff}^1_C(f^p(w_1)) = \texttt{suff}^1_C(f^p(w_2)) = t$, then $t \in A$ or $t \in B$. If $t \in A$, then $\texttt{suff}^1_A(f^p(w_1)) = \texttt{suff}^1_A(f^p(w_2)) = t$. Now, since f is OTSL$_2$ and A is a tier for f, this implies that $\texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$. If $t \in B$, then $\texttt{suff}^1_B(f^p(w_1)) = \texttt{suff}^1_B(f^p(w_2)) = t$. Now, since f is OTSL$_2$ and B is a tier for f, this implies that $\texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$. These two cases together tell us that $\texttt{suff}^1_C(f^p(w_1)) = \texttt{suff}^1_C(f^p(w_2))$ implies $\texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$, which means that C is a tier for f. □

Definition B.6 Canonical tier**.**
Given an OTSL$_2$ function $f : \Sigma^* \to \Delta^*$, the tier $T \subseteq \Delta$ for f is the canonical *tier for f if and only if there is no other tier $\Omega \subseteq \Delta$ for f such that $|\Omega| > |T|$.*

Remark B.1.
Given an OTSL$_2$ function $f : \Sigma^* \to \Delta^*$, its canonical tier T is a superset of any other tier for f. (This follows immediately from Lemma B.10.)

Lemma B.11 Absolute non-tier status**.**
Let $f : \Sigma^* \to \Delta^*$ be an OTSL$_2$ function where $T \subseteq \Delta$ is the canonical tier. For every Ω such that $T \subset \Omega$, there will exist a symbol $a \in (\Omega - T)$, a symbol $x \in \Sigma \cup \{\ltimes\}$, and a pair of inputs $w_1, w_2 \in \Sigma^*$ such that $\texttt{suff}^1_\Omega(f^p(w_1)) = \texttt{suff}^1_\Omega(f^p(w_2)) = a$ and $\texttt{cont}_f(x, w_1) \neq \texttt{cont}_f(x, w_2)$.

Proof. By contradiction. Suppose that the lemma is false. This means that for all symbols $a \in (\Omega - T)$, for all pairs of words $w_1, w_2 \in \Sigma^*$, and for all symbols $x \in \Sigma \cup \{\ltimes\}$, it is the case that $\texttt{suff}^1_\Omega(f^p(w_1)) = \texttt{suff}^1_\Omega(f^p(w_2)) = a$ implies that $\texttt{cont}_f(x, w_1) = \texttt{cont}_f(x, w_2)$. Now, since T is a tier for f, it is also the case that for all symbols $b \in T$, for all pairs of words $w_1, w_2 \in \Sigma^*$, and for all symbols $x \in \Sigma \cup \{\ltimes\}$, it is the case that $\texttt{suff}^1_\Omega(f^p(w_1)) = \texttt{suff}^1_\Omega(f^p(w_2)) = b$ implies that $\texttt{cont}_f(x, w_1) = \texttt{cont}_f(x, w_2)$. Together these imply that for all symbols $c \in \Omega$, for all pairs of words $w_1, w_2 \in \Sigma^*$, and for all symbols $x \in \Sigma \cup \{\ltimes\}$, it is the case that $\texttt{suff}^1_\Omega(f^p(w_1)) = \texttt{suff}^1_\Omega(f^p(w_2)) = c$ implies that $\texttt{cont}_f(x, w_1) = \texttt{cont}_f(x, w_2)$.

When $\texttt{suff}^1_\Omega(f^p(w_1)) = \texttt{suff}^1_\Omega(f^p(w_2))$ and $\texttt{cont}_f(x, w_1) = \texttt{cont}_f(x, w_2)$, it is necessarily the case that $\texttt{suff}^1_\Omega(f^p(w_1x)) = \texttt{suff}^1_\Omega(f^p(w_2x))$. This in turn tells us that $\texttt{cont}_f(y, w_1x) = \texttt{cont}_f(y, w_2x)$ for all $y \in \Sigma \cup \{\ltimes\}$. This applies recursively, so it is the case that $\texttt{suff}^1_\Omega(f^p(w_1)) = \texttt{suff}^1_\Omega(f^p(w_2))$ implies that $\texttt{tails}_f(w_1) = \texttt{tails}_f(w_2)$, which means that Ω is a tier for f. However, $|\Omega| > |T|$, contradicting the initial premise that T is the canonical tier for f. □

A final thing to prove before discussing the tier learning algorithm pertains to PTTs that have been made onward. The property below will be crucial in proving the learner's convergence.

Definition B.7 Supported state**.**
Given an onward PTT $P = (Q, q_0, q_f, \Sigma, \Delta, \delta)$, the state $q \in Q$ is supported *if and only if:*

$$(\forall x \in \Sigma \cup \{\ltimes\})[\exists (q, x, y, q') \in \delta]$$

Lemma B.12.
Let P be the onward PTT constructed according to sample S drawn from function f. Given a supported state $q \in Q$, it is the case that we will have written exactly $f^p(q)$ upon entering q after starting in q_0.

Proof. Since q is supported, it is the case that $(\forall x \in \Sigma \cup \{\ltimes\})[\exists (q, x, y, q') \in \delta]$. This in turn means that we have $(q, f(q)) \in S$ and for each $a \in \Sigma$ we have $(qab, f(qab)) \in S$ for some $b \in \Sigma^*$. An onward PTT is deterministic and acyclic, so inputs will only pass through q if they have q as a prefix, and all such inputs in S are guaranteed to do so. Let the set `match` be this subset of the inputs in S. Because all and only the inputs in S that are also in `match` will pass through q, the process of making the PTT onward will push $\texttt{lcp}(f(\texttt{match}))$ past q towards the root such that exactly this `lcp` will have been written when q is entered after starting in q_0. Now recall that $f^p(w) = \texttt{lcp}(\{u \mid u = f(wy) \wedge y \in \Sigma^*\})$. It is sufficient to use a set containing $f(w)$ and at least one $f(wv) = f(wab)$ for each $a \in \Sigma$ (where $b \in \Sigma^*$) because every $v \in \Sigma^*$ is either λ or begins with some $a \in \Sigma$. The set `match` fulfills this criterion and so $\texttt{lcp}(f(\texttt{match})) = f^p(q)$. □

Remark B.2.
Let P be an onward PTT constructed according to a sample drawn from function f. A corollary of Lemma B.12 is that for any supported state q:

- $(q, \ltimes, u, q_f)$ *is such that* $u = \texttt{cont}_f(\ltimes, q) = f^p(q)^{-1} \cdot f(q)$
- (q, σ, v, r) *for* $\sigma \in \Sigma$ *is such that* $v = \texttt{cont}_f(\sigma, q) = f^p(q)^{-1} \cdot f^p(q\sigma)$ *if r is also supported.*

Now I can discuss the tier learning process. I first show that obtaining a tier from a sample does not take prohibitively more time as the sample grows.

Lemma B.13 Polynomial time**.**
For any input sample S, running $\texttt{get\_tier}(S)$ produces a tier T in $\mathcal{O}(|S|^2)$ time.

Proof. Let $l = \sum_{(w,u)\in S} |w|$ be the summed lengths of all inputs in the sample, let $o = max\{|u| : (w, u) \in S\}$ be the longest output length in the sample, let $i = max\{|w| : (w, u) \in S\}$ be the longest input length in the sample, and let s be the number of pairs in the sample. These are all linear in the size of the sample.

First we extract information about the prefix function Constructing $PTT(S)$ requires a single read through S, taking l steps. Making this PTT onward takes at most ol steps (Chandlee et al., 2014). There are at most l non-initial/non-final states in P and `estimate_fp` starts by checking each of these once. For each, it verifies whether all possible $|\Sigma| + 1$ outgoing transitions exist. There are at most $l + s$ transitions in P, meaning that checking whether a transition exists takes at most $l + s$ steps. The first portion of `estimate_fp` thus takes at most $l((l + s)(|\Sigma| + 1))$ steps,

although $|\Sigma| + 1$ is a constant and can be ignored. The second portion sends the sample through P one input letter at a time, checking on each step whether the landing state is in A. Checking whether a state is in A takes at most l steps, so the second portion of `estimate_fp` takes at most l^2 steps. Taken together, the run time of these preparatory steps is in $\mathcal{O}(l + ol + l(l + s) + l^2)$, which is quadratic in the size of the sample.

Now we begin whittling down the tier hypothesis. The *while* loop can run up to $|\Delta|$ times, which is a constant. This loop first initializes the contribution sets that will be constructed, of which there are at most $|\Delta| \cdot |\Sigma|$, another constant. Then, for each of the $l + s$ transitions in P, it searches A up to two times to check whether the origin is supported and whether the destination is supported or final. A single search of A take at most l steps, and if both conditions are met we calculate the relevant output tier suffix, taking at most o steps. Finally, after inspecting all the transitions in P, we check the cardinality of each contribution set, which takes at most $|\Delta| \cdot |\Sigma|$ steps. The overall run time of the tier-learning half of $\texttt{estimate\_fp}(S)$ is thus in $\mathcal{O}((l+s)(l+o))$, which is quadratic in the size of S.

□

Now let $\mathcal{M}$ be the minimal DSFST that computes the target function f. The remaining proofs assume that the sample S contains the seed that we would get from Definition B.5 given $\mathcal{M}$.

Lemma B.14 Evidence availability**.**
If a learning sample S contains a seed and P is the onward PTT for S then for all $w \in \Sigma^*$ and all pairs $x, y \in \Sigma$ there is at least one transition in P corresponding to each of the following that (1) leaves a supported state and (2) ends in q_f or a supported state:

- $\texttt{cont}_f(x, w)$
- $\texttt{cont}_f(\ltimes, w)$
- $\texttt{cont}_f(y, wx)$
- $\texttt{cont}_f(\ltimes, wx)$

Proof. For any input string $w \in \Sigma^*$, reading $\rtimes w$ will lead to some non-initial and non-final state q in $\mathcal{M}$. The target function is subsequential which means that that either $w = w_q$ or else can be replaced thereby since subsequentiality implies that $\texttt{cont}_f(i, w) = \texttt{cont}_f(i, w_q)$ for any $i \in \Sigma \cup \{\ltimes\}$. By the definition of the seed, for every state q in $\mathcal{M}$ and for every triple $x, y, z \in \Sigma$, the learner will see w_q, $w_q x$, $w_q xy$, and $w_q xyzv$ where $v \in \Sigma^*$. This means that the states w_q, $w_q x$, and $w_q xy$ in P are all supported. As Remark B.2 notes, Lemma B.12 then implies that:

- $(w_q, x, a_1, w_q x)$ in P is such that $a_1 = f^p(w_q)^{-1} \cdot f^p(w_q x) = \texttt{cont}_f(x, w_q)$

- $(w_q, \ltimes, a_2, q_f)$ in P is such that $a_2 = f^p(w_q)^{-1} \cdot f(w_q) = \texttt{cont}_f(\ltimes, w_q)$
- $(w_q x, y, a_3, w_q xy)$ in P is such that $a_3 = f^p(w_q x)^{-1} \cdot f^p(w_q xy) = \texttt{cont}_f(y, w_q x)$
- $(w_q x, \ltimes, a_4, q_f)$ in P is such that $a_2 = f^p(w_q x)^{-1} \cdot f(w_q x) = \texttt{cont}_f(\ltimes, w_q x)$

□

Lemma B.15 Tier convergence**.**
Given a learning sample S that contains a seed, running `get_tier`(S) *will produce the canonical tier of f.*

Proof. Let T be the canonical tier of f, and let H be the tier constructed by the algorithm. The algorithm begins with $H = \Delta$, and so either $H = T$ already, or else $H \supset T$. The algorithm is designed to consider all and only the transitions (q, x, y, r) in $P = \texttt{onward}(PTT(S))$ such that q is a supported state and r is a supported state or q_f. As Remark B.2 notes, Lemma B.12 implies that $y = \texttt{cont}_f(x, q)$ for all such transitions. For each considered transition (q, x, y, r) in P, the algorithm sorts y into the bin associated simultaneously with x and with $z = \texttt{suff}^1_H(p)$, where p is equal to the output produced upon reading q in P. Note that the algorithm has easy access to the string p because it is paired with q in the auxiliary set A. Note also that since q is a supported state, Lemma B.12 tells us that $p = f^p(q)$. Now, we know from Lemma B.11 that if $H \supset T$, there will exist a pair of input strings w_1 and w_2 in the domain of f such that $\texttt{cont}_f(x, w_1) \neq \texttt{cont}_f(x, w_2)$ even though $\texttt{suff}^1_H(f^p(w_1)) = \texttt{suff}^1_H(f^p(w_2)) = a$ for some $a \in (H - T)$ and some $x \in \Sigma \cup \{\ltimes\}$. Furthermore, Lemma B.14 tells us that for all non-initial/non-final states s in the minimal DSFST $\mathcal{M}$ producing f, each transition along every possible sequence of two or fewer steps out of s will have at least one equivalent transition in P that is considered by the algorithm. If the first transition along one of these paths produces any elements not in T, we stand the chance of incorrectly binning the second transition when $H \supset T$. Since we see all possible paths of two transitions, at least one pair of unequal contributions (which *should* be placed into two different bins linked to two different members of T since f is OTSL$_2$) will be placed together into a bin that shouldn't exist (because that bin is linked to a non-member of T) when $H \supset T$. Accordingly, at least one of the bins associated with some $b \in (H - T)$ will have a cardinality greater than 1 (assuming repeated strings are counted only once) when $H \supset T$. The algorithm will thus flag and remove at least one $b \in (H - T)$ when $H \supset T$. Conversely, there will be no pair of input strings w_3 and w_4 in the domain of f such that $\texttt{cont}_f(x, w_3) \neq \texttt{cont}_f(x, w_4)$ when $\texttt{suff}^1_H(f^p(w_3)) = \texttt{suff}^1_H(f^p(w_4)) = c$ for any $c \in T$ and any $x \in \Sigma \cup \{\ltimes\}$. Consequently, none of the bins associated with any $c \in T$ will ever surpass a cardinality of 1 (assuming repeated strings are counted only once). When $H = T$, then, the algorithm will add all $d \in H$ to `keep`, at which point `keep` $= H = T$. □

Theorem B.3.
`get_tier` *identifies the canonical tier of any total OTSL$_2$ function in quadratic time and data.*

Proof. Immediate from Lemmas B.7, B.13, and B.15. □

Putting together the results of this section and of Section B.2 tells us that the OTSL$_2$ functions can be learned efficiently from example input-output pairs without any prior knowledge of the function's tier. Furthermore, the properties of OTSL$_2$ functions that form the basis of Burness & McMullin's (2019) tier-learning strategy (Lemmas B.10 and B.11) also hold for ITSL$_2$ functions after switching focus from the output string to the input string, implying that they too can be learned efficiently from example input-output pairs alone.

B.3.2 Strongly target-specified OMTSL$_2$ functions

As I did in the previous section, I provide formal statements of the crucial properties exhibited by strongly target-specified OMTSL$_2$ functions before showing how `get_x_tier` (Algorithm 4 from Section 7.4.1) is guaranteed to succeed by exploiting them. These proofs are mostly the same as their corresponding proofs in the previous section, but I shown them in full for completeness' sake.

Lemma B.16 Free combination of x-tiers**.**
Given a strongly target-specified OMTSL$_2$ function $f : \Sigma^* \to \Delta^*$ and given $x \in \Sigma \cup \{\ltimes\}$, if A and B are both x-tiers for f, then $C = A \cup B$ is also an x-tier for f.

Proof. If $\texttt{suff}^1_C(f^p(w_1)) = \texttt{suff}^1_C(f^p(w_2)) = t$, then $t \in A$ or $t \in B$. If $t \in A$, then $\texttt{suff}^1_A(f^p(w_1)) = \texttt{suff}^1_A(f^p(w_2)) = t$. Now, since f is a strongly target-specified OMTSL$_2$ function and A is an x-tier for f, this implies that $\texttt{cont}_f(x, w_1) = \texttt{cont}_f(x, w_2)$. If $t \in B$, then $\texttt{suff}^1_B(f^p(w_1)) = \texttt{suff}^1_B(f^p(w_2)) = t$. Now, since f is a strongly target-specified OMTSL$_2$ function and B is an x-tier for f, this implies that $\texttt{cont}_f(x, w_1) = \texttt{cont}_f(x, w_2)$. These two cases together tell us that $\texttt{suff}^1_C(f^p(w_1)) = \texttt{suff}^1_C(f^p(w_2))$ implies $\texttt{cont}_f(x, w_1) = \texttt{cont}_f(x, w_2)$, which means that C is an x-tier for f. □

Definition B.8 Canonical x-tier**.**
Given a strongly target-specified OMTSL$_2$ function $f : \Sigma^* \to \Delta^*$ and some $x \in \Sigma \cup \{\ltimes\}$, the x-tier $T \subseteq \Delta$ for f is the canonical *x-tier for f if and only if there is no other x-tier $\Omega \subseteq \Delta$ for f such that $|\Omega| > |T|$.*

Remark B.3.
Given a strongly target-specified OMTSL$_2$ function $f : \Sigma^* \to \Delta^*$ and some $x \in \Sigma \cup \{\ltimes\}$, its canonical x-tier T is a superset of any other x-tier for f. (This follows immediately from Lemma B.16.)

Lemma B.17 Absolute non-x-tier status**.**
Let f be a strongly target-specified OMTSL$_2$ function where $T \subseteq \Delta$ is the canonical x-tier given $x \in \Sigma \cup \{\ltimes\}$. For every Ω such that $T \subset \Omega$, there will exist a symbol $a \in (\Omega - T)$ and two inputs $w_1, w_2 \in \Sigma^*$ such that $\mathtt{suff}^1_\Omega(f^p(w_1)) = \mathtt{suff}^1_\Omega(f^p(w_2)) = a$ and $\mathtt{cont}_f(x, w_1) \neq \mathtt{cont}_f(x, w_2)$.

Proof. By contradiction. Suppose that the lemma is false. This means that for all symbols $a \in (\Omega - T)$ and for all pairs of words $w_1, w_2 \in \Sigma^*$ it is the case that $\mathtt{suff}^1_\Omega(f^p(w_1)) = \mathtt{suff}^1_\Omega(f^p(w_2)) = a$ implies that $\mathtt{cont}_f(x, w_1) = \mathtt{cont}_f(x, w_2)$. Now, since T is an x-tier for f, it is also the case that for all symbols $b \in T$ and for all pairs of words $w_1, w_2 \in \Sigma^*$, we get $\mathtt{suff}^1_\Omega(f^p(w_1)) = \mathtt{suff}^1_\Omega(f^p(w_2)) = b$ implies that $\mathtt{cont}_f(x, w_1) = \mathtt{cont}_f(x, w_2)$. Together these imply that for all symbols $c \in \Omega$ and for all pairs of words $w_1, w_2 \in \Sigma^*$, it is the case that $\mathtt{suff}^1_\Omega(f^p(w_1)) = \mathtt{suff}^1_\Omega(f^p(w_2)) = c$ implies that $\mathtt{cont}_f(x, w_1) = \mathtt{cont}_f(x, w_2)$, which means that Ω is an x-tier for f. However, $|\Omega| > |T|$, contradicting the initial premise that T is the canonical x-tier for f. □

While the modifications that derive `get_x_tier` from `get_tier` affect the relationship between the run time and the size of the alphabets, they do not affect the relationship between the run time and the size of the sample. Let $\mathcal{M}$ be the minimal DSFST that computes the target function f. The remaining proof assumes that the sample S contains the seed that we would get from Definition B.5 given $\mathcal{M}$.

Lemma B.18 Tier convergence**.**
Given a learning sample S that contains a seed, running $\mathtt{get\_x\_tier}(S, x)$ will produce the canonical x-tier of f.

Proof. Let T be the canonical x-tier of f, and let H be the x-tier constructed by the algorithm. The algorithm begins with $H = \Delta$, and so either $H = T$ already, or else $H \supset T$. The algorithm is designed to consider all and only the transitions (q, x, y, r) in $P = \mathtt{onward}(PTT(S))$ such that q is a supported state and r is a supported state or q_f. As Remark B.2 notes, Lemma B.12 implies that $y = \mathtt{cont}_f(x, q)$ for all such transitions. For each considered transition (q, x, y, r) in P, the algorithm sorts y into the bin associated simultaneously with x and with $z = \mathtt{suff}^1_H(p)$, where p is equal to the output produced upon reading q in P. Note that the algorithm has easy access to the string p because it is paired with q in the auxiliary set A. Note also that since q is a supported state, Lemma B.12 tells us that $p = f^p(q)$. Now, we know from Lemma B.17 that if $H \supset T$, there will exist a pair of input strings w_1 and w_2 in the domain of f such that $\mathtt{cont}_f(x, w_1) \neq \mathtt{cont}_f(x, w_2)$ even though $\mathtt{suff}^1_H(f^p(w_1)) = \mathtt{suff}^1_H(f^p(w_2)) = a$ for some $a \in (H - T)$. Furthermore, Lemma B.14 tells us that for all non-initial/non-final states s in the minimal DSFST $\mathcal{M}$ producing f, each transition along every possible sequence of two or fewer steps out of s will have at least

one equivalent transition in P that is considered by the algorithm. If the first transition along one of these paths produces any elements not in T, we stand the chance of incorrectly binning the second transition when $H \supset T$. Since we see all possible paths of two transitions, at least one pair of unequal contributions (which *should* be placed into two different bins linked to two different members of T since f is strongly target-specified OMTSL$_2$) will be placed together into a bin that shouldn't exist (because that bin is linked to a non-member of T) when $H \supset T$. Accordingly, at least one of the bins associated with some $b \in (H - T)$ will have a cardinality greater than 1 (assuming repeated strings are counted only once) when $H \supset T$. The algorithm will thus flag and remove at least one $b \in (H - T)$ when $H \supset T$. Conversely, there will be no pair of input strings w_3 and w_4 in the domain of f such that $\texttt{cont}_f(x, w_3) \neq \texttt{cont}_f(x, w_4)$ when $\texttt{suff}^1_H(f^p(w_3)) = \texttt{suff}^1_H(f^p(w_4)) = c$ for any $c \in T$. Consequently, none of the bins associated with any $c \in T$ will ever surpass a cardinality of 1 (assuming repeated strings are counted only once). When $H = T$, then, the algorithm will add all $d \in H$ to `keep`, at which point $\texttt{keep} = H = T$. □

Theorem B.4.
`get_x_tier` identifies the canonical tiers of any total strongly target-specified OMTSL$_2$ function in quadratic time and data.

Proof. Immediate from Lemmas B.7, B.13, and B.18. □

Putting together the results of this section and of Section B.2 tells us that the OMTSL$_2$ functions can be learned efficiently from example input-output pairs without any prior knowledge of the function's tiers. Furthermore, the properties that form the basis of the learning strategy for strongly target-specified OMTSL$_2$ functions (Lemmas B.16 and B.17) also hold for strongly target-specified IMTSL$_2$ functions after switching focus from the output string to the input string, implying that they too can be learned efficiently from example input-output pairs alone.