Doing Computational Phonology

September 23, 2024

Contents

Part I

Foundations

Chapter 1

Intensional and Extensional Descriptions of Phonological Generalizations

JEFFREY HEINZ

1.1 Generative Phonology

Within languages, the pronunciation of a morpheme often differs depending on its morpho-phonological context. While examples like English *go/went* indicate that these different pronunciations may have almost nothing in common, it is much more typical that the pronunciations of the same morpheme in different contexts are in fact similar, as with common English plural *cat[s]/dog[z]*. The main empirical conclusion linguists have drawn with respect to this phenomena is that the variation in the pronunciation of morphemes is *systematic*. It is no accident that the plural form of *tip* uses [s] just like *cat[s]* and that the plural form of *dud* is [z] just like *dog[z]*. Explaining this systematic variation is an important goal of linguistic theory.

The central hypothesis of Generative Phonology (GP) is presented below.

(★) The observed systematic variation in the pronunciation of morphemes is best explained if people hold a single mental representation of the pronunciation of each morpheme (the underlying representation, UR) which is *lawfully transformed* into its pronounced variants (the surface representation, SR).

This book assumes this hypothesis is correct, and does not review any arguments for it.[1] Readers interested in arguments for this position are directed to Odden (2014, chapter 4) and Kenstowicz and Kisseberth (1979, chapter 6).

If this hypothesis is correct, then there are three questions every theory of generative phonology must address.

(★★)
1. What is the nature of the underlying representations?
2. What is the nature of the surface representations?
3. What is the nature of the transformations between these representations?

These questions are certainly not exhaustive but they are of critical importance. Another related important question is "How different can the underlying representations be from the surface representations?" This has been called the question of abstraction (Kenstowicz and Kisseberth, 1977).

This book provides a general framework which addresses these questions from a *computational* perspective. The computational perspective addresses both the nature of the representations and the nature of the transformations. It is flexible in the sense that different representational schemes can be studied and compared. This is accomplished through *model-theoretic* representations of words and phrases. It is also flexible in the sense that different types of computational power can be studied and compared. This is accomplished by studying what can be accomplished with different kinds of logical expressions. As will be explained, model theory and logic provide a mathematical foundation for theory construction, theory comparison, and descriptive linguistics.

The study of phonology from the computational perspective allows one to construct theories of phonology which provide answers to the above questions. Representational choices and choices of logical power essentially determine the theory and its empirical predictions. Theories of phonology developed under this framework are examples of Computational Generative Phonology (CGP).

[1] The words *transformed* and *transformation* are used here in their original meaning simply to signify that the URs become SRs, and that the SR derived from some UR may not be identical to this UR. If a UR is related to a SR via the transformative component of a phonological grammar, it is also often said the UR is mapped to the SR. These words are deliberately neutral with respect to the specific type of grammar being employed.

To begin motivating CGP, I would like to give some examples of how current phonological theories aim to answer these questions. It is not possible to comprehensively survey here the range of answers that have been offered. Therefore, I only highlight some answers and do so only in very broad strokes.

Rule-based theories, as exemplified by Chomsky and Halle (1968), for example, have argued that the abstract underlying representations are subject to language-specific morpheme structure constraints (MSCs). The transformation from underlying forms to surface forms are due to language-specific rules, which are applied in a language-specific order. Constraints on surface representations were, generally speaking, not part of the ontology of these theories, and therefore were not posited to have any psychological reality. Such generalizations—the phonotactic generalizations—were derivable from the interaction of the MSCs and the rules (Postal, 1968).

On the other hand, in classic Optimality Theory (Prince and Smolensky, 1993, 2004), there are no constraints on underlying representations (richness of the base), but there are psychologically real, universal constraints on surface forms (markedness constraints). The transformation from underlying forms to surface forms is formulated as a process of *global optimization* over these markedness constraints as well as constraints which penalize differences between surface and underlying forms (faithfulness constraints). While both the markedness and faithfulness constraints are universal, their relative importance is language-specific. So in every language the surface pronunciation of an underlying representation is predicted to be the globally optimal form (the one that violates the most important constraints the least). Of course what is optimal varies across languages because the relative importance of the constraints varies across languages.

These two theories are radically different in what they take to be psychologically real. The ontologies of the theories are very different. Perhaps this is most clear with respect to the concept of phonemes (Dresher, 2011). Phonemes exist as a consequence of the ontology of rule-based theories, but they do not as a consequence of the ontology of OT. This is simply because phonemes are a kind of MSC; underlying representations of morphemes must be constructed out of them, and nothing else. In OT, there are no MSCs and hence there are no phonemes. The principle of **Lexicon Optimization** guarantees that the URs of *pit* and *spit* are /$\mathrm{p^hɪt}$/ and /$\mathrm{spɪt}$/, respectively (Kager, 1999). The underlying, mental representation of the voiceless labial stops in both words are not the same. Consequently, the

complementary distribution of speech sounds (allophonic variation) are explained in a very different manner in the two theories, and these theories promote different views of the notion of *contrast*. Despite these differences however, there is an important point of agreement: In both theories, complementary distribution of speech sounds in surface forms is the outcome of a transformation of underlying forms to surface forms.

This is the point I wish to emphasize: neither theory abandons the fundamental insight stated on page 3 in (★). The theories offer radically different *answers* to the questions asked on 4 in (★★), but *they agree on the questions being asked*.[2]

In the remainder of this chapter, I motivate a computational approach to phonology. I first make an important distinction between extensional and intensional descriptions of linguistic generalizations and argue that the former is important for understanding the latter. I then argue that neither rule-based nor constraint-based formalisms as practiced provide adequate intensional descriptions of phonological generalizations.

This is then contrasted with automata and logical descriptions of language. The chapter concludes that logical descriptions of linguistic generalizations have some advantages over automata-theoretic descriptions. This is not to say automata are not useful (they are!) but that logic offers more immediate rewards to linguists interested in writing and analyzing grammars. So when we consider the ways in which we spend our time, logic is a good place to start.

1.2 Extensional and Intensional Descriptions

McCarthy (2008a, pp. 33–34) emphasizes the importance of descriptive generalizations in preparing analyses. "Good descriptive generalizations," he writes "are accurate characterizations of the systematic patterns that can be observed in the data." They are, as he explains, "the essential intermediate step between data and analysis." This is because descriptive generalizations go beyond the data; they make predictions about things not yet observed.

[2]It is true that periodically some work is published which challenges these core ideas, for example the work on output-to-output correspondence (Benua, 1997, and others) or the recent work of Archangeli and Pulleyblank (2022).

Descriptive generalizations are important for computational phonology too. They are typically stated in prose. For example, consider the phonological generalizations below.

Word final vowels are prohibited. (1.1)

Consonant clusters are prohibited word-finally. (1.2)

These generalizations are good ones because they allow the analyst to recognize that potentially unobserved forms like *tapaka* is ill-formed but *tanak* is well-formed with respect to 1.1. Similarly, we recognize that 1.2 distinguishes between forms like *tapakt* and *tanakta*.

The generalizations above divide every possible word of every length cleanly into two sets: those that obey the description and those that do not. This is illustrated in the figure below for the generalization in (1.1). The

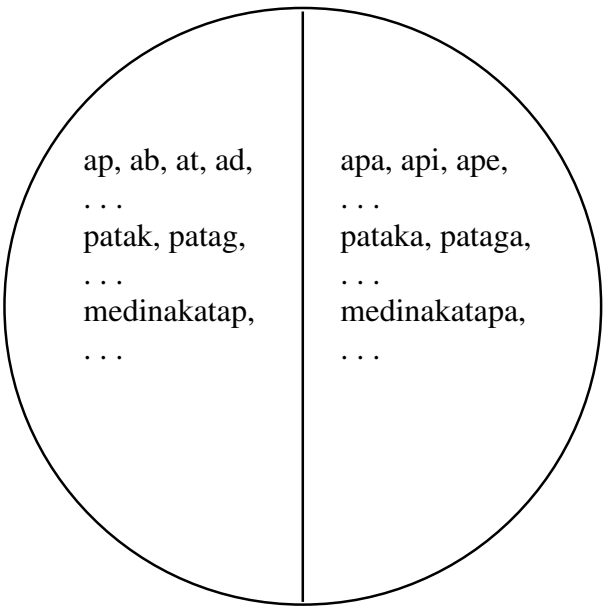


Figure 1.1: The generalization that "Word final vowels are prohibited" partitions the set of all possible forms into two sets.

set of words that are well-formed according to (1.1) is called its *extension*.

Importantly, this set—the extension—is infinite in size. For instance, it is not possible to write down every possible word that obeys the generalization in (1.1). If a set of words formed from a finite alphabet is infinite

then there is no upper bound on the length of words. Likewise, if there is no upper bound on the length of words formed from a finite alphabet then this set is infinite in size. Thus whether the size of a set of words is infinite or not is intertwined with whether or not there is an upper bound on the length of words. These issues are so important to get clear that they are discussed in further detail below.

Extensional descriptions contrast with *intensional descriptions* of generalizations. For now, intensional descriptions can be thought of as grammars that denote the extension. The prose in (1.1) and (1.2) are examples of intensional descriptions. Rule-based grammars and OT grammars are also examples of intensional descriptions. A good intensional description is one where the the extension can be rigorously and precisely defined from the intensional description. Generally, English prose does not make for good intensional descriptions. Further below, I will argue that in their current forms and practice, rule-based grammars and OT grammars are more like English prose than good intensional descriptions.

Let us now return to the infinitely-sized extensions. Is it reasonable for descriptive generalizations like (1.1) to denote an infinite set of words? Yes, it is. One reason is that these generalizations make no reference to length at all. If the length of words mattered, it ought to be part of the generalization. Another way of thinking about this is that if there were a principled upper bound on the length of words, then that would be a generalization *distinct* from (1.1) above, and hence ought not be included within it. Finally, even if for some reason (1.1) ultimately denoted a finite set, there are reasons to treat its extension as infinite anyway. Savitch (1993) argues that large finite sets of strings are often best understood if they are factored into two parts: an infinite set of strings and a separate finite-length condition. They are, in his words, "essentially infinite." The basis of the argument is a demonstration that intensional descriptions of infinite sets can be smaller in size than the intensional descriptions of finite sets.

These infinite-sized extensions do not exist in the same way that your fingernails, your bed, or your brain exists. Instead they exist mathematically. Each generalization is an infinite object like a circle, which is a set of infinitely many points each exactly the same distance from a center. But we can never see the mathematical object in its entirety in the real world. It is a fact that circles as infinite objects do not exist. The situation with linguistic generalizations is similar. The extension is there mathematically,

but we cannot write down every element of the extension in a list for the same reason all points of a circle cannot be written down in a list since there are infinitely many. But we can write down a grammar which can be understood as generating the infinite set, in the same way that a perfect circle can be generated by specifying a center point and a distance (the radius).

The same circle can be described in other ways as well. If we employ the Cartesian plane, we could generate a circle with an equation of the form $(x-a)^2 + (y-b)^2 = r^2$ where the r is the radius of the circle and (a, b) is its center. The equation is interpreted as follows: all and only points (x, y) which satisfy the equation belong to the circle. The equation is an intensional description and the set of (x, y) points satisfying this equation—the circle itself—is its extension.

We can also describe a circle on a plane with polar coordinates instead of Cartesian ones. Recall that polar coordinates are of the form (r, θ) where r is the radius and θ is an angle. The equation $r = 2a\cos(\theta) + 2b\sin(\theta)$ provides the general form of the circle with the radius given by $\sqrt{a^2 + b^2}$ and the center by (a, b) (in Cartesian coordinates). The polar equation is interpreted like the Cartesian one: all and only points (r, θ) which satisfy the equation belong to the circle.

There are some interesting differences between these two coordinate systems. Each point in the Cartesian system has a unique representation, but each point in the polar system has infinitely many representations (since the same angle can be described in infinitely many ways, e.g. $0^\circ = 360^\circ = 720^\circ = \ldots$). If the center of the circle is the origin of the graph, the polar equation simplifies to $r = a$ whereas the Cartesian equation remains more complicated $x^2 + y^2 = r^2$. Thus, the polar equation $r = 4$ and the Cartesian equation $x^2 + y^2 = 16$ are different equations with different interpretations, but they describe the same unique circle: one of radius four centered around the origin. The two equations differ intensionally, but their extension is the same.

It seems strange to ask which of these two descriptions is the 'right' description of this circle. They are different descriptions of the same thing. Some descriptions might be more useful than others for some purposes. It also interesting to ask what properties the circles have irrespective of a particular description. For instance the length of a circle's perimeter and the size of a circle's area are certainly relatable to these descriptions, but they are also in a sense independent of the particulars. The perimeter and

area depend on the radius but not the center, though both the radius and the center appear in the equations above. Perhaps this suggests that the radius is a more fundamental structure to a circle than its center, though both certainly matter.

The analogy I wish to draw is that rule-based and OT-theoretic formalisms are like the Cartesian and polar coordinate systems. The analogy is far from perfect, but it is instructive. Both rule-based and OT analyses provide descriptions of platonic, infinitely sized objects. In many cases, but not all, the two formalisms describe the same object, insofar as the empirical evidence allows.

What is this object? The transformations from underlying representations to surface representations can be thought of as a *function*, in the mathematical sense of the word. Another word for function prevalent in the phonological literature is *map* (Tesar, 2014). For example, consider the two descriptive generalizations below.

Word final vowels delete. (1.3)

Word final vowels delete except when preceded by a consonant cluster. (1.4)

These generalizations also have infinite-sized extensions, but the extensions are better understood as functions. Figure 1.2 illustrates the extension of the generalization expressed in (1.3).

There are three parts to a function. First, there is its domain, which is the set of objects the function applies to. Second, there is its co-domain, which is the set of objects to which the elements of the domain are mapped. Third, there is the map itself, which says which domain elements are transformed (mapped) to which co-domain elements. Thus to specify a function, one needs to provide a description of its domain, its co-domain, and a description of which domain elements become which co-domain elements. Following traditional phonological terminology, I use the term **constraint** to refer to intensional descriptions of either the domain or co-domain.

The parts of a function align nearly perfectly with the fundamental questions of phonological theory given in (★★) on page 4. The underlying representations correspond to the domain. The surface representations make up the co-domain. And the transformation from underlying to surface

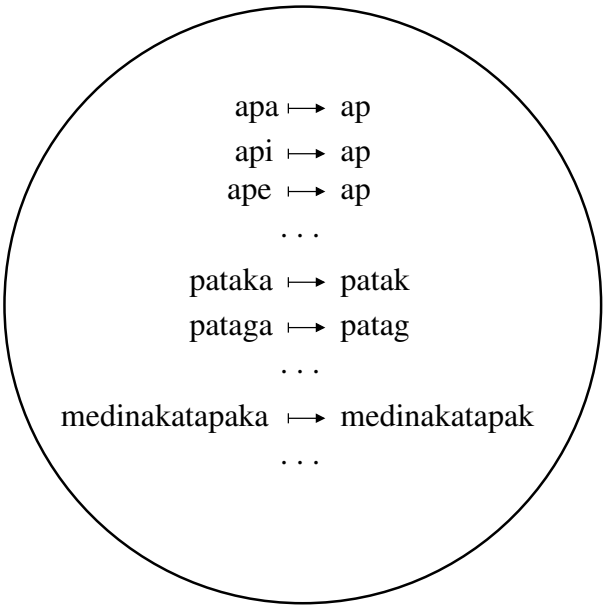


Figure 1.2: The function corresponding to the generalizations that "Word final vowels delete."

forms is the map from domain elements to co-domain elements. From this perspective, describing the phonology of a language requires identifying aspects of this function.

Further, in linguistic typology we are actually interested in the *class* of such functions that correspond to *possible* human phonologies. If the phonologies of languages are circles we would be interested in the universal properties of circles and the extent of their variation. Circles are pretty simple, so the answers are straightforward. All circles have a center and a radius, but their centers can be different points and their radii can have different lengths. What universal properties do phonological functions share? What kind of variation does the human animal permit across these functions?

The point is that when we develop a linguistic generalization, it is important to know what its extension is. Ultimately, the intensional description—the grammar—must generate this extension. The emphasis placed here on the extensional description as an infinite object should not be taken to mean intensional descriptions do not matter. Of course they matter: theories of these intensional descriptions ought to make predictions about

what is psychologically real, predictions that in principle are testable with the right kinds of psycholinguistic and neurolinguistic experimentation. They also can make predictions about linguistic typology since the available intensional descriptions limit the extensions accordingly. In addition to making correct predictions, phonologists expect that intensional descriptions express the 'right' generalizations. Clarity about the extensional descriptions are an essential, intermediate step between the descriptive generalizations stated in prose and formal intensional descriptions (the grammatical analysis).

It is critically important that it is well-understood how the intensional descriptions relate to the extensional ones. We want to be able to answer questions like the following:

1. Given a word w and an intensional description of a constraint C, does w violate C? (We may also be interested in the number of violations of C and the where within the word the violations occur.)

2. Given a word w in the domain of a transformation f what words in the co-domain of f does f map w to, if any?

3. Given a word v in the co-domain of a transformation f what words in the domain of f map to v, if any?

Question 1 is often called the membership problem. Question 2 is often called the generation problem. Question 3 is often called the recognition or parsing problem. Good intensional descriptions allow answers to these questions to be computed correctly and effectively. In the next section, I argue that rule-based intensional descriptions and OT grammars are not good intensional descriptions in this narrow sense.

1.3 Issues with Familiar Grammars

Chomsky and Halle (1968) present a formalization based on rewrite rules. The basic rewrite rule is of the form $\text{A} \longrightarrow \text{B} \;/\; \text{C} \_\_ \text{D}$. This notation is intended to mean that if an input string contains CAD then the output string will output CBD (so A is rewritten as B in the context C __ D). To understand the extension of a rule, we need to know how to apply it. Originally, Chomsky and Halle (1968, p. 344) intended for the rules to apply

simultaneously to all the relevant targets in an input string. They wrote, "To apply a rule, the entire string is first scanned for segments that satisfy the environmental constraints of the rule. After all such segments have been identified in the string, the changes required by the rule are applied simultaneously." For many phonological rules, this explanation appears sufficient to denote the extension. For instance the rule corresponding to the descriptive generalizations (1.3) is $\text{V} \longrightarrow \varnothing \; / \; \_\_ \; \#$. Humans have no difficulty using this rule to answer the generation and parsing problems above given this intensional description. However, it is much less clear what the extension of *any* rule would be. Determining this depends in part on what A, B, C and D themselves are able to denote, and how rules apply when application of the rule can create more CAD sequences.

The phonological literature after SPE addressed the question of rule application (Anderson, 1974), and other types of rule application were identified such as left-to-right or right-to-left. It was clear that the mode of application determined the extension of the rule. For example, for the input string /oana/ and rule $\text{V} \longrightarrow [+\text{nasal}] \; / \; \_\_ \; [+\text{nasal}]$ simultaneous application yields output [oãna] but right-to-left application yields output [õãna]. While linguistically-chosen examples served to distinguish one mode of application from another, general solutions to the generation and recognition questions by Johnson (1972) and Kaplan and Kay (1994) were for the most part ignored by generative phonologists.

It is my contention that rule application is still not well-understood by most students of phonology, despite the careful computational analyses by Johnson (1972); Kaplan and Kay (1994) and Mohri and Sproat (1996). In informal surveys of phonologists in-training, many have difficulty of applying the rule $\text{aa} \longrightarrow \text{b}$ simultaneously to the input /aaa/. People wonder whether the right output is [ab], [ba], or [bb]. According to Kaplan and Kay's analysis, there are two outputs for this input when the rule $\text{aa} \longrightarrow \text{b}$ is applied simultaneously. They are [ab] and [ba]. Their analysis translates rewrite rules into finite-state automata, which are grammars whose extensions are very well defined and understood. These will be explained in a bit more detail in the next section.

Interestingly, Kaplan and Kay's analyses of rule application, which has been implemented in software programs like `xfst` (Beesley and Karttunen, 2003), `openfst` (Allauzen *et al.*, 2007), `foma` (Hulden, 2009a,b), and `pynini` (Gorman, 2016; Gorman and Sproat, 2021) do not exhaust the possible natural interpretations of the rewrite rule $\text{A} \longrightarrow \text{B} \; / \; \text{C} \; \_\_ \; \text{D}$. Like Johnson

and Kaplan and Kay's analyses, Chandlee's (2014) analysis also uses finite-state automata to determine an extension of a rule $A \longrightarrow B \ / \ C \_\_ D$, provided that CAD is a finite set of strings. Unlike Kaplan and Kay, her interpretation of the extension of the rule $aa \longrightarrow b$ maps input /aaa/ to [bb]. This result is arguably what Chomsky and Halle in mind when they described simultaneous application because each *aa* sequence satisfies "the environmental constraints of the rule."

The point of the foregoing discussion is simply this: a rule $A \longrightarrow B \ / \ C \_\_ D$ underdetermines its extension. The extensions are a critical part of any rule-based theory and there is more than one way such rules determine extensions. This point is neither new nor controversial. It is a well-known chapter in the history of phonological theory. Chandlee's (2014) discussion shows that this chapter is not closed. To my knowledge, Bale and Reiss (2018) is the first textbook on phonology that provides an adequate interpretation of the application of rewrite rules.

Optimality Theory is an improvement in some sense. Given an OT grammar and an input form, there is a well-defined solution to the generation problem. This solution follows from the architecture of the OT grammar. The GEN component generates the set of possible candidates and the EVAL component uses the grammar of ranked constraints to select the optimal candidates.

Nonetheless in actual phonological analyses the generation problem faces two difficulties, each acknowledged in the literature. The first one is ensuring that all the possible candidates are actually considered by EVAL. The absence of an overlooked candidate can sink an analysis. The proposed optimal candidate turns out to be less harmonic than some other candidate that the analysts failed to consider. How can analysts ensure that every candidate has been considered?

The second is ensuring that all the relevant constraints are present in the analysis. The absence of a relevant constraint can also sink an analysis. (Prince, 2002, p. 276) makes this abundantly clear. He explains that if a constraint that must be dominated by some other constraint is ignored then the analysis is "dangerously incomplete." Similarly, if a constraint that may dominate some other constraint is omitted then the analysis is "too strong and may be literally false."

As a result, any phonological analysis of a language which does not incorporate the entire set of constraints is not guaranteed to be correct. This makes studying some aspect of the phonology of the language difficult.

The constraints deemed irrelevant to the fragment of the phonology under investigation (and which are therefore excluded) actually need to be shown to be irrelevant for analysts to establish the validity of their OT analyses.

Both these problems in OT can be overcome. The solution again comes from the theory of computation, in particular from the theories of finite-state automata and so-called regular languages (defined and discussed in the next section). The earliest result is that even if the constraints and GEN can be defined in these terms, the maps OT produces are not guaranteed to be definable in these terms — unless the constraints have a finite bound on the maximum number of violations they can assign (Frank and Satta, 1998). Karttunen (1998) uses this fact to provide a solution and software for the generation and recognition problems (see also (Gerdemann and Hulden, 2012)), and so he assumes each constraint has some maximum number of violations. While some theoretical phonologists have argued for this position (McCarthy, 2003), most do not adopt it. Riggle (2004) provides a different solution which does not require bounding the number of violations constraints assign. His solution is guaranteed to be correct provided the map the OT grammar is in fact representable as a finite-state relation (not all of them are). Another solution is present in Albro's (2005) dissertation, which provides a comprehensive OT analysis of the phonology of Malagasy.

Each of these authors make use of finite-state automata to guarantee the correctness of their solutions. However, none of these approaches have yet to make its way into the more commonly used software for conducting OT analyses such as OTSoft (Hayes *et al.*, 2013), OT-Help (Staubs *et al.*, 2010), and OTWorkplace (Prince *et al.*, 2016). A particular weakness of this software, unlike Karttunen's, Riggle's, and Albro's is that they can only work with finite candidate sets, despite the fact that GEN is typically understood as generating an infinite candidate set. Consequently, the commonly used software amounts to nothing more than pen-and-paper approaches with lots of paper and lots of pens, and so the aforementioned issues remain (Karttunen, 2006).

McCarthy (2008a, p. 76) argues the aforementioned computational approaches are only possible in a "narrowly circumscribed phenomenon." However, this ignores Albro's detailed, thorough analysis of the whole phonology of Malagasy (Albro, 2005). McCarthy also argues the methods are only as good as the algorithm that generates the candidates. Of course

that is true, but the alternatives are manual, heuristic methods.[3] People may differ on which is better, but I will place my bets on the algorithm which is guaranteed not to leave out candidates that GEN produces. McCarthy's dismissal of the value of computational approaches is unfortunate, but it is representative of attitudes in the field.

Regardless of the extent to which different researchers appreciate the computational treatments of phonological theories, it is noteworthy and no accident that every attempt to guarantee a solution of the recognition and generation problems (and the membership problem when constraints are involved) makes use of finite-state automata and the theory of regular languages. Even OTWorkplace employs the finite-state calculus by way of regular expressions to automatically assign constraint violations to candidates. What are these devices? And what makes them so good for denoting extensions of phonological generalizations?

1.4 Computational Theory of Language

Automata are a cornerstone of the computational theory of language. Automata are machines that process specific types of data structures like strings or trees. They form a fundamental chapter of computer science. There are many kinds of automata. The Turing machine is just one example. Pushdown automata are another. Readers are referred to texts such as Kozen (1997), (Hopcroft *et al.*, 2006) and Sipser (2012) for overviews of the theory of computation.

There are also deep connections between automata and logic. In this section, I will briefly review finite-state automata for string processing. Then I will informally introduce logic as another way of providing an intensional description of phonological generalizations. Their extensions are also well-defined; and in fact in many cases there are algorithms which convert a logical description into an automaton that describes exactly the same extension (Büchi, 1960; Thomas, 1997; Engelfriet and Hoogeboom, 2001).

We begin with a simple automaton, the **finite-state acceptor**. It is an intensional description with a well-defined extension. As a matter of fact,

[3] It is true that the GEN function in the Albro's, Karttunen's, and Riggle's methods is not exactly the same as the one assumed in Correspondence Theory (McCarthy and Prince, 1995), but it is instructive to understand why.

it is a precise, finite description of a potentially infinite set of strings.

A finite-state acceptor contains a finite set of states. We give the states names so we can talk about them; for instance they are often indexed with numbers. Some states are designated 'start' states. Some states are designated 'accepting' states. (States can be both 'start' and 'accepting' states.) Transitions lead from one state to another; they are labeled with letters from some alphabet.

So a finite-state acceptor is a finitely-sized collection of states and transitions. What is its extension? Well the extension is defined as follows. Informally, a word w is accepted/generated/recognized by a finite-state acceptor A if there is a path along the transitions of A which begins in a start state of A, which ends in a final state of A, and which spells out w exactly.

As an example, consider Figure 1.3, which shows the finite-state acceptor for the generalization in (1.1) that word-final vowels are prohibited. Per convention, the start state is designated by the unanchored incoming arrow and final states are marked with a double perimeter. The word *nok*

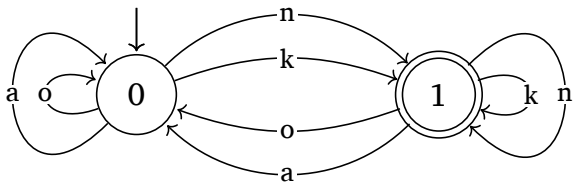


Figure 1.3: A finite state acceptor for the generalization "Word final vowels are prohibited." A simple alphabet {n,k,a,o} is assumed.

is generated by this machine since there is a path beginning in a start state and ending in a final state which spells it out. This path is shown below.

Input:		n		o		k	
States:	0	→	1	→	0	→	1

A minute of inspection reveals that every path for every word which ends in a vowel ends in state 0, which is not an accepting state. But every path for every word which does not end in a vowel ends in state 1, which is

accepting. Algorithms which solve membership problems for finite-state acceptors are well understood (Kozen, 1997; Hopcroft *et al.*, 2006; Sipser, 2012).

Finite-state automata are not limited to acceptors. String-to-string functions can be described with automata that are called **transducers**. These are acceptors whose labels have been augmented with an additional coordinate. Instead of a single symbol, Labels are now symbols paired with strings. Figure 1.4 shows the finite-state transducer for the generalization that word-final vowels delete. As before, valid paths through this machine (those that begin in start states and end in accepting states) spell out input words and the output words they map to. In the figure, the colon separates the left coordinate (input) from the right coordinate (output). The symbol λ denotes the empty string. To illustrate, consider the path which shows

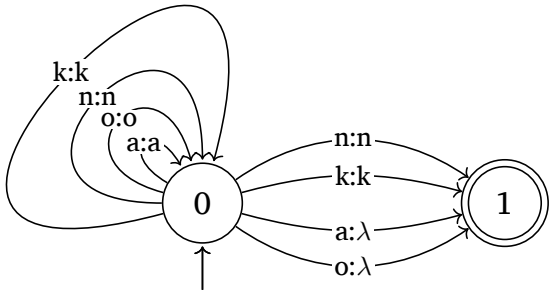


Figure 1.4: A finite state trasnsducer for the generalization "Word final vowels delete." A simple alphabet {n,k,a,o} is assumed.

that the output of *nako* is *nak*.

Input:		n		a		k		o	
States:	0	→	0	→	0	→	0	→	1
Output:		n		a		k		λ	

As with the membership problem and finite-state acceptors, there are algorithms which solve the generation and recognition problems for finite-state transducers.

There are some interesting things to observe about the finite-state transducer in Figure 1.4. The first is that it is non-deterministic. This means for a given input, there may be more than one path. For instance,

the input /kon/ maps to [kon], and there are two paths that spell it out. But only one is valid: the one that reads and writes n and moves from state 0 to state 1.[4]

Another point is that the transducer in Figure 1.4 maps the input word /nakao/ to [naka]. As such, this machine is a formal description of the extension of the rule $\text{V} \longrightarrow \varnothing \; / \; \_\_ \; \#$ applying simultaneously. In OT, if FINAL-C outranks MAX, then the output would be *nak* with the last two vowels deleting. With rules, this could be accomplished by applying the aforementioned rule right-to-left. The finite-state transducer shown in Figure 1.5 realizes this mapping. For readability, distinct transitions with the same origin and destination are shown as multiple labels on a single arrow.

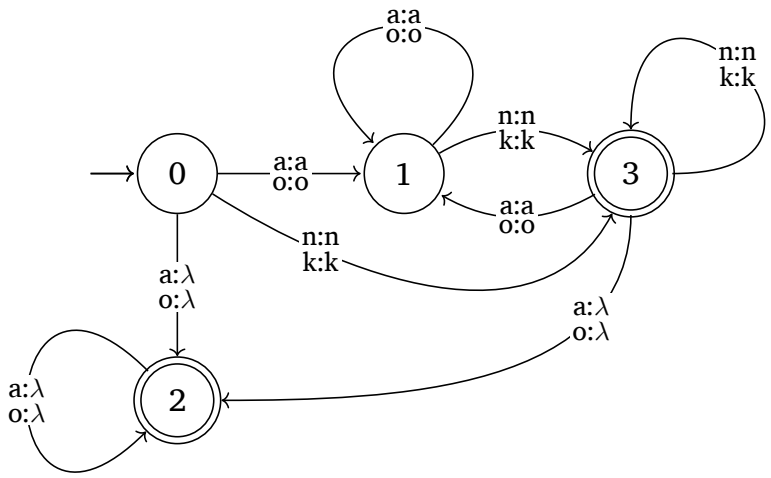


Figure 1.5: A finite state transducer for the generalization "Strings of vowels word-finally delete." A simple alphabet {n,k,a,o} is assumed.

Transducers can also map strings to numbers. The simple one shown in Figure 1.6 counts the number of *o*s in a word. The idea here is that instead of combining the outputs of valid paths with *concatenation* as for strings, they are combined with *addition*. Below is an example of the only valid path for the word *naoko* which would be mapped to 2.

[4]Non-determinism is one way optionality can be handled with finite-state transducers. If state 0 was also an accepting state then there would be two valid paths for the input /noko/. One path would yield the output [noko] and the other the output [nak].

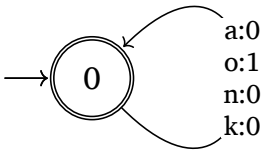


Figure 1.6: A finite state transducer which counts the number of *o*s in words. A simple alphabet {n,k,a,o} is assumed.

Input:		n		a		o		k		o	
States:	A	→	A	→	A	→	A	→	A	→	A
Output:		0		0		1		0		1	

This is the approach used by Riggle (2004) to define markedness and faithfulness constraints in OT. There are many generalizations of this kind available to transducers made possible by the study of semirings (Roark and Sproat, 2007; Droste and Kuich, 2009; Goodman, 1999). Semirings are discussed in more detail in Chapter 4. However the main point I wish to express is that the extension of the transducers discussed so far are all precisely defined and the corresponding generation problems solvable.

What of the recognition problem? Another important advantage of finite-state automata is that they are **invertible**. Consequently, a solution to the generation problem entails a solution to the recognition problem. Given a string *nak*, the transducer can tell you that it is the output of the each of the following inputs: *nak, naka, nako*.

Nonetheless, despite the advantages well-defined extensions bring, there are some shortcomings to using finite-state automata for phonological analyses. One is that letters of the alphabet are treated atomically. For instance, there is no sense in which the symbols [p,t,k] share any properties. It remains unclear how to incorporate phonological features and natural classes in a natural way into these machines. The most common way seems to just group the letters together that behave together as I have done in the examples above. While this is certainly sufficiently expressive, it may not be completely satisfying. We want our intensional descriptions to somehow speak directly to the descriptive ones. In the case of "Word final vowels are prohibited" we want to be able to express the relevant natural class directly.

Another drawback is that as the generalizations become more complex, so do the finite-state automata. They become spaghetti-like and difficult to read. This drawback is mitigated, however, in a couple of ways. The first is that it is very well understood how to combine different finite-state automata to produce new ones. This allows the generalizations instantiated by the 'primitive' ones to persist to some degree in the complex ones. For instance, it is straightforward to construct a finite-state acceptor that generates exactly the intersection of two infinite sets of strings which are generated by two acceptors. (Heinz (2014) provides concrete examples in the domain of stress.) Similarly, it is straightforward to construct a finite-state transducer that generates the composition of two functions which are generated by finite-state transducers (Roark and Sproat, 2007; Gorman and Sproat, 2021). In this way, more complex finite-state automata can be constructed from simpler parts, much in the same way more complex phonological grammars are built up from identifying generalizations that interact in some manner.

A third problem is that even simple machines are not easy to write in text. They are often pictured as diagrams, and in the same way it can be tiring to read them, it can be tiring to draw them as well. This problem is mitigated in a couple of ways. First, there are helpful software packages which can automatically draw machines, like GraphViz.[5] Some researchers use tables or matrix notation, others use types of regular expressions (Beesley and Karttunen, 2003; Hulden, 2009b; Lambert, 2022), and still others use logic.

In this book, we are going to use logic and not automata to represent linguistic generalizations. There are several reasons for this. Most importantly, like automata, the extensions of logical formula are precisely defined. Another key reason is that the representations are *flexible*. We can represent words exactly as any phonologist would want. As this book will show, phonological features, syllable structures, autosegmental representations, hand shapes, phonetic information, and a host of as-yet-unconsidered possibilities are available and directly representable with logic. Thirdly, as this book will show, the combination of logical power and representation provides a natural way to entertain *distinct* theories of phonology and compare them. Additionally, there is a literature showing how logical formula can be translated into automata which are equivalent in the sense that they solve the same membership, generation, and recognition problems.

[5] `https://graphviz.org`.

While this literature does not address every phonological representation proposed, the basic analytical methods which show how this can be done for strings and trees are there. As long as the phonological representations the analyst uses can be encoded as strings, the translations to automata are possible.

Finally, logic is not going anywhere. This is very important. If a linguist describes a generalization with logical experessions using the representations they prefer, they can be guaranteed that people in will be able to read their description and understand it *hundreds of years later*.

In short, logical formula have all of the advantages, and none of the disadvantages, of automata.

1.5 Doing Computational Phonology

How does one do computational generative phonology? This book provides an answer.

In the first part, logical foundations and model theory are presented in the context of strings. It is explained how model theory allows one to precisely formulate different representations of words and phrases. It is explained how the primitive elements in these representations would have ontological status in the theory. It is also explained how logical expressions can be used to define constraints to delimit possible representations in words and phrases, and how they can also define possible transformations which map one representation to another. It is explained how **weighted** logical expressions allow one to express a variety of linguistic generalizations, including gradient ones, if desired. These definitions and techniques are illustrated with examples drawn from phonology, as well as examples showing the terrific expressivity of the framework. The first part of this books opens a large window into the techniques and possibilties.

In the second part, these techniques are applied to the kinds of phonology problems one finds in standard textbooks on phonology. The focus here is descriptive in the following sense. Linguists marshall arguments from a collection of linguistic forms they have before them in favor of particular linguistic generalizations. These arguments are presented and then the linguistic generalizations are formalized in terms of model-theoretic representations and logic. The chapters are short, each dealing with one relatively small and straightforward phonological problem. These exam-

ples serve as models for how analysis of other small and straightforward phonological problems can be analyzed within CGP.

In the third part, the chapters address a variety of theoretical issues addressing both aspects of representation and computional power. Sebastian shows how to incorporate insights from phonetically-based phonology into CGP representationally. Hwangbo shows how representing vowel height in terms of degrees of aperture leads to straightforward analysis of vowel lowering in a language like Danish. Strother-Garcia analyzes syllable structure and the sonority sequencing principle. Lambert and Rogers show how the stress patterns in the world's languages can be understood as a particular combination of primitive constraints. They further characterizes the complexity of those constraints. Lindell and Chandlee provide a logical characterization of Input Strictly Local functions, which Chandlee showed earlier to well-characterize an important natural class of phonological transformations. Dolatian shows that the Raimy-style linearization is computationally actually very complex. Having identified the source of complexity, he suggests way to mitigate it. Payne provides similar results for the comptuational complexity of GEN. Vu shows how transformations can also be expressed as constraints on correspondence structures. These chapters are but a small sample of the kinds of research questions and investigations that can be addressed with the tools introduced in part one. **TODO: update these mentions and add mentions to Rawski's chapter, Nelson's chapter.**

Computational generative phonology is simple. It is not hard. We believe theories of generative phonology developed in this tradition will lead to advances in our understanding of the nature of phonological grammars and the minds which know them.

Chapter 2

Representations, Models, and Constraints

JEFFREY HEINZ AND JAMES ROGERS

2.1 Logic and Constraints in Phonology

In this chapter, we show how to use logic and model-theoretic representations to define well-formedness condtitions over phonological representations (such as markedness constraints). The power in this kind of computational analysis comes from the framework's flexibility in both the kind of logic used and the choice of representation.

As will be explained, these choices provide a "Constraint Definition Language" (CDL) in the sense of (de Lacy, 2011). A CDL is a language with a formal syntax and semantics, with which one can precisely define constraints and with which one can interpret those constraints with respect to representations. Each CDL has consequences for typology, learnability, and the psychology of language, which can be carefully studied. Conversely, psychological, typological, and learnability considerations provide evidence for the computational nature of phonological generalizations on well-formedness; that is for the choices we can make.

This is not the first effort to apply logic to phonological theory. In fact, there is considerable history. A notable turning point occurred in the early 1990s with the developments of two theories: Declarative Phonology and Optimality Theory.

Declarative Phonology made explicit use of logical statements in describing the phonology of a language. For instance (Scobbie *et al.*, 1996, p. 688) expressed a general principle of theories of syllables which prohibit ambisyllabicity this way: $\forall x \neg(\text{onset}(x) \wedge \text{coda}(x))$, which in English reads "For all segments x, it is not the case that x is both an onset and a coda."

In Optimality Theory, first-order logic was often used implicitly to define constraints. For example, the definition of the constraint MAX-IO in OT given by McCarthy and Prince (1995, p. 16) is "Every segment of the input has a correspondent in the output." On page 14, they define the correspondence relation: "Given two strings S_1 and S_2, correspondence is a relation R from the elements of S_1 to those of S_2. Elements $\alpha \in S_1$ and $\beta \in S_2$ are referred to as **correspondents** of one another when $\alpha R \beta$." As will be clear by the end of this chapter, this definition of MAX-IO is essentially a statement in First Order Logic: For all $\alpha \in S_1$ there exists $\beta \in S_2$ such that $\alpha R \beta$.

Unlike Optimality Theory, the CDLs introduced in this chapter are assumed to provide language-specific, inviolable constraints. For a representation to be well-formed it must not violate any constraint. This is a property the CDLs in this chapter have in common with Declarative Phonology. Scobbie et al. explain:

> The actual model of constraint interaction adopted is maximally simple: the declarative model. In such a model, all constraints must be satisfied. The procedural order in which constraints are checked (or equivalently, in which they apply) is not part of the grammar, but part of an implementation of the grammar (as a parser, say) which cannot affect grammaticality. (Scobbie *et al.*, 1996, p. 692)

What Scobbie et al. are emphasizing is that logical specifications of grammar specify *what is being computed* as opposed to *how it is being computed*. We agree with Scobbie *et al.* (1996) that this is an attractive property of logical languages.

While this chapter, and others in this book, assume the constraints are language-specific and inviolable, it is a mistake to conclude that this line of work only applies to grammars that make binary distinctions between well-formed and ill-formed structures. In fact, the model-theoretic and logical framework advocated here can also describe gradient well-formedness with **weighted logical languages**. These allow one to specify what is

being computed when linguistic representations are assigned numbers of violations of a constraint, as in the case in Optimality Theory when evaluating candidates, or real numbers, as in the case of assigning some probabilities to structures (Droste and Gastin, 2009). This chapter does not discuss weighted logical languages, but they are reviewed with some examples in Chapter 4.

2.2 Chapter Outline

In the remainder of this chapter, we informally introduce model-theoretic representations of strings and different logics. We focus on strings because they are widely used and well-understood. Most importantly, they are sufficient to illustrate how different CDLs can be defined and how these CDLs have consequences for psychological and typological aspects of language as well as learnability. Several chapters later in the book provide concrete examples of non-string representations motivated by phonological theory. **Add forward references to autosegmental representations (Chapter XYZ), syllable structure (Chapter XYZ), morphological representations, gradient phonetic representations, etc.)**

A formal, mathematical treatment of the representations and logic is given in Chapter 6. Concepts and definitions introduced here are presented there precisely and unambiguously. Some readers may benefit by consulting this chapter in parallel with that one.

Essentially, this chapter compares several CDLs by varying the representation of words (called models) and the logical language (First Order vs Monadic Second Order). The models we consider vary along two dimensions: the representation of speech sounds (segments vs feature bundles), and the representation of order (successor vs precedence).

The first model we introduce is the canonical word model, which is known as the successor model. This is followed by an informal treatment of First-Order (FO) logic. This yields the first CDL we consider (FO with successor) and we show how to define a constraint like *NT—voiceless obstruents are prohibited from occurring immediately after nasals—in this CDL.

Next we alter the successor model so that the representations make use of phonological features. This yields another CDL (FO with successor and features). We comment on some notable points of comparison between the

two CDLs, again using the *NT constraint.

The narrative continues by discussing one typological weakness of the aforementioned CDLs: they are unable to describe long-distance constraints which are arguably part of the phonological competence of speakers of some languages. This provides some motivation for a CDL defined in terms of a more powerful logic, Monadic Second Order (MSO) logic. This CDL we call 'MSO with successor and features,' and we explain how it is able to define such long-distance constraints. The key is that with MSO logic it is possible to deduce that one element in a string *precedes* another element, no matter how much later the second element occurs. The availability of the precedence relation makes it possible to define long-distance constraints.

We continue to evaluate the MSO with successor CDL from a typological perspective. We argue that there are significant classes of constraints definable in this CDL that are bizarre from a phonological perspective. An example is the constraint which forbids words to have evenly many nasals (*Even-N). In other words, we motivate seeking a more restrictive CDL which is still capable of describing local and long-distance constraints in phonology.

One solution we consider is to make the precedence order a primitive relation of the representation. This model of words is called the precedence model, which stands in contrast to the successor model. We show how the CDL "FO with precedence and features" is also able to describe both local and long-distance constraints of the kind found in the phonologies of the world's languages and excludes (some) of the bizarre constraints that the CDL 'MSO with successor and features' is able to describe.

Finally, the chapter concludes with a high-level discussion seeking to emphasize the following points. First, there is a tradeoff between representations and logical power. Second, as mentioned, the choice of representation and the choice of logic has consequences for typology, psychological reality, memory, and learnability. Third, the representations and logics discussed in this chapter are only the tip of the iceberg. Readers undoubtedly will have asked themselves "What about this possible representation?" and "Why don't we consider this variety of logic?" Later chapters in this book address some such questions. Comprehensively answering such questions, however, is beyond the scope of this book. But it is not beyond the scope of phonological theory. If some readers of this book pose and answer such questions, then this book will have succeeded in its goals.

2.3 The Successor Model

This section introduces the central ideas of model-theoretic representations with a concrete example. The concrete example comes from the "successor" model, which is one of the canonical model-theoretic representations for strings.

Model-theoretic representations provide a uniform framework for representing all kinds of objects. Here the objects under study are strings. We need to be clear about two things: what the objects are, and what counts as a successful model-theoretic representation of a set of objects.

Strings are sequences of events. If we are talking about words, the events could be given as speech sounds from the International Phonetic Alphabet, or as gestural events in speech or sign, or as perceptual landmarks in auditory space or visual space. In this chapter, we consider models of strings over the following alphabet of IPA symbols: *a, b, d, e, g, i, k, l, m, n, o, p, r, s, t, u, z*. Limiting this alphabet in this way is pedagogically useful, and it will be clear that it can be expanded as needed for one's purpose. In general, the set of alphabetic symbols is denoted Σ and Σ^* denotes the set of all possible sequences of finite length that can be constructed from symbols in Σ.

A successful model theoretic-representation of a set of objects must provide a representation for each object and must provide distinct representations for distinct objects. It may be strange to ask the question "How can we represent strings?" After all if we are talking about the string *sans*, isn't *sans* itself a representation of it? It is, but the information carried in such representations is implicit. Model-theoretic representations make the information explicit.

Model-theoretic representations for objects of finite size like strings are structures which contain two parts. The first is a finite set of elements called the **domain**, written D. The second is a finite set of relations $\mathfrak{R} = \{R_1, R_2, \ldots R_n\}$. The relations provide information about the domain elements and how those elements relate to each other. These relations constitute the **signature** of the model. In this book, a model-theoretic representation with signature $\mathfrak{R}$ is called an $\mathfrak{R}$**-structure**, and it is written like this: $\langle D \mid R_1, R_2, \ldots R_n \rangle$.

We first show a model-theoretic representation of a word and then we explain it. While this may seem backwards to some, it seems to work better pedagogically. It can be helpful to refer to the end-product as one goes

about explaining how one got there.

Figure 2.1 shows the successor structure for the word *sans* in addition to a graphical diagram of it on its right. The graphical diagram puts the domain elements in circles. Edges labeled with $\triangleleft$ indicate the binary relation called "successor." Finally, the unary relations, one for each symbol in the alphabet, are shown in typewriter font above the domain elements that belong to them. Throughout this book we will often use graphical diagrams instead of displaying the literal mathematical representation on the left. The order of the relations in the signature is fixed but it is also arbitrary.

$$
\begin{aligned}
&\mathcal{M}_{tent} \\
&= \langle D \mid \mathtt{s}, \mathtt{a}, \mathtt{n}, \mathtt{b}, \mathtt{d}, \ldots, \mathtt{z}, \triangleleft \rangle \\
&= \Big\langle \{1,2,3,4\} \mid \{1,4\}, \{2\}, \{3\}, \\
&\quad \varnothing, \varnothing, \ldots \varnothing, \\
&\quad \{(1,2),(2,3),(3,4)\} \Big\rangle
\end{aligned}
$$

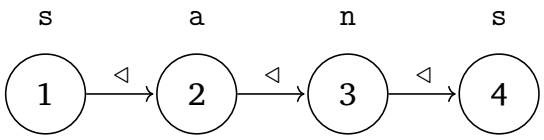


Figure 2.1: At left, the successor model of the word *sans*. At right, a graphical diagram of this model.

In the case of strings, the number of domain elements matches the length of the string. So a model-theoretic representation of a word like *sans* would have a domain with four elements, one for each event in the sequence. We can represent these domain elements with the suits in a deck of cards $\{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$ or we could use numbers $\{1, 2, 3, 4\}$ as we did in Figure 2.1. We will usually use numbers because as strings get longer we can always find new numbers. However, keep in mind that the numbers are just names of elements in the model in the same way the suits would have been. They get their meaning from the relationships they stand in, not from anything inherent in the numbers themselves.

In the signature for successor structures, for each symbol b in the alphabet, there is a unary relation $\mathtt{b}$. In Figure 2.1, the alphabet is the limited set of IPA symbols mentioned previously. We use the typewriter font to distinguish the relations from the symbols. We write $(\mathtt{b})_{b \in \Sigma}$ to mean this finite set of relations. For each $b \in \Sigma$, if a domain element belongs to the unary relation $\mathtt{b}$ then it means this element has the property of being b.

Regarding the word *sans*, the relational structure shown in Figure 2.1 indicates that there are four distinct elements. Two of them belong to s; a different element belongs to a; and the remaining element belongs to n. For every symbol $b \in \Sigma - \{s, a, n\}$, the relation b is empty. For all $x \in D$ and all $b \in \Sigma$, when we write $x \in \mathtt{b}$ or $\mathtt{b}(x)$ we mean that domain element x belongs to the unary relation b.

The signature of the successor model also includes a single binary relation called "successor". A domain element x indicating some event stands in the successor relation to y if y corresponds to the event which is in fact the *next* event after x. In this book, we use the symbol $\lhd$ to indicate the successor relation. For the word *sans*, if $2 \in R_a$ and $3 \in R_n$ then $(2, 3)$ would be in the successor relation. There are at least three common ways to write the fact that domain elements 2 and 3 stand in the successor relation: $(2, 3) \in \lhd$ (set notation), $\lhd(2, 3)$ (prefix notation), and $2 \lhd 3$ (infix notation).

The signature $\mathfrak{R}$ for the successor model is thus $\{(\mathtt{b})_{b\in\Sigma}, \lhd\}$ and $\mathfrak{R}$-structures would have the form $\langle D \mid (\mathtt{b})_{b\in\Sigma}, \lhd\rangle$. It is also customary to use salient aspects of the signature to refer to the signature itself. In the case of the successor model, it is the successor relation that plays a critical role. For this reason, we will refer to structures with the aforementioned signature $\mathfrak{R}$ as $\lhd$-structures.

The successor model is not the only way to represent words. From a phonological perspective, it is arguably a strange model. After all, there are no phonological features! We will consider more phonologically natural models of words below.

It is easy to see that there is a general method for constructing a unique model for each logically possible string. Given a string w of length n we can always construct a successor model for it as follows. Since w is a sequence of n symbols, we let $w = b_1 b_2 \dots b_n$. Then set the domain $D = \{1, 2, \dots n\}$. For each symbol $b \in \Sigma$ and i between 1 and n inclusive, $i \in \mathtt{b}$ if and only if $b_i = b$. And finally, for each i between 1 and $n - 1$ inclusive, let the only elements of the successor relation be $(i, i + 1)$.[1] This is summarized in Table 2.1.

This construction guarantees the soundness of the successor model: each string has one structure and distinct strings will have distinct structures. It is also important to recognize that removing any one of the unary or binary relations will result in a signature which does not guarantee that

[1] Here we are taking advantage of the numeric interpretation of the domain elements.

D	$\stackrel{\text{def}}{=}$	$\{1, 2, \ldots n\}$
b	$\stackrel{\text{def}}{=}$	$\{i \in D \mid b_i = b\}$ for each unary relation b
$\lhd$	$\stackrel{\text{def}}{=}$	$\{(i, i+1) \subseteq D \times D\}$

Table 2.1: Creating a successor model for any word $w = b_1 b_2 \ldots b_n$.

models of distinct strings are distinct.

Model-theoretic representations provide an ontology and a vocabulary for talking about objects. They provide a primitive set of facts from which we can reason. For instance in the word *random*, we know that the *m* occurs sometime after the *n*. However this fact is not immediately available from the successor model. It can be deduced, but that deduction requires some computation. Measuring the cost of such computations is but one facet of what model theory accomplishes. On the other hand, the successor model makes immediately available the information that *d* occurs immediately after the *n*. As will hopefully be clear by the end of this chapter, this distinction can shed light on differences between local and long-distance constraints in phonology.

From a psychological perspective, the primitive set of facts a model-theoretic representation encodes about a word can be thought of as primitive psychological units. In its strongest form, the model-theoretic representation of words as embodied in its signature makes a concrete claim about the psychological reality of the ways words are represented mentally.

2.4 First Order Logic

Now that the models provide explicit representations, what do we do with them? Logic provides a language for talking about these representations. First Order logic is a well-understood logical language which we introduce informally here. For those already familiar with FO logic, you will see that we take advantage of things like prenex normal form without discussion.[2]

In addition to the Boolean connectives such as conjunction, disjunction, implication, and negation, FO logic also includes existential and universal

[2]Readers are referred to Keisler and Robbin (1996); Enderton (2001) and Hedman (2004) for complete treatments of first order logic including prenex normal form.

quantification over variables that range over domain elements. These variables are called **first order variables**. Apart from these logical connectives and quantified variables, the basic vocabulary of FO logic comes from the *relations in the signature*. Thus each model-theoretic representation supplies essential ingredients for the logical language. Table 2.2 summarizes the vocabulary of FO logic with an arbitrary model $\langle D \mid R_1, R_2, \ldots R_n \rangle$. The expressions in the category "Model Vocabulary" in Table 2.2 are also called **atomic formulas** because they are the primitive terms from which larger logical expressions are built. In other words, not only can a signature $\mathfrak{R}$ give rise to model theoretic representations of a class of objects, but a signature $\mathfrak{R}$ also gives rise to a first-order **logical language**. This language is made up of the expressions that can be built from the model vocabulary and the logical connectives and quantifiers in a syntactically valid way. We call this logical language FO($\mathfrak{R}$).

Since Chapter 6 defines FO logic formally, here we introduce the concept of valid sentences and formulas of FO logic ostensively. Below we give examples of three types of expressions: sentences of FO logic, formulas of FO logic, and syntactically ill-formed expressions. Sentences of FO logic are complete, syntactically valid sentences that can be interpreted with respect to a signature. Formulas of FO logic are syntactically valid expressions, but are not complete in the sense that they contain variables which are not bound to anything. The sentences and formulas that belong to a logical language FO($\mathfrak{R}$) will be called $\mathfrak{R}$-sentences and $\mathfrak{R}$-formulas, respectively. The syntactically ill-formed expressions demonstrate common ways expressions are incorrectly stated.

Example 1 (Sentences of FO($\lhd$)**.** Below are five $\lhd$-sentences of FO logic with English translations below.[3]

1. Sentences of FO logic.
   (a) $\exists x, y, z\ (\neg(x = y) \wedge \neg(x = z) \wedge \neg(y = z))$
   (b) $\exists x, y\ (\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \lhd y)$
   (c) $\neg\exists x, y\ (\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \lhd y)$
   (d) $\forall x, y\ (\ \neg(\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \lhd y))$
   (e) $\forall x \exists y\ (\mathtt{n}(x) \rightarrow (\mathtt{t}(y) \wedge x \lhd y))$

[3] In the examples, we use the word 'element' to refer to domain elements. However, since we are talking about strings, we could have equally well used words like 'event' or 'position'.

Boolean Values	
`true`	true
$\top$	true
`false`	false
$\bot$	false
Boolean Connectives	
$\wedge$	conjunction
$\vee$	disjunction
$\neg$	negation
$\rightarrow$	implication
$\leftrightarrow$	biconditional
Syntactic Elements	
(	left parentheses
)	right parentheses
,	comma for separating variables
Variables, Quantifiers, and Equality	
x, y, z	variables which range over elements of the domain
$\exists$	existential quantifier
$\forall$	universal quantifier
$=$	equality between variables
Model Vocabulary	
$\mathtt{R}(x)$	for each unary relation R in $\{R_1, R_2, \ldots R_n\}$
$\mathtt{R}(x, y)$	for each binary relation R in $\{R_1, R_2, \ldots R_n\}$
$x\mathtt{R}y$	for each binary relation R in $\{R_1, R_2, \ldots R_n\}$
...	
$\mathtt{R}(x_1, x_2 \ldots x_m)$	for each m-ary relation R in $\{R_1, R_2, \ldots R_m\}$

Table 2.2: Symbols and their meaning in FO logic. Certain sequences of these symbols are valid FO sentences and formulas. Note binary relations are often written in two ways.

2. Literal English translation.

- There exist elements x, y, z such that x is not y, x is not z, and y is not z.
- There exist elements x, y such that x satisfies property `n`, y satisfies property `t`, and y is the successor of x.
- There does not exist elements x, y such that x satisfies property `n`, y satisfies property `t`, and y is the successor of x.
- For all elements x, y it is not the case that x satisfies property `n`, y satisfies property `t`, and y is the successor of x.
- For all elements x, there is element y such that if x satisfies property `n` then y satisfies property `t` and y is the successor of x.

3. English translation (in terms of the models).

   (a) There are three distinct domain elements.
   (b) There are two domain elements in the successor relation; the former has the property of being *n*; the latter has the property of being *t*.
   (c) It is not the case that there exists two domain elements in the successor relation of which the former has the property of being *n* and the latter has the property of being *t*.
   (d) For every pair of domain elements that stand in the successor relation, it is not the case that the former has the property of being *n* and the latter has the property of being *t*.
   (e) For all domain element which have the property of being *n*, it is succeeded by a domain element which has the property of being *t*.

4. English translation (in terms of the strings the models represent).

   (a) There are at least three symbols.
   (b) There is a substring *nt*.
   (c) There is no substring *nt*.
   (d) There is no substring *nt*.
   (e) Every *n* is immediately followed by *t*.

$\mathfrak{R}$-**sentences** of FO logic are **interpreted** with respect to $\mathfrak{R}$-structures. A structure for which the sentence is true are said to **satisfy** the sentence. If a structure (or model) $\mathcal{M}$ of string w satisfies a sentence ϕ we write

$\mathcal{M}_w \models \phi$. Consequently, every FO sentence ϕ divides the objects being modeled into two classes: those that satisfy ϕ and those that do not. In this way, logical sentences define **constraints**. The strings whose models satisfy the sentence do not violate the constraint; strings whose models do not satisfy the constraint do violate it.

Table 2.3 provides examples of strings whose models satisfy the formulas in Example 1 and examples of strings whose models do not. An important

ϕ	$\mathcal{M}_w \models \phi$	$\mathcal{M}_w \not\models \phi$
(a)	*too, sans, ttt*	*to, a*
(b)	*sant, rent, ntnt*	*ten, to, phobia*
(c)	*ten, to, phobia*	*sant, rent, ntnt*
(d)	*ten, to, phobia*	*sant, rent, ntnt*
(e)	*rent, antler*	*ten, nantucket*

Table 2.3: Some strings whose models satisfy the formulas in Example 1 and some whose models do not.

feature of FO logic is that there are algorithmic solutions to the problem of deciding whether a given $\mathfrak{R}$-structure satisfies a given $\mathfrak{R}$-sentence. This algorithm works because the syntactic rules that build up larger sentences from smaller ones have clear semantic interpretations with respect to the structure under consideration. In short, it is an unambiguous and compositional system. For instance, $\mathcal{M} \models \phi \wedge \psi$ if and only if $\mathcal{M} \models \phi$ and $\mathcal{M} \models \psi$. The interpretation of quantifiers is discussed after introducing formulas below.

$\mathfrak{R}$-**formulas** of FO logic are incomplete sentences in the sense that they contain variables that are not **bound**. A variable is bound only if it is has been introduced with a quantifier and is within that quantifier's scope. Variables that are not bound are called **free**. $\mathfrak{R}$-formulas are only interpretable with respect to an $\mathfrak{R}$-structure $\mathcal{M}$ if the free variables are assigned some interpretation as elements of the domain of $\mathcal{M}$.

Example 2 (Formulas of FO($\lhd$))**.** 1. Formulas of FO logic.

(a) $\mathtt{n}(x) \vee \mathtt{m}(x) \vee \mathtt{ŋ}(x)$
(b) $\exists y\ (\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \lhd y)$
(c) $\neg\exists y\ (x \lhd y)$

(d) $\neg\exists y\ (y \lhd x)$
(e) $\neg(x = y) \wedge \neg(x = z) \wedge \neg(y = z)$
(f) $x \lhd y \wedge y \lhd z$

2. English translation.

(a) x has the property of being ***n***, ***m***, or $\mathtt{ŋ}$.
(b) x has the property of being ***n*** and coming immediately before an element which has the property of being ***t***.
(c) There is no element which succeeds x.
(d) There is no element which x succeeds.
(e) x, y and z are distinct.
(f) x is succeeded succeeded by y which is succeeded by z.

The difference between formulas and sentences is that sentences admit no free variables. Since sentences have no free variables, they must begin with quantifiers. Because formulas can only be interpreted in terms of one or more un-instantiated variables, formulas are often used to define **predicates**. Predicates are essentially abbreviations for formulas with the unbound variables serving as parameters. Below we repeat the formulas from above, but use them to define new predicates. We also write predicates in typewriter font, but with a very light gray highlight to distinguish them from atomic formulas.

$$\texttt{nasal}\ (x) \stackrel{\text{def}}{=} \mathtt{n}(x) \vee \mathtt{m}(x) \vee \mathtt{ŋ}(x) \tag{2.1}$$

$$\texttt{nt}\ (x) \stackrel{\text{def}}{=} \exists y\ (\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \lhd y) \tag{2.2}$$

$$\texttt{last}\ (x) \stackrel{\text{def}}{=} \neg\exists y\ (x \lhd y) \tag{2.3}$$

$$\texttt{first}\ (x) \stackrel{\text{def}}{=} \neg\exists y\ (y \lhd x) \tag{2.4}$$

$$\texttt{distinct3}\ (x, y, z) \stackrel{\text{def}}{=} \neg(x = y) \wedge \neg(x = z) \wedge \neg(y = z) \tag{2.5}$$

$$\texttt{string3}\ (x, y, z) \stackrel{\text{def}}{=} x \lhd y \wedge y \lhd z \tag{2.6}$$

These predicates can then be used to define new expressions. For example, the sentence $\forall x(\neg\ \texttt{nt}\ (x))$ is equivalent to (1d) in Example 1 above. In the same way that programmers write functions which encapsulate snippets of often-used programming code, predicates generally help writing and reading complex logical expressions.

Determining whether a structure satisfies a sentence is compositional. It also depends on the **assignment** of variables to elements in the model's domain. For instance, to determine whether $\mathcal{M}$ satisfies $\phi = \exists x(\psi(x))$, we must find an element of the domain of $\mathcal{M}$, which if assigned to x, has the consequence that ψ evaluates to `true`. If no such element exists, then $\mathcal{M}$ does not satisfy ϕ. Similarly, $\mathcal{M}$ satisfies $\phi = \forall x(\psi(x))$ if and only if every element of the domain $\mathcal{M}$, when assigned to x, results in ψ evaluating to `true`. The formal semantics of FO logic is given in Chapter 6.

Finally we give some examples of syntactically ill-formed sequences. The following expressions are junk; they are not interpretable at all.

Example 3 (Syntactically ill-formed sequences)**.**

1. Syntactically ill-formed sequences.
   (a) $x\exists\ )x($
   (b) $\forall\exists\ (\mathtt{n} \vee \mathtt{t})$
   (c) $\neg\exists(\mathtt{n} \lhd \mathtt{t})$
2. Comments.
   (a) Quantifiers always introduce variables to their left and parentheses are used normally.
   (b) No quantifier can be introduced without a variable and n-ary relations from the model vocabulary must always include n variables.
   (c) Many beginning students make this sort of error when trying to express a logical sentence which forbids *nt* sequences. This expression breaks the same rules as the one before it.

We conclude this section by providing an example of a logical sentence defining a constraint which bans voiceless obstruents after nasals. This is a constraint in the literature often abbreviated *NT (Pater, 1999). Since the model signature does not include relations for concepts like nasals and voiceless consonants, we first define predicates for these notions.

Example 4 (The constraint *NT defined under the FO with successor model)**.**

$$\texttt{nasal}\,(x) \stackrel{\text{def}}{=} \texttt{n}(x) \vee \texttt{m}(x) \quad (2.7)$$

$$\texttt{voiceless}\,(x) \stackrel{\text{def}}{=} \texttt{p}(x) \vee \texttt{t}(x) \vee \texttt{k}(x) \vee \texttt{s}(x) \quad (2.8)$$

$$\texttt{*NT} \stackrel{\text{def}}{=} \neg\exists x, y(x \lhd y \wedge \texttt{nasal}\,(x) \wedge \texttt{voiceless}\,(y)) \quad (2.9)$$

It is easy to see that $\lhd$-structures of words like *sans* and *lampoon* do not satisfy $\texttt{*NT}$ but $\lhd$-structures of words like *ten* and *moon* do. For example, in the $\lhd$-structure of *sans*, the expression $\exists x, y(x \lhd y \wedge \texttt{nasal}\,(x) \wedge \texttt{voiceless}\,(y))$ is true when $x = 3$ and $y = 4$. Hence, $\texttt{*NT}$ evaluates to `false`. On the other hand, in the $\lhd$-structure of the word *moon*, every value assigned x and y results in the sentence $\exists x, y(x \lhd y \wedge \texttt{nasal}\,(x) \wedge \texttt{voiceless}\,(y))$ evaluating to `false`. Hence the sentence $\texttt{*NT}$ evaluates to `true` and so $\mathcal{M}_{moon} \models \texttt{*NT}$.

This section has presented the first CDL: FO with successor, also written as FO($\lhd$). The FO with successor model has been studied carefully and it is known precisely what kinds of constraints can and cannot be expressed with this CDL (Thomas, 1982), as will be discussed further below.

2.5 Word Models with Phonological Features

One way in which the successor model above is strange from a phonological perspective is its absence of phonological features. The properties associated with the elements of the domain are singular, atomic segments. However, nothing in model theory itself prohibits domain elements from having more than one property. It is a consequence of the construction in Table 2.1 that each domain element will satisfy exactly one of the unary relations $\texttt{b}$, no more and no less. We can formalize this statement of the successor model in Remark 1 as follows.

Remark 1 (The successor model entails disjoint unary relations)**.** For all $\lhd$-structures $\mathcal{M} = \langle D \mid (\texttt{b})_{b\in\Sigma}, \lhd\rangle$, and for all $\texttt{a}, \texttt{b} \in (\texttt{b})_{b\in\Sigma}$, it is the case that $\texttt{a} \cap \texttt{b} = \varnothing$.

It is possible to design different models of words, where the unary relations do not represent segments like *a, b,* or *n* but phonetic or phonological features such as *vocalic, labial,* or *nasal*. Crucially, these models would not entail disjoint unary relations: a domain element could be both *voiced* and *labial* for instance.

In this part of the chapter, we give one example of such a model. There are many others, as many as there are theories of phonological features. The model we give here is primarily for pedagogical reasons; we are not stating particular beliefs or arguments regarding the nature of feature systems. We are only choosing a simple system that illustrates some key points.

We set up a feature system with **privative** features for the simple alphabet Σ discussed earlier *a, b, d, e, g, h, i, k, l, m, n, o, p, r, s, t, u, z*. The use of privative features contrasts with the typical assumption in phonological theory that features are **binary** (Hayes, 2009; Odden, 2014; Bale and Reiss, 2018). We choose not to pick a minimal nor maximal set of features for distinguishing this set. Instead we choose somewhat arbitrarily a middle ground based on standard descriptive phonetic terms used for describing the manner, place and laryngeal quality in articulating sounds. We call this model "the successor model with features." Its signature, which we denote as (feat,$\triangleleft$), is shown below.

$$\{\texttt{vocalic}, \texttt{low}, \texttt{high}, \texttt{front}, \texttt{stop}, \texttt{fricative}, \texttt{nasal}, \texttt{lateral}, \texttt{rhotic}, \texttt{voiced}, \texttt{voiceless}, \texttt{labial}, \texttt{coronal}, \texttt{dorsal}, \triangleleft\} \quad (2.10)$$

This contrasts with the successor model in the previous section, which we will call "the successor model without features," or sometimes "the successor model with letters." Table 2.4 shows how to construct a (feat,$\triangleleft$)-structure for any string in Σ^*. Again this model ensures that distinct strings from Σ^* have different models and that every string has some model.

As an example, Figure 2.2 shows the (feat,$\triangleleft$)-structure of the word *sans*.

The successor model with features contrasts sharply with the successor model without features in an important way. To see how, first consider the constraint *NT. Under the successor model with features, this constraint would be defined as in Equation 2.11

Example 5 (The constraint *NT defined under the FO with successor model

D	$\stackrel{\text{def}}{=}$	$\{1, 2, \ldots n\}$
vocalic	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{a, e, i, o, u\}\}$
low	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i = a\}$
high	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{i, u\}\}$
front	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{e, i\}\}$
stop	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{b, d, g, k, p, t\}\}$
fricative	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{h, s, z\}\}$
nasal	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{m, n\}\}$
lateral	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i = l\}$
rhotic	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i = r\}$
voiced	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{b, d, g, z\}\}$
voiceless	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{k, p, s, t, h\}\}$
labial	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{b, p, m\}\}$
coronal	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{d, s, t, z\}\}$
dorsal	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{g, k\}\}$
$\lhd$	$\stackrel{\text{def}}{=}$	$\{(i, i+1) \mid 1 \leq i < n\}$

Table 2.4: Creating a successor model with features for any word $w = b_1 b_2 \ldots b_n$.

with features).

$$\texttt{*NT} \stackrel{\text{def}}{=} \neg\exists x, y(x \lhd y \wedge \texttt{nasal}(x) \wedge \texttt{voiceless}(y)) \tag{2.11}$$

This looks similar to the definition of *NT under the successor model (Equation 2.7), but there is a critical difference. The predicates above in Equation 2.11 are *atomic* formulas and not user-defined predicates as they are in Equation 2.7.

This is an important ontological difference between these two models. In the successor model with features there is no primitive representational concept that corresponds to a sound segment like [t] as there is in the successor model without features. Conversely, in the successor model without features there is no primitive representational concept that corresponds to a phonological feature like *voiceless* as there is in the successor model

$$\mathcal{M}_{sans} = \Big\langle \{1,2,3,4\} \mid$$

$$\begin{aligned}
\texttt{vocalic} &= \{2\},\\
\texttt{low} &= \{2\},\\
\texttt{fricative} &= \{1,4\},\\
\texttt{nasal} &= \{3\},\\
\texttt{voiceless} &= \{1,4\},\\
\texttt{coronal} &= \{1,3,4\},\\
\texttt{lateral} &= \varnothing\\
&\ldots\\
\triangleleft &= \{(1,2),(2,3)\\
&\quad (3,4)\}\Big\rangle
\end{aligned}$$

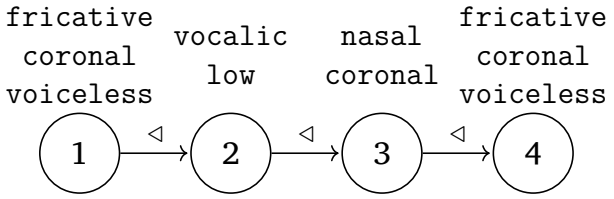


Figure 2.2: At left, the successor model with features of the word *sans*. Unary relations which equal the empty set are omitted for readability. At right, a graphical diagram of this model.

with features. Features are *derived* concepts in the the successor model without features, and segments are *derived* concepts in the the successor model with features.

In the successor model with features we can write user-defined predicates that define properties of domain elements that we can interpret to mean "being *t*".

$$\texttt{t}(x) \stackrel{\text{def}}{=} \texttt{stop}(x) \wedge \texttt{coronal}(x) \wedge \texttt{voiceless}(x) \tag{2.12}$$

Other sound segments would be defined similarly.

One way to put this difference is that in the successor model with features one can immediately determine whether a domain element is voiced or not, but in the successor model without features one cannot immediately determine this fact. Instead one can deduce it by checking the appropriate user-defined predicate. Likewise, in the successor model with features one cannot immediately determine whether a domain element is *t* or not. With the featural representations, such a fact must be deduced with a user-defined predicate like the one above.

Also, the fact that such user-defined predicates exist should not be taken for granted. They exist here because the only logical system discussed so far is FO. With FO logic, it is possible to define a predicate for any subset of the alphabet Σ for both successor models with and without features. If the logical system was restricted in some further way then some user-defined predicates may not be possible to define. For example, if the logical system only permitted conjunction and no other Boolean connective then it would not be possible to define a predicate for voiceless stops in the successor model without features. This interplay between representations and logical power with respect to expressivity is an important theme of this chapter. It will be discussed at length with respect to the successor relation, and we will return to it in the context of features when restricted logics are introduced in Chapter 5.

It is a consequence of FO logic that any constraint definable with one of the successor models discussed so far is definable in the other. This leads to the conclusion that there are no typological distinctions between a theory that holds that the right CDL for phonology is one based on First-Order logic over the successor model with features and a theory that holds the right CDL for phonology is one based on First-Order logic over the successor model without features. Both admit exactly the same class of constraints, with respect to some alphabet Σ.

However, while the two models do not make different typological predictions, they make different predictions in other ways. This is because in regard to phonological theory, the model signature is an ontological commitment to the psychological reality of the model vocabulary. Taken seriously, the successor model with features says that the mental representations of words carries only the information shown in Figure 2.2. Thus, taken seriously, the successor model with features says that the segments in the word *sans* are not perceived as such but are instead perceived in terms of their features. Clever psycholinguistic experiments could bring evidence to bear on which model more accurately resembles the actual mental representations of words.

2.6 Monadic Second-Order Logic

This section introduces Monadic Second-Order (MSO) logic. This logic is strictly *more expressive* than FO logic. We motivate the discussion of MSO

logic from a linguistic perspective by showing that FO with successor, both with and without features, is not sufficient to account for long-distance phonotactic constraints.

What are long-distance phonotactic constraints? Odden (1994) draws attention to an unbounded nasal assimilation in Kikongo whereby underlying /ku-kinis-il-a/ becomes [kukinisina] 'to make dance for.' From one perspective, this assimilation could be said to be driven by a phonotactic constraint that forbids laterals from occurring after nasals. Similar long-distance constraints have been posited for a variety of long-distance assimilation and dissimilation processes (Rose and Walker, 2004; Hansson, 2010).

We first show that the phonotactic constraint which bans laterals from occurring *anywhere* after nasals cannot be expressed in the FO with successor model. We refer to this constraint as *N..L. As we hope to make clear, the problem is that the notion of *precedence* is not FO-definable from successor. To illustrate this problem, consider that the logically possible word [kukinisila] is ill-formed in Kikongo. The nasal [n] has only one successor [i], but it *precedes* many segments including the second and third [i]s as well as the [s,l] and [a]. It is the fact that [n] precedes [l] which makes [kukinisila] ill-formed according to the phonotactic constraint *N..L.

Constraint *N..L is not FO definable with successor. To prove this we use an abstract characterization of the constraints definable with FO($\lhd$) due to Thomas (1982) and reviewed in Rogers and Pullum (2011). Thomas called the class of formal languages obeying this characterization **Locally Threshold Testable**.

Theorem 1 (Characterization of FO($\lhd$) definable constraints)**.** *A constraint is FO-definable with successor if and only if there are two natural numbers k and t such that for any two strings w and v, if w and v contain the same substrings x of length k* the same number of times counting only up to t*, then either both w and v violate the constraint or neither does.*

Essentially, this theorem says constraints that are FO-definable with successor cannot distinguish among strings that are composed of the same *number* and *type* of substrings of some length k, where substrings can be counted only up to some threshold t.

We can use this theorem to show that *N..L is not FO definable with successor by presenting two strings which *N..L distinguishes but which are not distinguishable according to the criteria in Theorem 1. This would

prove that *N..L is not LTT and thus not FO-definable with successor. Importantly, we have to present two such strings for any k and t. (These strings can depend on k and t.)

We use notation b^k to mean the string consisting of k consecutive bs. So $b^3 = bbb$. For any numbers k and t larger than 0, consider the words $w = o^k n o^k \ell o^k$ and $v = o^k \ell o^k n o^k$. Table 2.5 below shows the substrings up to length k, and their number of occurrences. Each word has the same substrings and the same number of them. Note the left and right word boundaries ($\rtimes$ and $\ltimes$ respectively) are customarily included as part of the strings.

count	$w = \rtimes o^k n o^k \ell o^k \ltimes$	Notes
1	$\rtimes o^{k-1}$	
3	o^k	
1	$o^i n o^j$	for each $0 \leq i, j \leq k-1$, $i + j = k - 1$
1	$o^i \ell o^j$	for each $0 \leq i, j \leq k-1$, $i + j = k - 1$
1	$o^{k-1} \ltimes$	

count	$v = \rtimes o^k \ell o^k n o^k \ltimes$	Notes
1	$\rtimes o^{k-1}$	
3	o^k	
1	$o^i n o^j$	for each $0 \leq i, j \leq k-1$, $i + j = k - 1$
1	$o^i \ell o^j$	for each $0 \leq i, j \leq k-1$, $i + j = k - 1$
1	$o^{k-1} \ltimes$	

Table 2.5: The k-long substrings with their number of occurrences in the strings $w = o^k n o^k \ell o^k$ and $v = o^k \ell o^k n o^k$ with word boundaries.

As can be seen from the above table, the two strings have exactly the same number of occurrences of each k-long substring. Consequently, for any threshold t, the counts of the k-long substrings will also be the same. It follows, from Theorem 1 that these two strings cannot be distinguished by any constraint which is FO-definable with successor.

More precisely, *any constraint which is FO-definable with successor is unable to distinguish in strings w and v whether n precedes ℓ or whether ℓ precedes n.* As such, no FO-definable constraint with successor can be violated by w but not by v and vice versa. It follows that *N..L is not FO

definable with successor for precisely the reason that it is this distinction that *N..L makes.

Having established that linguistically motivated long-distance phonotactic constraints are not FO-definable with successor, we turn to the question of how such constraints can be defined from the logical perspective offered here. Essentially, there are two approaches. One is to increase the power of the logic. The other is to change the signature—the representational primitives—of strings. This section examines the first option and the next section examines the second option. This interplay between logical power and representations and how it affects the expressivity of the linguistic system is a running theme of this book.

Monadic Second Order (MSO) logic is a logical language that is strictly more powerful than FO logic. Constraints that are MSO-definable with successor include every constraint which is FO-definable with successor because every sentence and formula in FO($\lhd$) is also a sentence and formula in MSO logic with successor and is interpreted in the same way. In addition to first order variables, MSO comes with **second order variables**. Generally, variables that are second order are allowed to vary over n-ary relations. The restriction to monadic second order variables means the variables in this logic can only vary over unary relations, which correspond to *sets* of domain elements. This contrasts with first order variables, which vary only over individual elements of the domain.

MSO logic is defined formally in Chapter 6, so here we introduce it informally with examples. In MSO logic, the MSO variables are expressed with capital letters such as X, Y, and Z to distinguish them from first order variables which use lowercase letters like $x, y,$ and z. Observe that $x \in X$ and $X(x)$ are synonyms. As with first order variables, second order variables are introduced into sentences and formulas with quantifiers.

Additional Symbols in MSO logic	
X, Y, Z	variables which range over sets of elements of the domain
$x \in X$	checks whether an element x belongs to a set of elements X
$X(x)$	checks whether an element x belongs to a set of elements X

Table 2.6: Together with the symbols of FO logic shown in Table 2.2, these symbols make up MSO logic.

With MSO logic over successor, denoted MSO($\triangleleft$), it is now possible to define the precedence relation as shown below.

$$\texttt{closed}\,(X) \stackrel{\text{def}}{=} (\forall x, y)\big[(x \in X \wedge x \triangleleft y) \rightarrow y \in X\big] \tag{2.13}$$

$$x < y \stackrel{\text{def}}{=} (\forall X)\big[(x \in X \wedge \texttt{closed}\,(X)) \rightarrow y \in X\big] \tag{2.14}$$

Intuitively, a set of elements X in the domain of a model of some word w satisfies $\texttt{closed}\,(X)$ only if every successor of every element in X is also in X. In short, $\texttt{closed}\,(X)$ is true only for sets of elements X which are transitively closed under successor. Then x precedes y only if for every closed set of elements X which x belongs to, y also belongs to X.

Figure 2.3 below illustrates these ideas. The successor model for the string $onool o$ is shown. Six rectangular regions are shown, which identify the six nonempty sets of domain elements which are closed under successor and thus satisfy $\texttt{closed}\,(X)$.

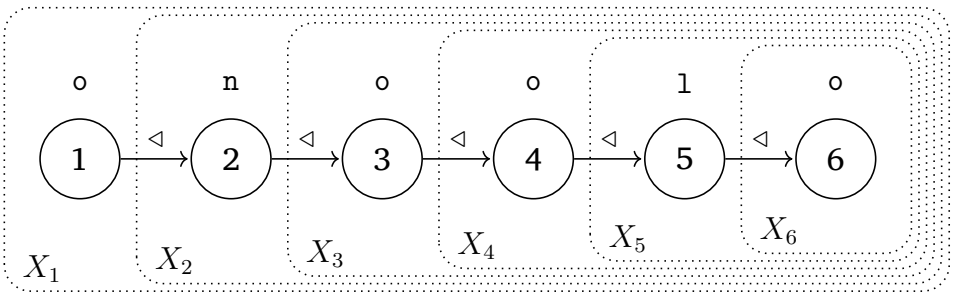


Figure 2.3: The successor model for the word $onool o$. The dotted rectangular regions indicate the sets of domain elements (X_i) which are closed under successor.

We can conclude that n (at position 2) precedes ℓ (at position 5) because every closed set which element 2 belongs to (X_1 and X_2) also includes the element 5. Similarly, we can conclude that ℓ does not precede n because it is not the case that all closed sets which contain element 5 also include element 2. Set X_4 for instance contains element 5 but not element 2.

Once the binary relation for precedence ($<$) has been defined, it is now

straightforward to define the constraint *N..L with features.

$$\texttt{*N..L} \stackrel{\text{def}}{=} \neg(\exists x, y)[x < y \wedge \texttt{nasal}(x) \wedge \texttt{lateral}(y)] \tag{2.15}$$

The sentence above may look like a sentence of FO logic since no second order variables are present. However, it is important to remember that the precedence relation ($<$) is a user-defined predicate, and as such it is just an abbreviation for a longer expression, which is defined using the second order variables of MSO logic. Therefore Equation 2.15 is not an expression of FO($\triangleleft$).

In many treatments of logic, whether a predicate is atomic or derived is not something that can be determined from inspecting a sentence or formula since the notation does not distinguish them. In this book, we are using very light gray highlighting to distinguish derived predicates from atomic formulas. Readers should be aware, however, that usually one must be being acutely aware of the model signature to know whether a predicate is atomic or derived.

At this point, we have established that the linguistically motivated long-distance phonotactic constraint *N..L is not definable with FO logic with successor but is definable with MSO logic with successor. We thus ask: What other kinds of constraints are MSO-definable with successor?

Another constraint that is not FO-definable with successor but is MSO-definable constraint with successor is a constraint that requires words to have an even number of nasals. Words like *man* and *phenomenon* obey this constraint since they have two and four nasals, respectively, but words like *trim, nanotechnology* and *nonintervention* do not since they have one, three and five nasals, respectively.

To see that this constraint is not FO-definable with successor, we use Theorem 1 as before. For any nonzero numbers k and t, consider the words $w = a^k(na^k)^{2t}$ and $v = a^k(na^k)^{2t}na^k$. Observe that w obeys the constraint since it contains $2t$ nasals and $2t$ is an even number. On the other hand, v contains $2t + 1$ nasals and therefore violates the constraint. However, as Table 2.7 shows, these words have the same substrings of length k, and the same numbers of each substring, counting only up to the threshold t.

However, this constraint is expressible with MSO logic with successor. We make use of some additional predicates, including general precedence ($<$) defined in Equation 2.14. The predicate $\texttt{firstN}$ is true of x only if x is the first nasal occurring in the word (Equation 2.16). The predi-

$w = \rtimes a^k(na^k)^{2t}a^k \ltimes$			
k-long substring	raw count	count up to threshold t	notes
$\rtimes a^{k-1}$	1	1	
a^k	$2t+2$	t	
$a^i n a^j$	$2t+2$	t	for each $0 \leq i,j \leq k-1$, $i+j=k-1$
$a^{k-1} \ltimes$	1	1	

$v = \rtimes a^k(na^k)^{2t}na^k \ltimes$			
k-long substring	raw count	count up to threshold t	notes
$\rtimes a^{k-1}$	1	1	
a^k	$2t+3$	t	
$a^i n a^j$	$2t+3$	t	for each $0 \leq i,j \leq k-1$, $i+j=k-1$
$a^{k-1} \ltimes$	1	1	

Table 2.7: The k-long substrings and the numbers of their counts in $w = a^k(na^k)^{2t}a^k$ and $v = a^k(na^k)^{2t}na^k$ with word boundaries.

cate $\texttt{lastN}$ is true of x only if x is the last nasal occurring in the word (Equation 2.17). Also, two variables x and y stand in the nasal-successor relation (denoted $\lhd_{\texttt{N}}$) only if x and y are nasals and y is the first nasal to occur after x (Equation 2.18). Essentially, $\lhd_{\texttt{N}}$ is the successor relation relativized to nasals (Lambert, 2023).

$$\texttt{firstN}(x) \stackrel{\text{def}}{=} \texttt{nasal}(x) \wedge \neg(\exists y)[\texttt{nasal}(y) \wedge y < x] \tag{2.16}$$

$$\texttt{lastN}(x) \stackrel{\text{def}}{=} \texttt{nasal}(x) \wedge \neg(\exists y)[\texttt{nasal}(y) \wedge x < y] \tag{2.17}$$

$$x \lhd_{\texttt{N}} y \stackrel{\text{def}}{=} \texttt{nasal}(x) \wedge \texttt{nasal}(y) \wedge x < y \wedge \neg(\exists z)[\texttt{nasal}(z) \wedge x < z < y] \tag{2.18}$$

Note we use the shorthand $x < y < z$ for $x < y \wedge y < z$.

With these predicates in place, we write EVEN-N as in Equation 2.19.

$$\begin{aligned}\textsc{even-n} \stackrel{\text{def}}{=} (\exists X)\Big[\quad & (\forall x)[\ \texttt{firstN}\ (x) \rightarrow X(x)] \\ & \wedge\ (\forall x)[\ \texttt{lastN}\ (x) \rightarrow \neg X(x)]\ \Big] \\ & \wedge\ (\forall x, y)\big[x \lhd_{\texttt{N}} y \wedge \big(X(x) \leftrightarrow \neg X(y)\big)\big] \end{aligned} \tag{2.19}$$

In English, this says that a model of word w satisfies EVEN-N provided there is a set of domain elements X that includes the first nasal (if one occurs), does not include the last nasal (if one occurs) and for all pairs of successive nasals (if they occur), exactly one belongs to X. Consequently, words containing zero nasals satisfy EVEN-N because the empty set of domain elements vacuously satisfies these three conditions. Words containing exactly one nasal do not satisfy EVEN-N because the first nasal and the last nasal are the same element x and they cannot both belong and not belong to X. However, words with exactly two nasals do satisfy EVEN-N because the first nasal belongs to X (satisfying the first condition), the last nasal does not (satisfying the second condition), and these two nasals are successive nasals and so are subject to the third condition, which they satisfy because exactly one of them (the first nasal) belongs to X. A little inductive reasoning along these lines lets one conclude that only words with an even number of nasals will satisfy EVEN-N as intended.

It is natural to wonder whether there is an abstract characterization of constraints that are MSO-definable with successor in the same way that Thomas (1982) provided an abstract characterization of constraints that are FO-definable with successor. In fact there is. Büchi (1960) showed that these constraints are exactly the ones describable with finite-state automata.

Theorem 2 (Characterization of MSO-definable constraints with successor)**.** *A constraint is MSO-definable with successor if and only if there is a finite-state acceptor which recognizes the words obeying the constraint.*

From the perspective of formal language theory, they are exactly the regular languages. Informally, these are formal languages for which the membership problem can be solved with a constant, finite amount of memory regardless of the size of the input.

In this section, we showed that FO-definable constraints with successor are not sufficiently powerful to express long-distance phonotactic constraints. One approach is to then increase the power of the logic. One

logical system extends FO by adding quantification over monadic second order variables. This logic—MSO logic with successor—is able to express long-distance phonotactic constraints. However, MSO logic with successor is also sufficiently expressive as a CDL to express constraints like EVEN-N.

Here is another way of putting it. In successor structures, the information that in the word $o\ell oono$ the ℓ precedes the n is not immediately available from the representation. That information can be *deduced* but the deduction requires some computational effort. From the logical perspective taken here, this deduction requires MSO power and not FO power. Furthermore, once MSO power is admitted then it becomes possible to similarly deduce whether or not there are even numbers of elements with certain properties.

Another approach to developing a CDL which can express long-distance phonotactic constraints is to change the representation of strings; that is, to change the model signature. This is precisely the topic of the next section.

2.7 The Precedence Word Model

So far, the logics we have considered have been defined with respect to the successor model of words. These representations include the successor relation in their signature. However, as we have seen with phonological features vis a vis atomic letters, there are different models of strings. In this section, we consider the *precedence* model of strings. Simply, this model contains the **precedence relation** instead of the successor relation in its signature.

A domain element x stands in the precedence relation to y if y is an event that occurs sometime later than x. In this book, we use the symbol $<$ to indicate the precedence relation. For the word *sans*, it holds that $1, 4 \in R_s$ and $(1, 4)$ belongs to the precedence relation since position 4 occurs later than position 1. We can write this fact in several ways, including $1 < 4$, $< (1, 4)$, and $(1, 4) \in <$. The signature for the precedence model with letters is thus $\{(\mathtt{b})_{b\in\Sigma}, <\}$ and $<$-structures would have the form $\langle D \mid (\mathtt{b})_{b\in\Sigma}, < \rangle$.

As with the successor structures, there is a general method for constructing precedence structures for strings. Given a string w of length n, the domain and unary relations of a precedence structure for w are constructed in the same way as was the case for the successor structure. Regarding the precedence relation itself, for each i and j between 1 and n inclusive, (i, j)

belongs to the precedence relation so long as $i < j$. This is summarized in Table 2.8. This construction guarantees the model's soundness: each string

D	$\stackrel{\text{def}}{=}$	$\{1, 2, \ldots n\}$
a	$\stackrel{\text{def}}{=}$	$\{i \in D \mid b_i = b\}$ for each unary relation b
$<$	$\stackrel{\text{def}}{=}$	$\{(i, j) \subseteq D \times D \mid i < j\}$

Table 2.8: Creating a precedence model for any word $w = b_1 b_2 \ldots b_n$.

has a model and distinct strings will have distinct models.

Figure 2.4 shows the $<$-structure for the word *sans* on the left and a graphical diagram of it on the right.

$$
\begin{aligned}
&\mathcal{M}_{sans} \\
&= \langle D \mid \mathtt{s}, \mathtt{a}, \mathtt{n}, \mathtt{b}, \mathtt{c}, \ldots, \mathtt{z}, < \rangle \\
&= \Big\langle \{1,2,3,4\} \mid \{1,4\}, \{2\}, \{3\}, \\
&\quad \varnothing, \varnothing, \ldots \varnothing, \\
&\quad \{(1,2),(1,3),(1,4), \\
&\quad\ (2,3),(2,4),(3,4)\} \Big\rangle
\end{aligned}
$$

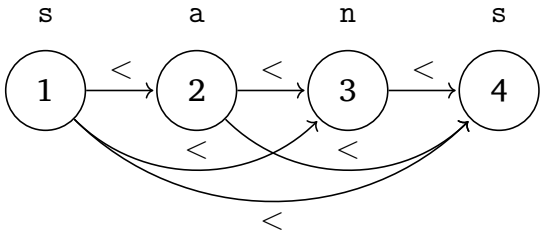


Figure 2.4: At left, the precedence model of the word *sans*. At right, a graphical diagram of this model.

The difference between the $<$-structures and the $\lhd$-structures is how the order of segments in the word are represented. In the precedence model, the fact that the n is preceded by s in the word $sans$ is immediately available because the element corresponding to n (position 3) is in the precedence relation with the element corresponding to the first s (position 1). Under the successor model, this information was not immediately available as it was not part of the representation. However, under the precedence model it is.

Taken seriously from a psychological perspective, the precedence model can be taken to mean that as words are perceived, information about the precedence relations is being stored in memory as part of the lexical representation of the word.

Also, in the same way that we considered the successor model both with and without features, we can also consider a precedence model with and without features. The precedence model introduced above was without features, but it is a simple matter to replace the unary relations in that model with the ones in Table 2.4.

It is straightforward to now write the constraint *N..L in the CDL which we call "FO with precedence with features" denoted FO(feat,$<$).

$$\texttt{*N..L} \stackrel{\text{def}}{=} \neg\exists x, y(x < y \wedge \texttt{nasal}(x) \wedge \texttt{lateral}(y)) \tag{2.20}$$

Equation 2.20 looks identical to Equation 2.15. However, there is a critical difference. In Equation 2.20, the precedence relation is an atomic formula but in Equation 2.15 it is a user-defined predicate in MSO logic.

It is natural to ask of course whether a constraint like *NT is expressible in this CDL. The answer is Yes because successor is FO-definable from precedence. Equation 2.21 shows how. Essentially, x is succeeded by y only if x precedes y and there is no element z such that $z < y$ and $x < z$.

$$x \lhd y \stackrel{\text{def}}{=} x < y \wedge \neg(\exists z)[x < z < y] \tag{2.21}$$

It is a striking fact that successor is FO-definable from precedence but precedence is MSO-definable from successor. This is a considerable asymmetry between the successor and precedence models of strings.

There are two important consequences. The first is the CDL FO($<$) properly subsumes the CDL FO($\lhd$). Not only is every constraint expressible with the FO($\lhd$) also expressible with the FO($<$), but there are constraints like *N..L above that are expressible with the FO($<$) but not with the FO($\lhd$).

Another important consequence is that the CDL MSO($<$) is equivalent in expressive power to the CDL MSO($\lhd$) discussed in the previous section. This is because with MSO logic, precedence can be defined from successor as shown previously in Equation 2.14 on page 47 and because successor can be defined from precedence as shown above in Equation 2.21. So at the level of MSO, these two models make no distinctions among the kinds of constraints that can be expressed. Furthermore it has been known since Büchi (1960), that these constraints correspond to exactly the regular stringsets.

There is also an abstract characterization of FO($<$) due to McNaughton and Papert (1971).

Theorem 3 (Characterization of FO-definable constraints with precedence)**.** *A constraint is FO-definable with precedence if and only if there is a positive integer n such that for all strings x, y, z if xy^nz obeys the constraint then for all $k > n$, xy^kz obeys the constraint too.*

This characterization says that FO-definable constraints with precedence can only distinguish iterations within strings up to some finite n. In other words, two strings xy^iz and xy^jz, with both $i, j > n$ but $i \neq j$, cannot be distinguished by any FO-definable constraint with precedence. As McNaughton and Papert (1971) amply document, there are other independently-motivated characterizations of this class as well.

The above characterization can be used to show that EVEN-N is not FO-definable with precedence. Again, the strategy is to consider any n and then to find strings w, v such that (1) EVEN-N distinguishes w and v in the sense that one violates EVEN-N and the other does not while (2) ensuring that the forms of w, v conform to $w = xy^iz$ and $v = xy^jz$ for some x, y, z and numbers $i, j > n$. If the constraint were FO-definable with precedence such strings could not exist by Theorem 3. In this case, one solution is to set $x = z = \lambda$ (the empty string), $y = ma$, $i = 2n$ and $j = 2n + 1$. Then $w = (ma)^{2n}$ and $v = (ma)^{2n+1}$. Clearly, w has an even number of nasals since it has $2n$ [m]s but v has an odd number since it has $2n+1$ [m]s. Thus EVEN-N distinguishes these strings and thus by Theorem 3 it cannot be FO-definable with precedence.

In this section, we considered a model of words where order is represented with the precedence relation instead of the successor relation. It was shown that long-distance constraints can be readily expressed with FO($<$). Furthermore, local phonotactic constraints like *NT can also be expressed because successor is FO-definable from precedence. However, the converse is not true. This asymmetry means that FO($<$) is strictly more expressive than FO($\lhd$). Despite this richer expressivity, it was also shown that EVEN-N cannot be expressed in FO($<$). Finally, it was noted that MSO($<$) is equally expressive as MSO($\lhd$). Once there is MSO power, successor and precedence are each definable from the other. Which of these constraints can be expressed by which CDLs is summarized in Figure 2.5.

More generally, this section established the following. Although one way to increase the expressivity of a CDL is to increase the power of the logic, another way is to change the representations underlying in the model signatures. This speaks directly to the interplay between representations

	$\lhd$	$<$
MSO	*N..L, EVEN-N	EVEN-N
FO	*NT	*NT, *N..L

Figure 2.5: Classifying the constraints *NT, *N..L, and EVEN-N.

and computational power, one of the themes of this chapter.

We conclude that the only CDL discussed so far that can express both local and long-distance phonotactic constraints (like *NT and *N..L) but that fails to express constraints like EVEN-N is FO($<$).

2.8 Discussion

This chapter has been about many things. On the one hand, it introduced model theory and logical languages as a toolkit for providing what De Lacy termed a Constraint Definition Language.

It then proceeded to show how different words can be represented in different ways based on the primitive relations one chooses to include in the model theoretic signature. Four examples were introduced: representations with letters and successor, representations with features and successor, representations with letters and precedence, and representations with features and precedence.

Additionally, two logics were introduced, First Order logic and Monadic Second Order logic. We explained how the choice of representation and choice of logic gives rise to a logical language which can express constraints over those representations.

Finally, we explored some of the consequences of these choices. The most important ones we stressed are the following.

1. If order is represented with successor and not precedence, then MSO logic is needed to be able to express long distance phonological constraints.

2. Logical languages defined with MSO logic over successor structures can also express constraints that forbid (or require) words to have n many structures modulo m ($0 \leq n < m$) (for example, "Words must contain evenly many nasals").

3. If order is represented with precedence and not successor, then FO logic is sufficient to express long distance phonological constraints, in addition to local phonological constraints.

4. The above results follow directly from a fact of mathematical logic: precedence needs MSO logic to be defined from successor but successor only needs FO logic to be defined from precedence.

We also return to the point that the symbolic and featural representations in Tables 2.1 and 2.4 can be defined in terms of the other using FO logic because a symbol can be defined in terms of the conjunction of the features that make up that symbol and similarly a feature-value can be defined in terms of a disjunction of symbols which have that feature-value. It follows that for any constraint C expressible in FO(feat,$\triangleleft$), there is a constraint D in FO($\triangleleft$) such that exactly the same strings violate both constraints, and vice versa.

Does this mean that there are no differences between symbolic and featural representations? No. While it *does* mean that one cannot distinguish the constraints one can express if FO logic is used along with features or symbols, it does not say anything about a logic that is *weaker* than FO logic. A weaker logic may very well distinguish the expressible constraints these distinct representational primitives can express. It also doesn't say anything about the psychological implications or learning. Other kinds of evidence from psycholinguistics (Durvasula and Nelson, 2018) or learning (Wilson and Gallagher, 2018) may be brought to bear on the best choice of representational primitive.

Here are some of the reasons exploring different representational schemes and different logics—that is exploring the space of possible Constraint Definition Languages—is a worthwhile goal. First, the choice of representation and the choice of logic yields a rigorous, logical language whose formulas are readable by both humans and machines which can be used to always correctly answer the question whether a given structure satisfies a given formula or not. Second, this logical language can be studied explicitly to reveal what kinds of constraints it can and cannot express, the facts of which should then be compared with the typology of phonological constraints. This lets us draw conclusions like "If there are long-distance constraints in phonology, then FO with successor is insufficient as a *theory* of phonological constraints."

In addition to evaluating a logical language in terms of its typological predictions, we can also examine its psychological predictions as well as what it would mean for learnability and acquisition. The representational primitives of the logical language can be understood as a hypothesis of the psychologically real representational primitives. We can also ask whether there are algorithms that can learn the formulas of a logical language and which of these algorithms exhibit behavior observed when humans learn language.

We hope that this chapter helps persuade readers that exploring different representational schemes and different logics—that is exploring the space of possible Constraint Definition Languages—is a worthwhile goal.

Chapter 3

Transformations, Logically

JEFFREY HEINZ

This chapter explains how transformations from one representation to another can be described with the same logical tools introduced in the last chapter. Transformations are a central component of phonological theory, which posits a mapping exists between the long-term memory representations of the pronunciation of morphemes (the underlying forms) to the more more directly observable, surface representations (the surface forms) (Hyman, 1975; Kenstowicz and Kisseberth, 1979; Krämer, 2012; Odden, 2014). The mathematical and computational basis for this work originates with Courcelle (1994), a thorough survey of which is provided by Courcelle and Engelfriet (2012).

This chapter aims to introduce these ideas in an accessible way to linguists with a basic knowledge of phonology. However, the techniques have application beyond the theory of phonology to any other subfield of linguistics, notably morphology and syntax, in part because these methods apply equally well to trees and graphs, not just strings. Also this chapter is merely an introduction to these methods. As such, it introduces them in the context of string-to-string transformations; that is, **functions** from strings to strings. As a matter of fact, these methods have been generalized by computer scientists to describe **weighted relations** between strings (Droste and Gastin, 2009). These generalizations permit one to describe and characterize optionality and exceptionality, in addition to gradient and probabilistic generalizations. Weighted logics are treated separately in Chapter 4. Here however, and throughout most of this book, unweighted

logic is used primarily because weighted logics can obfuscate the central ideas, which are easier to first understand without them.

The application of these methods for phonological description and theory is what primarily distinguishes this work from One-Level Declarative Phonology developed by Bird, Coleman, and Scobbie some thirty years ago. That research, like the research in this book, emphasizes a declarative approach to phonological description and theory. The key difference is that thirty years ago transformations were studied within "one level." In other words, transformations were understood as *constraints* on *unspecified* underlying representations. As such, those 'transformations' could only *add* (further specify) information to those representations. In contrast, in this chapter we will see how logic can be used to literally add, subtract, change, or more generally *transform* one representation into another. For this reason, one could say that the Computational Generative Phonology approach in this book is essentially a form of *two level* Declarative Phonology (Dolatian, 2020a).

3.1 String-to-string Transformations

A logical description of a string-to-string function uses logic to explain how an input string is mapped to an output string. As with the constraints in the previous chapter, the logic does not operate over the strings themselves, but over the model-theoretic representation of those strings. Therefore, a logical description of a string-to-string function uses logic to convert an input *structure* of a string into an output *structure* (possibly representing a different string). Recall that the structure of a string depends on the model *signature*, and that the signature lists the relations over the domain of the model which must be specified in order to uniquely identify some string. Therefore, the logical description needs to specify the *output structure* in terms of a logical language given by the signature of the *input structure*.

Logical descriptions of string-to-string functions must accomplish two things. First, they must specify the domain of the function – which strings does the function apply to? Second, for each of (the structures representing) these input strings, it must specify (structures of) the output strings these input strings map to. This means specifying both the domain of the output

structure and the relations over it.[1] These goals are accomplished with a collection of logical formulas. For a logical description of a function f, these formulas together answer questions like the following. Does f apply to this string? For a given string w, what is $f(w)$?

As mentioned, there are several ingredients making up a logical transformation, each with their own names. The domain of the function is specified by the **domain formula**. The domain of the output structure is specified by two ingredients: the **copy set** and the **licensing formulas**. The relations over the output structure are specified by **relational formulas**. There is one relational formula for each of the relations in the model signature of the output. All of these formulas are evaluated with respect to the **input structure** in a way that will be made clear below.

In the remainder of this chapter, these ingredients will be explained with familiar phonological processes. We begin with **word-final obstruent devoicing**, which changes a single feature. We next consider **word-final vowel deletion** where the output can be smaller than the input. This is followed by **word-final vowel epenthesis** where the output can be larger than the input. We then show how logical transductions can be used to describe total reduplication. With those basics in place, we consider the power of MSO-definable transformations by illustrating two logically possible string-to-string transformations that are not attested, as far as I know, as phonological processes.

3.2 Word-final obstruent devoicing

For concreteness, let us provide a logical description of the phonological process of word-final obstruent devoicing. This process maps strings with word-final voiced obstruents to voiceless ones. For example, this process maps the string *hauz* to *haus* and the string *bad* to *bat*. Words without word-final voiced obstruents surface faithfully so this process can be said

[1] Note there are two distinct meanings of the word 'domain' in use here. The first has to do with the domain of a *function* and the second with the domain of a *structure*. A function's domain is the set of elements over which the function is defined. For instance for $F : A \to B$, the domain is the set A. In contrast, the domain of a structure is the elements in the 'universe' the structure is describing. In finite model theory, which is used in this book, the domain of a structure is a finite set D of natural numbers 1,...n, representing the finitely many elements in the universe.

to map the string *haus* to *haus*.

We choose to model this process with the feature-based successor model FO (feat, $\triangleleft$) described in 2.5 (see Table 2.4). More precisely, strings in both the input and the output will be represented with (feat, $\triangleleft$)-structures. Note this is a choice and one can choose to model the input, the output, or both, with other word models.

It follows we want to provide a logical transformation which, for example, maps the (feat, $\triangleleft$)-structure of *hauz* to the (feat, $\triangleleft$)-structure of *haus*, as shown in Figure 3.1. We introduce the logical formulas one at a time and

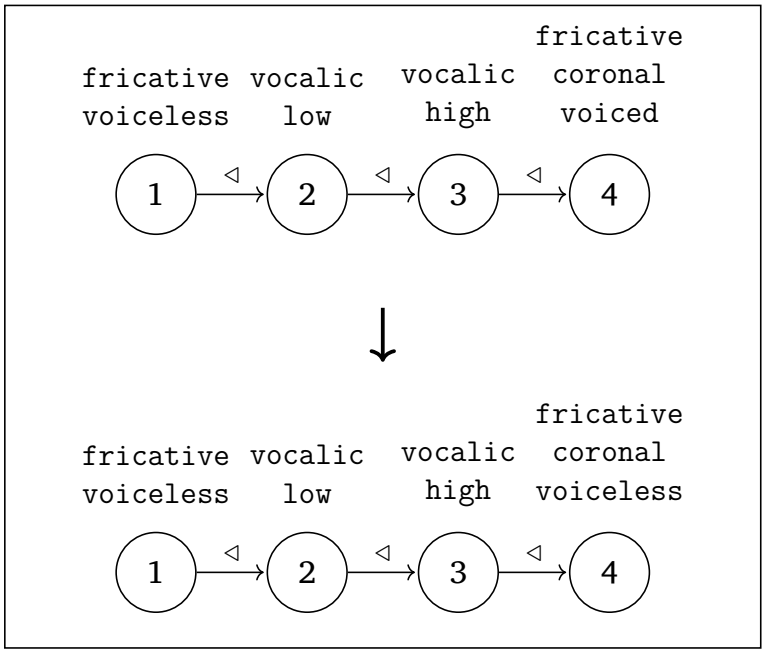


Figure 3.1: A graphical diagram of the feature-based successor model of *hauz* being mapped to the feature-based successor model of *haus*.

then summarize them at the end of the example.

The domain of the function f is specified with the **domain formula** $\phi_{\texttt{domain}}$. This is a logical formula with no free variables. For all strings w, $f(w)$ is defined if and only if the structure of w satisfies the formula ($\mathcal{M}_w \models \phi_{\texttt{domain}}$). For word-final obstruent devoicing, we want this function to apply to every string. Hence we set $\phi_{\texttt{domain}} \stackrel{\text{def}}{=} \texttt{true}$.

How is the domain of the output structure determined? Logical trans-

ductions fix the domain of the output as a *copy* of the input domain. For example, as shown in Figure 3.1, the domain of $\mathcal{M}_{hauz}$ is {1,2,3,4}. Therefore, the domain of the output structure of $f(hauz)$ is also the set {1,2,3,4}.

One consequence of constituting the domain of the output structure this way is that it appears that functions cannot alter the size of the input upon which they are acting. However, it is precisely the copy set C and the licensing formula $\phi_{\texttt{license}}$, discussed later in sections 3.4 and 3.3, respectively, which ultimately determine the precise size of the output structure. To give a basic preview, the copy set allows transformations to relate *larger* outputs to *smaller* inputs and the licensing formula allows transformations to relate *smaller* outputs to *larger* inputs. Working together, these ingredients let one relate inputs to outputs of different sizes. For now, since the word-final voiced obstruent devoicing does preserve the size of each input in the output, we postpone the particulars of how exactly copy-sets and the licensing formulas work until sections 3.4 and 3.3.

For obstruent devoicing, setting the copy set and licensing formula to $C = \{1\}$ and $\phi_{\texttt{license}} \stackrel{\text{def}}{=} \texttt{true}$ suffices to ensure that, given the input structure $\mathcal{M}_{hauz}$, the domain of the output structure is {1,2,3,4}.

Finally, we must determine the relations which hold over the domain elements of the output structure. For each relation R of arity n in the signature of output structure, we must specify a formula $\phi_{\texttt{R}}$ with n free variables $\phi_{\texttt{R}}(x_1, \ldots x_n)$. For word-final obstruent devoicing, the signature of the output structures has one binary relation (the successor relation $\lhd$) and several unary relations (the phonological features). Therefore, to specify this phonological process, we need to specify one logical formula with two free variables for the successor relation and several logical formulas with one free variable for the phonological features.

How are these logical formulas for the relations interpreted? For any string-to-string function f, input structure $\mathcal{M}_w$, and relation R of arity n in the output signature, the elements $x_1, \ldots x_n$ in the domain of the output structure stand in relation R if and only if $\mathcal{M}_w \models \phi_{\texttt{R}}(x_1, \ldots x_n)$. In other words, the formula $\phi_{\texttt{R}}(x_1, \ldots x_n)$ is evaluated with respect to the input structure, and the logical language to which $\phi_{\texttt{R}}(x_1, \ldots x_n)$ belongs is based on the input signature.

For example, the output signature contains the successor relation, which is a binary relation. So we must define the formula $\phi_{\lhd}(x, y)$. Since word-final obstruent devoicing does not affect the successor relations, we define

this function as follows.

$$\underbrace{\phi_{\triangleleft}(x,y)}_{\text{Do x and y in the output model stand in the successor relation?}} \stackrel{\text{def}}{=} \underbrace{x \triangleleft y}_{\text{Evaluate with respect to the input model.}}$$

This means the following: elements x and y *in the output structure* stand in the successor relation if and only if corresponding elements x and y satisfy the successor relation *in the input structure*. Since $1 \triangleleft 2$ in the input structure, it follows that elements 1 and 2 likewise stand in the successor relation in the output structure. Similarly, since elements 1 and 3 *do not stand* in the successor relation in the input structure, it follows that they *do not* stand in the successor relation in the output structure. Consequently, the formula above guarantees (in fact literally says) that the successor relation in the output will be the same as the successor relation in the input.

As another example, consider the unary relation `vocalic`. As this is a unary relation, we must define a formula with one free variable $\phi_{\texttt{vocalic}}(x)$. Let us define it as follows.

$$\underbrace{\phi_{\texttt{vocalic}}(x)}_{\text{Does x have the feature vocalic in the output structure?}} \stackrel{\text{def}}{=} \underbrace{\texttt{vocalic(x)}}_{\text{Evaluate with respect to the input structure.}}$$

It follows from this definition that domain element x in the output model is vocalic if and only if the corresponding domain element x in the input is vocalic. This formula captures the fact that word final obstruent devoicing does not affect the vocalic nature of any elements within a string.

As we know, the only features affected by word-final devoicing are *voicing* features, which are the relations `voiced` and `voiceless` in our model signatures. All other unary relations in the signature of the output structure will be defined similarly to $\phi_{\texttt{vocalic}}(x)$ (as shown in Table 3.1 on page 66). However, the voicing features are affected by this process, so how do we specify which domain elements are voiced or voiceless? The voiced elements will be the ones that were voiced in the input and are not word-final obstruents. We can formalize this as follows. It will be useful to

write some user-defined predicates.

$$\texttt{wordfinal}\,(x) \stackrel{\text{def}}{=} \neg\exists y\ (x \lhd y) \quad (3.1)$$

$$\texttt{obstruent}\,(x) \stackrel{\text{def}}{=} \texttt{stop}(x) \vee \texttt{fricative}(x) \quad (3.2)$$

$$\texttt{devoicingcontext}\,(x) \stackrel{\text{def}}{=} \texttt{wordfinal}\,(x) \wedge \texttt{obstruent}\,(x) \quad (3.3)$$

We thus define $\phi_{\texttt{voiced}}(x)$ as follows.

$$\underbrace{\phi_{\texttt{voiced}}(x)}_{\substack{\text{Does x have the feature } \texttt{voiced} \\ \text{in the output structure?}}} \stackrel{\text{def}}{=} \underbrace{\texttt{voiced(x)} \wedge \neg\,\texttt{devoicingcontext}\,(\texttt{x})}_{\substack{\text{Evaluate with respect to the in-} \\ \text{put structure.}}}$$

Similarly, the domain elements in the output which are voiceless are those that are voiceless in the input or those that are word-final obstruents.

$$\underbrace{\phi_{\texttt{voiceless}}(x)}_{\substack{\text{Does x have the feature } \texttt{voiceless} \text{ in the output} \\ \text{structure?}}} \stackrel{\text{def}}{=} \underbrace{\texttt{voiceless(x)} \vee\ \texttt{devoicingcontext}\,(\texttt{x})}_{\substack{\text{Evaluate with respect to the in-} \\ \text{put structure.}}}$$

As mentioned, since this process does not affect other phonological features in the string, each of those unary relations R in the signature of the output structure can be defined as follows: $\phi_{\texttt{R}}(x) \stackrel{\text{def}}{=} \texttt{R}(x)$. In other words, $\phi_{\texttt{vocalic}}(x) \stackrel{\text{def}}{=} \texttt{vocalic}(x)$ and $\phi_{\texttt{coronal}}(x) \stackrel{\text{def}}{=} \texttt{coronal}(x)$ and so on. For completeness, we show the complete logical description of word-final devoicing in Table 3.1.

3.3 Word-final vowel deletion

Let us consider another example, word-final vowel deletion, which will illustrate the role played by the licensing formula. Word-final vowel deletion has been argued to be a process in Yowlumne (also known as Yawelmani Yokuts) (McCarthy, 2008a). The process in Yowlumne is subject to additional conditions, which are set aside here. Word-final vowel deletion essentially maps strings like *paka* to *pak* and *pilot* to *pilot*.

$\phi_{\texttt{domain}}$	$\stackrel{\text{def}}{=}$	$\texttt{true}$
C	$\stackrel{\text{def}}{=}$	$\{1\}$
$\phi_{\texttt{license}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{true}$
$\phi_{\triangleleft}(x,y)$	$\stackrel{\text{def}}{=}$	$x \triangleleft y$
$\phi_{\texttt{vocalic}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{vocalic}(x)$
$\phi_{\texttt{low}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{low}(x)$
$\phi_{\texttt{high}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{high}(x)$
$\phi_{\texttt{front}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{front}(x)$
$\phi_{\texttt{stop}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{stop}(x)$
$\phi_{\texttt{fricative}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{fricative}(x)$
$\phi_{\texttt{nasal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{nasal}(x)$
$\phi_{\texttt{lateral}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{lateral}(x)$
$\phi_{\texttt{rhotic}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{rhotic}(x)$
$\phi_{\texttt{labial}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{labial}(x)$
$\phi_{\texttt{coronal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{coronal}(x)$
$\phi_{\texttt{dorsal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{dorsal}(x)$
$\phi_{\texttt{voiced}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{voiced}(x) \wedge \neg\, \texttt{devoicingcontext}\,(x)$
$\phi_{\texttt{voiceless}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{voiceless}(x) \vee \texttt{devoicingcontext}\,(x)$

Table 3.1: The complete logical specification for word-final obstruent devoicing when the input and output string models are both the feature-based successor model.

As before, the domain of this function is all strings and so $\phi_{\texttt{domain}} \stackrel{\text{def}}{=} \texttt{true}$. Also as before, the domain of the output structure is a copy of the domain elements of the input structure. However, these domain elements of the output structure do not automatically exist in the output structure; they must be *licensed* by a formula with one free variable called the licensing formula $\phi_{\texttt{license}}(x)$. In other words, the domain elements of the output

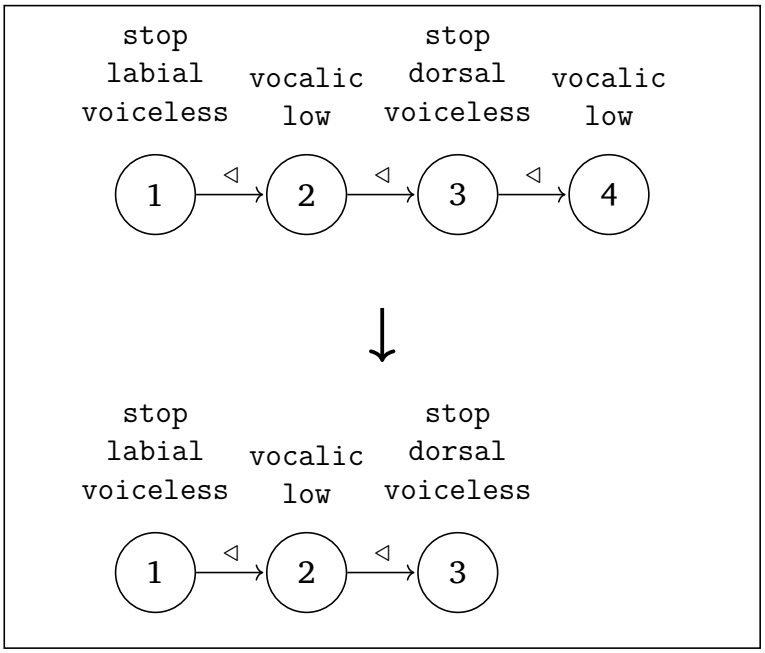


Figure 3.2: A graphical diagram of the feature-based successor model of *paka* being mapped to the feature-based successor model of *pak*.

structure are really the *licensed* copies of the domain elements of the input structure. Since word-final vowels delete in this process, all domain elements which do not correspond to word-final vowels are licensed.

$$\underbrace{\phi_{\texttt{license}}(x)}_{\text{Does x belong to the domain of the output model?}} \stackrel{\text{def}}{=} \underbrace{\neg(\ \texttt{wordfinal}\ (x) \wedge \texttt{vocalic}(x))}_{\text{Evaluate with respect to the input structure.}}$$

Also, this process does not affect any phonological features, so each of the unary relations R in the signature of the output structure can be defined as follows: $\phi_{\texttt{R}}(x) \stackrel{\text{def}}{=} \texttt{R}(x)$. In other words, $\phi_{\texttt{vocalic}}(x) \stackrel{\text{def}}{=} \texttt{vocalic}(x)$ and $\phi_{\texttt{voiced}}(x) \stackrel{\text{def}}{=} \texttt{voiced}(x)$ and so on. What about the binary successor relation? Letting $\phi_{\triangleleft}(x, y) \stackrel{\text{def}}{=} x \triangleleft y$ is sufficient. While it is true that $3 \triangleleft 4$ is true in the input, the fact that 4 is not licensed ensures that the pair $(3, 4)$ is not an element of the successor relation in the output model. The relations in the output structure are always *restricted* to tuples which only contain *licensed* domain elements. Readers are referred to Chapter 6 for details.

For completeness, Table 3.2 shows the complete logical description of word-final vowel deletion.

$\phi_{\texttt{domain}}$	$\stackrel{\text{def}}{=}$	$\texttt{true}$
C	$\stackrel{\text{def}}{=}$	$\{1\}$
$\phi_{\texttt{license}}(x)$	$\stackrel{\text{def}}{=}$	$\neg(\ \texttt{wordfinal}\ (x) \wedge \texttt{vocalic}(x))$
$\phi_{\triangleleft}(x, y)$	$\stackrel{\text{def}}{=}$	$x \triangleleft y$
$\phi_{\texttt{vocalic}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{vocalic}(x)$
$\phi_{\texttt{low}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{low}(x)$
$\phi_{\texttt{high}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{high}(x)$
$\phi_{\texttt{front}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{front}(x)$
$\phi_{\texttt{stop}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{stop}(x)$
$\phi_{\texttt{fricative}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{fricative}(x)$
$\phi_{\texttt{nasal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{nasal}(x)$
$\phi_{\texttt{lateral}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{lateral}(x)$
$\phi_{\texttt{rhotic}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{rhotic}(x)$
$\phi_{\texttt{labial}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{labial}(x)$
$\phi_{\texttt{coronal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{coronal}(x)$
$\phi_{\texttt{dorsal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{dorsal}(x)$
$\phi_{\texttt{voiced}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{voiced}(x)$
$\phi_{\texttt{voiceless}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{voiceless}(x)$

Table 3.2: The complete logical specification for word-final vowel deletion when the input and output string models are both the feature-based successor model.

This section explained in more detail how the domain elements of the output structure are determined. While these are always copies of the domain elements of the input structure, it is not the case that every domain element in the input structure becomes a domain element of the output

structure. Only those elements x which satisfy $\phi_{\texttt{license}}(x)$ become domain elements in the output structure.

3.4 Getting Bigger

So far we have exemplified logical transductions with phonological processes that change segmental material and processes that delete segmental material. How can logical transductions be used to define processes that *add* segmental material?

The answer to this question lies in the copy set. We have set aside this ingredient until now. In the previous examples, the copy set contained only *one* element. Thus each input element in the domain was copied exactly *once*. More generally, the copy set may contain n elements. It follows that the domain of the output model may contain n copies of *each* domain element of the input structure. The copies of a domain element x in the input structure are distinguished from each other using the names of the elements in the copy set. For example, consider the word *hauz* so that the domain elements of $\mathcal{M}_{hauz}$ are $\{1, 2, 3, 4\}$. If we are defining a logical transduction and define the copy set $C \stackrel{\text{def}}{=} \{1, 2\}$ then there are as many as *eight* domain elements in the output structure. It is customary to name these domain elements as *pairs*; the first coordinate indicates the domain element in the input structure being copied and the second coordinate indicates *which* copy. Thus the pair $(1, 2)$ indicates the second copy of the first domain element of the input structure and $(3, 1)$ indicates the first copy of the third element and so on. The eight possible domain elements in the output structure of our example are thus $\{(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2), (4, 1), (4, 2)\}$.

Whenever the copy set contains more than one element, the number of licensing formulas and relational formulas needed to describe the logical transduction multiplies as well. For each $i \in C$, there is a licensing formula $\phi^i_{\texttt{license}}(x)$. As before, this formula is evaluated with respect to the corresponding domain element in the input structure. If it evaluates to true on x then the domain element (x, i) is licensed and belongs to the domain of the output model. Thus for a copy set C, there are $|C|$ licensing formulas.

Similarly, for each unary relation R in the signature of the output model, there are $|C|$ relational formulas: for each $i \in C$, we must define $R^i(x)$. The domain element (x, i) – the ith copy of x in the output structure – belongs

to R in the output structure if and only if $R^i(x)$ evaluates to true in the input structure.

For each binary relation R in the output signature, there are $|C|^2$ relational formulas $R^{i,j}(x,y)$ with $i,j \in C$. If and only if $R^{i,j}(x,y)$ evaluates to true with respect to the input model then the ith copy of x stands in the R relation to the jth copy of y in the output structure. In which case, we have $((x,i),(y,j)) \in R$. If $R^{i,j}(x,y)$ evaluates to false with respect to the input structure then $((x,i),(y,j))$ does not belong to R. For relations of higher arity, the licensing and relational formula multiply out similarly. Since the word models developed so far involve at most binary relations, we ignore relations of higher arity here (though they are treated in the formalizations in Chapter 6).

How the copy set works along with the additional formulas it entails are illustrated in the next two examples: word-final vowel epenthesis and total reduplication. We provide complete logical descriptions of both these transformations.

3.4.1 Word-final vowel epenthesis

Hindi speakers epenthesize the low vowel *a* to words which end in sonorant consonants (Shukla, 2000). We provide a logical description of this process given the segments describable with the feature-based successor model. For example, this process would map the hypothetical word *pan* to *pana* as well as *pak* to *pak*. Figure 3.3 visualizes the mapping between the model structures *pan* and *pana*.

First we can define sonorant consonants as follows.

$$\texttt{sonorant\_C}\,(x) \stackrel{\text{def}}{=} \texttt{nasal}(x) \vee \texttt{lateral}(x) \vee \texttt{rhotic}(x) \tag{3.4}$$

Next, we need a copy set of at least size 2 and so we define $C \stackrel{\text{def}}{=} \{1,2\}$. Consequently, for the input *pan* which has three domain elements $\{1,2,3\}$, there are maximally 6 domain elements in the output structure: $\{(1,1),(2,1),(3,1),(1,2),(2,2),(3,2)\}$. Since the copy set C has two elements, we must define two licensing formulas, each with one free variable.

$$\phi^1_{\texttt{license}}(x) \stackrel{\text{def}}{=} \texttt{true} \tag{3.5}$$

$$\phi^2_{\texttt{license}}(x) \stackrel{\text{def}}{=} \texttt{sonorant\_C}\,(x) \wedge \texttt{wordfinal}\,(x) \tag{3.6}$$

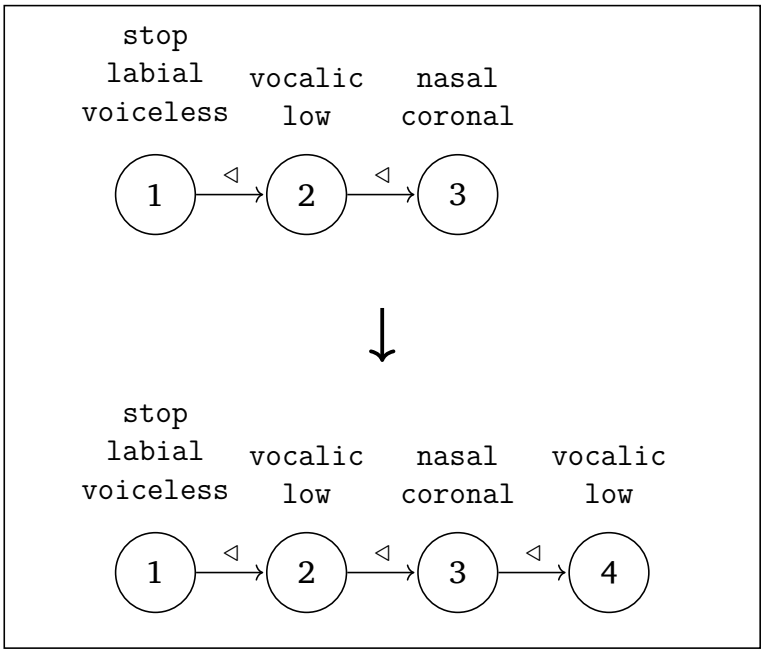


Figure 3.3: A graphical diagram of the feature-based successor model of *pan* being mapped to the feature-based successor model of *pana*.

$\phi^1_{\texttt{license}}(x)$ is always true so the first copy of each element is present. $\phi^2_{\texttt{license}}(x)$ is only true when $\texttt{sonorant\_C}\,(x) \wedge \texttt{wordfinal}\,(x)$ evaluates to true in the input structure. For the word *pan* this occurs for $x = 3$, but for the word *pak* no x satisfies $\phi^2_{\texttt{license}}(x)$. Consequently, the output structure of the process applied to *pan* has four domain elements $\{(1,1),(2,1),(3,1),(3,2)\}$ but the the output structure of the process applied to *pak* has three domain elements $\{(1,1),(2,1),(3,1)\}$.

This is illustrated in Figure 3.4, where the first and second copies of the domain elements of *pan* are arranged in rows and the unlicensed elements are in gray.

Next, we turn to the binary successor relation in the output model. Here, we must have four formulas to specify the successor relation in the

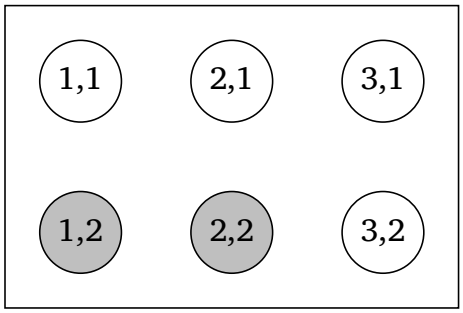


Figure 3.4: The possible domain elements of the output structure for input *pan* when the copy set $C \stackrel{\text{def}}{=} \{1, 2\}$. The unlicensed elements are colored gray.

output structure. We define these as follows.

$$\phi^{1,1}_{\triangleleft}(x, y) \stackrel{\text{def}}{=} x \triangleleft y \tag{3.7}$$

$$\phi^{1,2}_{\triangleleft}(x, y) \stackrel{\text{def}}{=} \texttt{sonorant\_C}\,(x) \wedge \texttt{wordfinal}\,(x) \wedge \texttt{wordfinal}\,(y) \tag{3.8}$$

$$\phi^{2,1}_{\triangleleft}(x, y) \stackrel{\text{def}}{=} \texttt{false} \tag{3.9}$$

$$\phi^{2,2}_{\triangleleft}(x, y) \stackrel{\text{def}}{=} \texttt{false} \tag{3.10}$$

There are two main consequences. First, within the first copy, the domain elements in the output structure preserve the successor relations present in the input structure. Second, the only elements which stand in the successor relation from the first copy to the second copy in the output structure are $x, 1$ and $y, 2$ when x satisfies both $\texttt{wordfinal}\,(x)$ and $\texttt{sonorant\_C}\,(x)$, and when y satisfies $\texttt{wordfinal}\,(y)$.

Finally, we must define two formulas for each unary relation R in the output signature, $\phi^1_{\mathtt{R}}(x)$ and $\phi^2_{\mathtt{R}}(x)$. These will tell us whether $(x, 1)$ and $(x, 2)$ belong to R, respectively. For each unary relation R, we define $\phi^1_{\mathtt{R}}(x) \stackrel{\text{def}}{=} R(x)$. Thus, the first copy of the domain elements are faithful to the unary relations they satisfied in the input. For the second copy, we can generally let the domain elements be faithful to the unary relations they satisfied in the input; however, there are two exceptions. In our feature-based successor model in Table 2.4, the low vowel *a* is low and

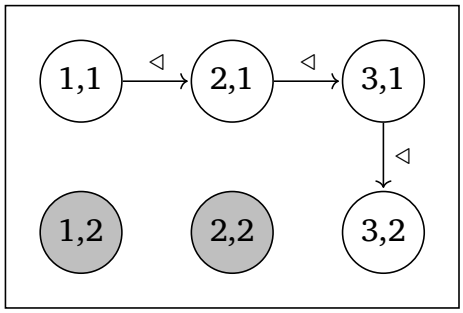


Figure 3.5: The successor relations in the output structure for input *pan* when the copy set $C \stackrel{\text{def}}{=} \{1, 2\}$. The unlicensed elements are colored gray.

vocalic and so $\phi^2_{\texttt{vocalic}}(x)$ and $\phi^2_{\texttt{low}}(x)$ must be defined to be true only when x corresponds to an element in the input that satisfies `sonorant_C` (x) and `wordfinal` (x). For other unary relations R, we can define $\phi^2_{\texttt{R}}(x) \stackrel{\text{def}}{=} \texttt{false}$.

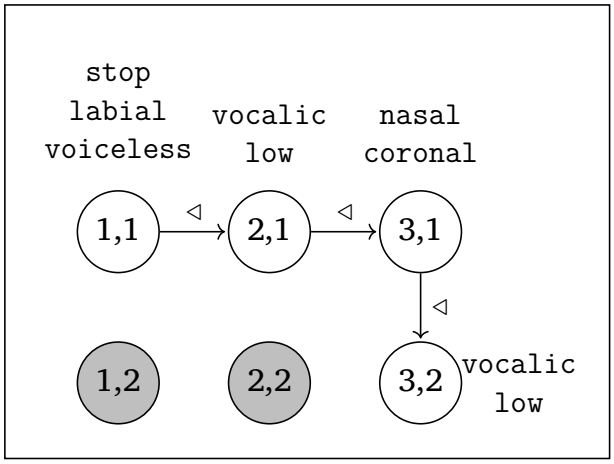


Figure 3.6: The model representing *pana* which is output for the input *pan*. The unlicensed elements are colored gray.

For completeness, Table 3.3 shows the complete logical description of word-final vowel epenthesis. The output structure obtained by applying this logical transformation to $\mathcal{M}_{pan}$ is shown in Figure 3.6. The structure in Figure 3.6 is equivalent (i.e. isomorphic) to the output structure shown in Figure 3.3.

$\phi_{\texttt{domain}}$	$\stackrel{\text{def}}{=}$	$\texttt{true}$	C	$\stackrel{\text{def}}{=}$	$\{1,2\}$
$\phi^1_{\texttt{license}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{true}$	$\phi^2_{\texttt{license}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{sonorant\_C}\,(x)$
					$\wedge\ \texttt{wordfinal}\,(x)$
$\phi^{1,1}_{\triangleleft}(x,y)$	$\stackrel{\text{def}}{=}$	$x \triangleleft y$	$\phi^{1,2}_{\triangleleft}(x,y)$	$\stackrel{\text{def}}{=}$	$\texttt{sonorant\_C}\,(x)$
					$\wedge\ \texttt{wordfinal}\,(x)$
					$\wedge\ \texttt{wordfinal}\,(y)$
$\phi^{2,1}_{\triangleleft}(x,y)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$	$\phi^{2,2}_{\texttt{succ}}(x,y)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{vocalic}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{vocalic}(x)$	$\phi^2_{\texttt{vocalic}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{sonorant\_C}\,(x)$
					$\wedge\ \texttt{wordfinal}\,(x)$
$\phi^1_{\texttt{low}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{low}(x)$	$\phi^2_{\texttt{low}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{sonorant\_C}\,(x)$
					$\wedge\ \texttt{wordfinal}\,(x)$
$\phi^1_{\texttt{high}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{high}(x)$	$\phi^2_{\texttt{high}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{front}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{front}(x)$	$\phi^2_{\texttt{front}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{stop}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{stop}(x)$	$\phi^2_{\texttt{stop}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{fricative}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{fricative}(x)$	$\phi^2_{\texttt{fricative}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{nasal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{nasal}(x)$	$\phi^2_{\texttt{nasal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{lateral}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{lateral}(x)$	$\phi^2_{\texttt{lateral}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{rhotic}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{rhotic}(x)$	$\phi^2_{\texttt{rhotic}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{labial}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{labial}(x)$	$\phi^2_{\texttt{labial}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{coronal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{coronal}(x)$	$\phi^2_{\texttt{coronal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{dorsal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{dorsal}(x)$	$\phi^2_{\texttt{dorsal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{voiced}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{voiced}(x)$	$\phi^2_{\texttt{voiced}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{voiceless}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{voiceless}(x)$	$\phi^2_{\texttt{voiceless}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$

Table 3.3: The complete logical specification for word-final vowel epenthesis when the input and output string models are both the feature-based successor model.

3.4.2 Duplication

Here we provide another example of a logical transduction, total reduplication. The idea is to make two faithful copies of the input and add a successor relation from the last segment of the first copy to the initial segment of the second copy.

Let the copy set $C \stackrel{\text{def}}{=} \{1, 2\}$. Then we essentially make all unary relations be faithful to their input: for all unary relations R in the output signature, let $\phi^1_{\mathtt{R}}(x) = \phi^2_{\mathtt{R}}(x) \stackrel{\text{def}}{=} R(x)$. As for the successor relation, two elements (x, i) and (y, j) stand in the successor relation if only if either one of two cases hold. First, when $i = j = 1$ or $i = j = 2$ then $(x, i) \lhd (y, j)$ only when $x \lhd y$ holds in the input structure. Second, when $i = 1$ and $j = 2$, we have $(x, i) \lhd (y, j)$ if and only if x is word-final and y is word-initial in the input model. When $i = 2$ and $j = 1$, no successor relation holds. We define $\texttt{wordinitial}\,(x)$ as follows.

$$\texttt{wordinitial}\,(x) \stackrel{\text{def}}{=} \neg\exists y\ (y \lhd x) \tag{3.11}$$

To illustrate, Figure 3.7 shows the output structure for the input *pan*. In other words, it is straightforward to define total reduplication using these methods.

For completeness, Table 3.4 shows the complete logical description of total reduplication.

3.4.3 Summary

At this point, we have covered how to define transformations logically. A domain formula determines which words the transformation applies to. In our examples, the transformations represent total functions and apply to all words. The signature of the output structure determines the relational formulas that need to be defined. These formulas belong to a logical language defined in terms of the relations present in the model signature of the input structure. A copy set and licensing formulas are used to calibrate the size of the output structure. For a logical transduction f defined with a copy set of size n, the maximal size of the output structure $f(x)$ will be $n|x|$ where $|x|$ is the cardinality of the domain of the model of x.

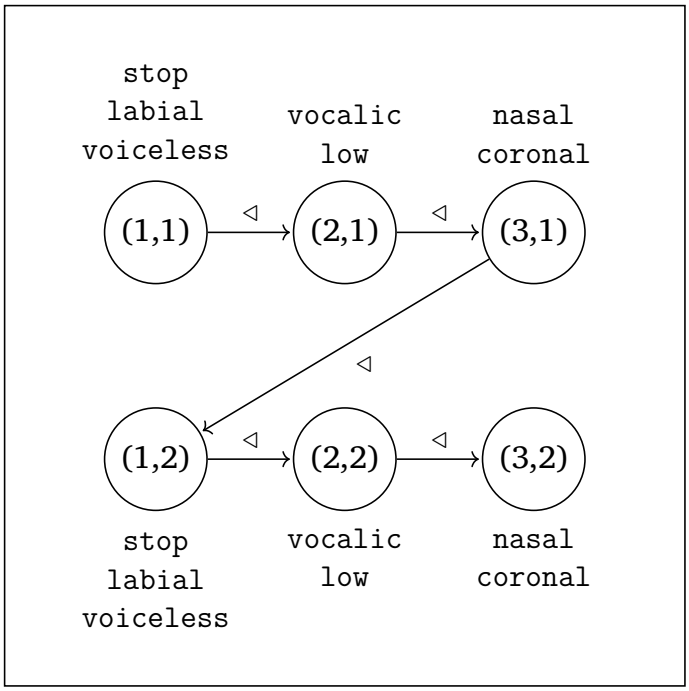


Figure 3.7: The model for *panpan*, which is the output of the reduplication process applying to the input *pan*.

3.5 Power of MSO-definable Transformations

What other kinds of transformations can be described with logical transformations? As the astute reader may no doubt have already gathered, many phonologically or morphologically *unnatural* processes are also easy to describe with logical transformations. This is a strength, not a weakness, of the formal methods advocated here. Basically, the formal methods do not constitute a *theory* of phonology; rather, they constitute a *meta-language* in which theories of phonology can be stated and compared.

In this section, however, we simply wish to establish concretely the fact that two unnatural processes—string mirroring and sorting—also permit logical descriptions.

$\phi_{\texttt{domain}}$	$\stackrel{\text{def}}{=}$	$\texttt{true}$	C	$\stackrel{\text{def}}{=}$	$\{1,2\}$
$\phi^1_{\texttt{license}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{true}$	$\phi^2_{\texttt{license}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{true}$
$\phi^{1,1}_{\triangleleft}(x,y)$	$\stackrel{\text{def}}{=}$	$x \triangleleft y$	$\phi^{1,2}_{\triangleleft}(x,y)$	$\stackrel{\text{def}}{=}$	$\texttt{wordfinal}\,(x)$ $\wedge$ $\texttt{wordinitial}\,(y)$
$\phi^{2,1}_{\triangleleft}(x,y)$	$\stackrel{\text{def}}{=}$	$\texttt{false}$	$\phi^{2,2}_{\texttt{succ}}(x,y)$	$\stackrel{\text{def}}{=}$	$x \triangleleft y$
$\phi^1_{\texttt{vocalic}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{vocalic}(x)$	$\phi^2_{\texttt{vocalic}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{vocalic}(x)$
$\phi^1_{\texttt{low}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{low}(x)$	$\phi^2_{\texttt{low}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{low}(x)$
$\phi^1_{\texttt{high}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{high}(x)$	$\phi^2_{\texttt{high}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{high}(x)$
$\phi^1_{\texttt{front}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{front}(x)$	$\phi^2_{\texttt{front}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{front}(x)$
$\phi^1_{\texttt{stop}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{stop}(x)$	$\phi^2_{\texttt{stop}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{stop}(x)$
$\phi^1_{\texttt{fricative}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{fricative}(x)$	$\phi^2_{\texttt{fricative}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{fricative}(x)$
$\phi^1_{\texttt{nasal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{nasal}(x)$	$\phi^2_{\texttt{nasal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{nasal}(x)$
$\phi^1_{\texttt{lateral}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{lateral}(x)$	$\phi^2_{\texttt{lateral}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{lateral}(x)$
$\phi^1_{\texttt{rhotic}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{rhotic}(x)$	$\phi^2_{\texttt{rhotic}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{rhotic}(x)$
$\phi^1_{\texttt{labial}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{labial}(x)$	$\phi^2_{\texttt{labial}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{labial}(x)$
$\phi^1_{\texttt{coronal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{coronal}(x)$	$\phi^2_{\texttt{coronal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{coronal}(x)$
$\phi^1_{\texttt{dorsal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{dorsal}(x)$	$\phi^2_{\texttt{dorsal}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{dorsal}(x)$
$\phi^1_{\texttt{voiced}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{voiced}(x)$	$\phi^2_{\texttt{voiced}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{voiced}(x)$
$\phi^1_{\texttt{voiceless}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{voiceless}(x)$	$\phi^2_{\texttt{voiceless}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{voiceless}(x)$

Table 3.4: The complete logical specification of total reduplication when the input and output string models are both the feature-based successor model.

3.5.1 Mirroring

String mirroring is a process that takes any string w as input and outputs ww^r where w^r is the reverse of the string w. For example if the string *pan* is submitted to the mirroring process, then the output would be *pannap*. Similarly, if *paka* were input to the mirroring process, the output would be

$pakaakap$. Mirroring makes palindromes.

Mirroring can be described with a logical transduction that is nearly identical to the one for total reduplication. The unary relations are defined in the same way. The only differences lie in two of the formulas for the successor relation in the output structure. Specifically, $\phi^{1,1}_{\triangleleft}(x,y) = x \triangleleft y$ and $\phi^{2,1}_{\triangleleft}(x,y) = \texttt{false}$ as before. However, $\phi^{2,2}_{\triangleleft}(x,y) \stackrel{\text{def}}{=} y \triangleleft x$, which essentially reverses the successor relations in the second copies of the domain elements. Finally, $\phi^{1,2}_{\triangleleft}(x,y) \stackrel{\text{def}}{=} \texttt{wordfinal}(x) \wedge \texttt{wordfinal}(y)$. Thus mirroring places the copies of the word-final element into the successor relation. Figure 3.8 shows the output structure of the string *pannap* that is produced by this logical description of string mirroring given the input *pan*.

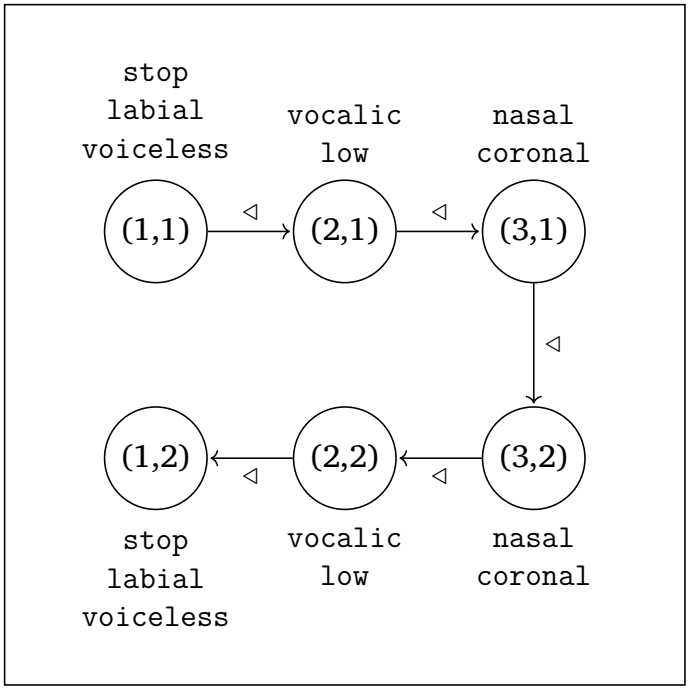


Figure 3.8: The model representing *pannap*, which is the output of the mirroring process applied to the input *pan*.

3.5.2 Sorting

String sorting is a process that takes any string as input and outputs a string of the same length where the symbols are sorted according to a predetermined order; here we will use alphabetical order. For instance if the input string is *paka* then the output string would be *aakp*. Similarly, if the input string was *banapi* the output string would be *aabinp*. While we can do this for any word model of strings discussed so far, we will assume an alphabet Σ and the precedence model with letters (section 2.7) for convenience. We also assume that the alphabet is totally ordered and denote this ordering relation with $<_\alpha$. In a phonological setting, the alphabetical order could be substituted for any other scale, for example markedness. This would allow one to express constraints like "the more marked a segment, the earlier it occurs in a word" or the "the more marked a segment, the later it occurs in a word."

Then sorting can be modeled with a logical transduction as follows. The idea is to have one copy associated with each letter a of the alphabet. Only letters a are licensed on the copy associated with a. This segregates the letters by the copies (which are rows in the visualizations) and the first copy (row) is associated with the first letter in lexicographic order, the second copy (row) with the second letter, and so on. The ordering relation is then defined so that earlier copies (rows) precede later copies (rows).

Formally, let the copyset $C = \Sigma$. This may seem unusual, but it means that we make as many copies as there are letters in the alphabet and that instead of labeling these copies with numbers, we label them with elements of Σ itself. This facilitates defining the formulas. For each $a, b \in \Sigma$, define the relational and licensing formulas as follows.

- For all a, b, let $\phi^b_{\mathtt{a}}(x) \stackrel{\text{def}}{=} \mathtt{a}(x)$.
- For all a, let $\phi^{a,a}_{<}(x, y) \stackrel{\text{def}}{=} x < y$.
- For $a \neq b$, let $\phi^{a,b}_{<}(x, y) \stackrel{\text{def}}{=} \mathtt{true}$ whenever $a <_\alpha b$ and $\mathtt{false}$ otherwise.
- For all a, let $\phi^a_{\mathtt{license}}(x) \stackrel{\text{def}}{=} \mathtt{a}(x)$.

The first item faithfully copies the unary relations in the input to each copy in the output.

The second item defines the binary precedence relations for domain elements in the output that belong to the same copy. In this case, domain elements (x, a) and (y, a) stand in the precedence relation in the output if and only if $x < y$ in the input. This ensures the familiar left-to-right ordering among elements, at least within a copy.

The third item defines the binary precedence relations for domain elements in the output that belong to different copies. The basic idea is that alphabetically earlier copies will precede alphabetically later ones. Whenever $a <_\alpha b$ (a is alphabetically earlier than b) is true then $\phi_<^{a,b}(x, y) \stackrel{\text{def}}{=} \texttt{true}$. Whenever $a <_\alpha b$ is not true, then $\phi_<^{a,b}(x, y) \stackrel{\text{def}}{=} \texttt{false}$.

Finally, we get to the licensing formulas, of which there will be $|\Sigma|$. We define these formulas so that only those domain elements that belong to the unary relation a in the ath copy are licensed. Everything else is unlicensed. Recall that relations in the output structure are restricted to the licensed domain elements. As a consequence of these licensing formulas, there will be only as many licensed elements as there are domain elements in the input structure.

Figure 3.9 illustrates this construction when the input is *paka*.

3.5.3 Summary

Both mirroring and sorting can be described with MSO logical transformations. In fact, mirroring only used FO with successor, and sorting only used FO with precedence.

3.6 Discussion

There are three important points which must be mentioned. The first is that the model signatures for the input and output structures of a transformation do not need to be the same. The examples earlier in this chapter kept the input and output structure signatures the same in order to explain how the logical transformations worked. However, generally, they can be distinct. As will be explained, this has important consequences for the comparison of representational theories.

The second point regards a useful property of some transformations; namely, **order preservation**. In the domains of morphology and phonology,

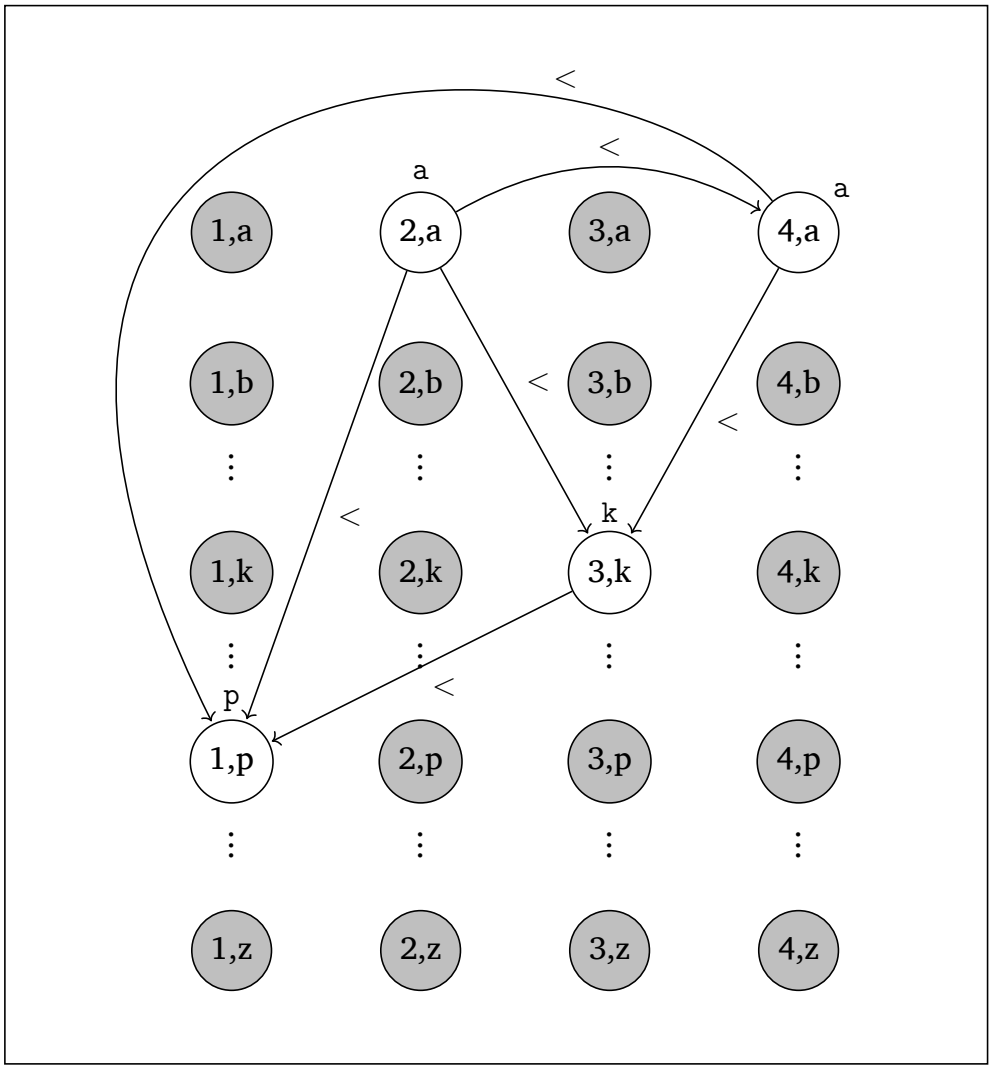


Figure 3.9: The model representing the output *aakp* of the sorting process applied to input *paka*. All potential domain elements shown. Unlicensed elements are in gray. Unary relations are only shown on licensed elements.

this turns out to be nearly universal. Reduplicative morphology is the exception (Dolatian and Heinz, 2020; Rawski *et al.*, 2023). As will be explained, if the transformation we are describing is order-preserving then describing processes like epenthesis and deletion become simpler because we can use the 'order preservation recipe' instead of trying to work out the

order relations by hand.

The third point is that logical transformations provide a feasible, flexible, descriptive tool for linguists to describe phenomena they find in the world, which can be further analyzed and used by many, both human and machine. Model theory and logic provide a universal language for expressing linguistic generalizations.

3.6.1 Transforming Representations

As mentioned, the logical transductions presented in this chapter so far all have used the same model signature for the input and output structures. In general, however, they can be different. For example, many phonologists consider syllable structure to be something not present in underlying representations but present in surface representations. The model theoretic signature for underlying representations would not include those syllabic relations, but the model theoretic signature for surface representations would. Chapter 11 presents an example of this.

To illustrate, we can write logical transductions between any of the model signatures we have considered so far: the successor model, the precedence model, the successor model with features, and the precedence model with features.

Consider a logical transduction which translates successor-based structures to precedence-based structures. For simplicity, let the alphabet be {a,b}. The input structure signature is thus $\{a, b, \lhd\}$ and the output structure signature is $\{a, b, <\}$. Table 3.5 provides the formulas for this logical transduction. The predicate `closed` is defined in Chapter 2 (see Equation 2.14). It is left as an exercise for the reader to write the transduction from the precedence-based structures to successor-based ones.

As another example, one can write translations from symbol-based models to feature-based models in FO logic straightforwardly. For example, to translate between the precedence model and the precedence model with features from Chapter 2, the logical formulas presented in Table 3.6 can be used. It is left as an exercise for the reader to complete Table 3.6 and to write a logical transduction from feature-based structures to conventional ones.

It is also possible to write a FO translation from representations with unary features to representations with binary features and vice versa. In this

$\phi_{\texttt{domain}}$	$\stackrel{\text{def}}{=}$	$\texttt{true}$
$\phi^1_{\texttt{license}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{true}$
C	$\stackrel{\text{def}}{=}$	$\{1\}$
$\phi_<(x,y)$	$\stackrel{\text{def}}{=}$	$(\forall X)\big[(x \in X \wedge \texttt{closed}\,(X) \rightarrow y \in X\big]$
$\phi_{\texttt{a}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{a}(x)$
$\phi_{\texttt{b}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{b}(x)$

Table 3.5: The complete logical specification for translating successor-based models of words in $\{a,b\}^*$ to precedence-based models.

$\phi_{\texttt{domain}}$	$\stackrel{\text{def}}{=}$	$\texttt{true}$
$\phi^1_{\texttt{license}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{true}$
C	$\stackrel{\text{def}}{=}$	$\{1\}$
$\phi_<(x,y)$	$\stackrel{\text{def}}{=}$	$x < y$
$\phi_{\texttt{vocalic}}(x)$	$\stackrel{\text{def}}{=}$	$\texttt{a}(x) \vee \texttt{e}(x) \vee \texttt{i}(x) \vee \texttt{o}(x) \vee \texttt{u}(x)$
$\phi_{\texttt{b}}(nasal)$	$\stackrel{\text{def}}{=}$	$\texttt{n}(x) \vee \texttt{m}(x)$
$\ldots$		

Table 3.6: The complete logical specification for translating successor-based models of words in $\{a,b\}^*$ to precedence-based models.

regard, readers may be interested in (Nelson, 2022), whose comparative study of the natural classes obtained by unary and binary feature systems, and the logical connectives used to combine them, reveals that logical negation effectively converts any feature system into a full binary one and that in order to effectively represent underspecification or non-binary feature oppositions, feature values should be encoded into the representational primitives.

One important consequence of being able to use logical transductions to describe translations between representations is that the weakest logic which can translate one representation into another can serve as a proxy for how similar those representations are. The more powerful the logic necessary to translate between two representations, the more significantly

different they are. In a sense, the minimally expressive logics required to translate between two representations are a measure of the informational content carrying by the representation. Conversely, the weaker the logic necessary to translate between them, the more they can be considered notational variants.

As an example, we have already seen that translating from a successor-based representation to a precedence-based representation requires MSO logic. On the other hand, translating from precedence-based representation to successor only requires FO logic. This indicates that the precedence-based representation carries more information than the successor-based one. For another example, (Strother-Garcia, 2019) shows that different linguistic representations of syllable structure can be translated to each other with a Quantifier-Free logic, which is weaker than First Order (see Chapter 22 for Quantifier-Free logic). Strother-Garcia's results indicates that the information content in those different linguistic representations of syllable structure are not so different. Other examples of this kind of research include studying different theories of tonal geometry (Oakden, 2020), and autosegmental representations vis a vis Q-theory (Jardine *et al.*, 2021).

The second major consequence of being able to use logical transductions to translate between representations is that such transformations actually provide a translation between *logical languages*. This means that if Abraham describes a theory of phonology with logic L and representation X (so L(X)) , and Barbara describes a theory of phonology with logic M and representation Y (so M(Y)), and Charlie presents a logical transduction T from X-representations to Y-representations then every constraint formula or sentence ϕ that can be expressed in L(X) can be translated into a formula or sentence in T(Y). For example, any first order constraint expressed over the successor model with letters can be translated into a FO order constraint over the successor model with features because a FO definable transduction exists from the conventional successor model to the successor model with features.

3.6.2 Order Preservation

A transformation is **order-preserving** provided there is some logical transduction such that for all inputs, the outputs are ordered so that all the

copies of the first position precede all copies of the second position and so on; and furthermore, it is the case that that within a copy, earlier positions precede later positions. It can be helpful to visualize order-preservation as follows. The domain of the input structure D and the copy set C form a $D \times C$ grid of possible output domain elements. In the visualizations throughout this chapter, these grids have $|D|$ columns and $|C|$ rows. If the elements in the output structure can be ordered by ordering elements according to earlier columns first and then by earlier rows, then the transformation is order-preserving. Figure 3.10 illustrates this order with four positions and a copy set of size four. In order to obtain order preservation,

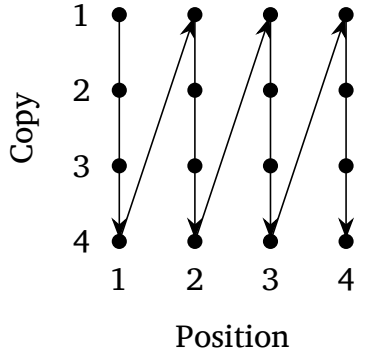


Figure 3.10: Order preservation

using the precedence relation, one simply asserts $\phi_{<}^{i,j}(x, y)$ is true whenever $x < y$ or whenever $x = y$ and $i < j$. Otherwise it is false.

The use of the precedence relation for order-preserving functions is especially helpful when not all domain elements are licensed. In this case, no special modifications need to be made to how the ordering relation is defined. This is because the relations in the output structure are restricted to the licensed domain elements and the precedence relation is total (so for any two elements, one has to precede the other).

The fact that the successor relation is not total means that writing formulas for the successor relation for order-preserving transformations is more complicated. Nonetheless, it can be done. A general solution is to write the formulas using the licensing function to ensure that the successor relation only holds between the appropriate licensed elements. One way to do this uses the definition of general precedence in the output structure $\phi_{<}$ above. Let $\phi_{\lhd}^{i,j}(x, y)$ to be defined as true if and only if (1) (x, i) and

(y, j) satisfy the relevant licensing formulas, (2) $\phi_<^{i,j}(x, y)$ is true, and (3) for all (z, k) such that if (z, k) is ordered between (x, i) and (y, j) then it must *not* satisfy the relevant licensing formulas. In other words, the licensed elements form a 'tier' and the successor relation is just the 'next' element on this tier.

Because there are general ways to write the order relations when the transformation is order-preserving, it follows that one can focus on the other formulas needed to define the relation. It also helps guide the analysis. For example, if there is epenthesis occurring between positions x and $x + 1$ then, in order to take advantage of order-preservation, the epenthesized element should go on a copy of x, and not on a copy of some other element.

3.6.3 Logic as a descriptive formalism

There are many reasons why linguists should use logic and model theory as a descriptive formalism. I highlight three: flexibility, theory comparison, and longevity. To some extent, these follow from the fact that logic and model theory provide a universal description language for structures.

Many linguists adopt representational choices as part of their analyses. Model theory and logic do not hinder this freedom. Linguists can choose their representations. These representations and the generalizations made over them can be expressed precisely with logic. Later chapters in this book provide some interesting examples of the kinds of representations that can be explored, especially within phonology and morphology. For example, **TODO: add refs to later chapters.**

Logic and model theory also facilitate theory comparison. Other linguists, either contemporary or belonging to later generations, can take descriptions of linguistic structures and constraints that have been presented with certain representations and logical languages and rigorously translate them into their own preferred representations and constraint languages. These logical languages can then *be compared* to find genuine areas where the theories make different predictions.

A third reason is longevity. If the linguistic description is, for example, in first order logic, one can be assured that someone in *hundreds of years* will be able to read and understand the description, and that machines will exist which can process it. The value of this for someone documenting languages should not be underestimated.

3.7 Conclusion

This chapter has explained how transformations can be expressed logically between model-theoretic representations following the ideas of Courcelle. The model signature of the output representation, together with the copy set, determines the formulas one needs to write. These formulas are written in a logical language based on the model signature of the input representation, and are likewise evaluated against the input structure.

Specifically, in order to specify a transformation, one must specify the following items.

- A formula with no free variables that establishes the domain of the transformation φ_{dom}. This determines those structures to which the function can apply.
- A copy set C of $k \geq 1$ elements which determines, along with the size of the input structure, the maximal size of the output structure. Each pairing (x, c) with $c \in C$ and x in the domain of the input structure is a possible element in the domain of the output structure.
- For each element c in the copy set, a licensing formula of one free variable $\varphi^c_{license}(x)$ which determines whether (x, c), which is the cth copy of element x, is licensed in the model of the output structure. Unlicensed elements are not part of the domain of output structure. Only licensed ones are.
- For each element c in the copy set, and for each unary relation U in the signature of the output structure, a formula $\varphi^c_U(x)$ of one free variable. This formula means that the cth copy of x bears the unary relation U in the output structure if only if
  1. the cth copy of x is licensed, and
  2. $\phi^c_U(x)$ is satisfied when evaluated against the input structure.
- For every pair of elements (c, d) in the copy set and for each binary relation B in the signature of the output structure, a formula $\varphi^{c,d}_B(x, y)$ of two free variables. This formula means that the cth copy of x stands in the relation B to the dth copy of y in the output structure if only if

1. both cth copy of x and the the dth copy of y are licensed, and
2. $\phi_B^{c,d}(x, y)$ is satisfied when evaluated against the input structure.

These formulas can be specified in any order as long as they are well-defined. Thus one formula φ can be defined in terms of another formula ψ only if ψ has been defined previously. MSO (and thus FO) logic do not permit recursive definitions.

The bulk of Parts II and III of this book apply these logical transformations to case studies and theoretical questions in linguistics, especially in phonology and morphology. The introduction to model theory provided in chapters 2 and 3 provide all the necessary background to understand the chapters in the later parts of this book. The remaining chapters in Part I do not be read to understand the work in Part II. While Chapter 4 is useful background to some of the first chapters in Part III, it is not relevant to later chapters in Part III.

The remaining chapters in Part I provide additional context and enrichment to the material presented thus far. Chapter 4 discusses *weighted* logics and explains how logic can also be used to describe non-categorical generalizations. Chapter 5 introduces logic weaker than FO logic for defining phonological constraints, and provides a logical perspective on work on the computational nature of phonological constraints (not transformations) studied in subregular approaches to phonology (Heinz, 2007, 2010; Rogers *et al.*, 2013; Rogers and Lambert, 2019b,a). Chapter 6 presents a rigorous mathematical treatment of models, signature, structures, MSO and FO logic, weighted logics, and propositional logics, that were introduced in Part I of this book.

Chapter 4

Weighted Logics

JEFFREY HEINZ

The logical sentences considered in previous chapters evaluate to `true` or `false` with respect to a model-theoretic structure. This leads some to believe that logical approaches have nothing to say for phenomena that does not fall into binary categories. In fact, however, the use of logic for the description of linguistic constraints and transformations does not preclude studying other kinds of linguistic generalizations, such as those deemed gradient or probabilistic. Weighted logics can be used for precisely such generalizations, among many other purposes (Droste and Gastin, 2009). The sentences can evaluate to a natural number, a real number, a string, or even a set of strings.

4.1 Four Key Points

In order to understand weighted logics, there are a few key points. The first is to realize that existential and universal quantification are essentially recipes for generating a series of disjunctions and conjunctions, respectively. For example the formula $\exists(x)\phi(x)$ is equivalent to the expanded formula $\phi(1) \vee \phi(2) \cdots \vee \phi(n)$ for structures whose domain equals $\{1, 2, \ldots n\}$. Similarly, the formula $\forall(x)\phi(x)$ is equivalent to the expanded formula $\phi(1) \wedge \phi(2) \cdots \wedge \phi(n)$ for structures with the same domain.

The second key point is that the concepts of disjunction and conjunction can be generalized to other binary operations. Generally, disjunction is

understood as a kind of addition, and conjunction is understood as a kind of multiplication. It is conventional to use the symbols $\oplus$ and $\otimes$ for these more general addition and multiplication operators. Consequently, when reading a formula of weighted logic, existential quantification and disjunction are interpreted with $\oplus$ and universal quantification and conjunction are interpreted with $\otimes$.

How the general addition and multiplication operators are instantiated depends on the kinds of values (weights) the logical formula are supposed to evaluate to. The value can be a real number or some other class of values. Weighted logics have been most carefully studied when the class of values under consideration is a **semiring**. Semirings are mathematical structures of values that are closed under the two binary operations $\otimes$ and $\oplus$. Additionally, S contains two **identity** elements: 0 for $\oplus$ and 1 for $\otimes$. In fact, the set $\{\texttt{true}, \texttt{false}\}$ is a semiring where the conjunction is $\otimes$, disjunction is $\oplus$, $0 = \texttt{false}$, and $1 = \texttt{true}$. Some examples of different semirings are shown below in Table 4.1.

Name	S	$\oplus$	$\otimes$	0	1
Boolean	$\{\texttt{true}, \texttt{false}\}$	$\vee$	$\wedge$	`false`	`true`
Natural	$\mathbb{N}$	$+$	$\times$	0	1
Real Interval	$[0, 1]$	$+$	$\times$	0	1
Viterbi	$[0, 1]$	`min`	$\times$	0	1
Finite Language	FIN	$\cup$	$\cdot$	$\emptyset$	$\{\lambda\}$

Table 4.1: Example Semirings

The third key point is that the logical language for the weighted logic presented here differs syntactically from the logical languages discussed previously in one important respect. For the weighted logic presented here, negation can only be applied to atomic expressions. In other words, negation cannot be applied to any well-formed formula to obtain another well-formed formula. The syntactic and semantic details are presented explicitly in Chapter 6.

Here is some rational for why negation is treated this way in weighted logic. As we will see, expressions in weighted logics evaluate to an element of the semiring. In the Boolean semiring, where we have `true` and `false`, it is clear how to interpret negation. But how do we interpret negation in an

arbitrary semiring? A natural interpretation would be to interpret negation as an **inverse operation**, but semirings are not required to contain inverses. To put it another way, semirings are not necessarily closed under inverses. For example, the negation of a natural number is not a natural number.

While one approach may be to only consider semirings which are closed under inverse, the approach pursued by Droste and Gastin (2009) is to restrict negation to atomic expressions. To illustrate, consider the atomic expression $\mathsf{a}(x)$. If a is true of x then this expression will evaluate to the identity of $\otimes$, which is 1. And the expression $\neg\mathsf{a}(x)$ would evaluate to identity of $\oplus$, which is 0. On the other hand, if a is false of x then these expressions will evaluate 1 and 0, respectively.

The fourth key point is that the elements of the semiring are atoms themselves in the logic. For example, in the natural semiring, the number 4 is a term! And syntactically, $4 \wedge 3$ is a well-formed expression. When we interpret it, conjunction is interpreted as $\otimes$, which in the natural number semiring is normal multiplication. So the denotation of $4 \wedge 3$, written $[\![4 \wedge 3]\!]$, is $4 \times 3 = 12$.

4.2 Examples

The remainder of this chapter illustrates with examples weighted logic formulas and their evaluation.

The first example shows how to count the number of marked structures in strings.

$$ {}^{*}\mathsf{c} \stackrel{\text{def}}{=} \exists x[\mathsf{c}(x)] \tag{4.1}$$

Consider how this formula is evaluated with respect to (a representation of) the string $acbc$ in the natural number semiring. The structure of $acbc$ has domain equal to $\{1, 2, 3, 4\}$. In the equations below we simply write $\mathcal{M}$ for $\mathcal{M}_{acbc}$. The existential quantifier in ${}^{*}c$ expands to a series of disjunctions, one for each position in the string.

$$[\![{}^{*}\mathsf{c}]\!](\mathcal{M}) = [\![\bigvee_{x \in \{1,2,3,4\}} \mathsf{c}(x)]\!](\mathcal{M}) = [\![\mathsf{c}(1) \vee \mathsf{c}(2) \vee \mathsf{c}(3) \vee \mathsf{c}(4)]\!](\mathcal{M})$$

Those disjunctions are interpreted as $\oplus$.

$$[\![\mathsf{c}(1)]\!](\mathcal{M}) \oplus [\![\mathsf{c}(2)]\!](\mathcal{M}) \oplus [\![\mathsf{c}(3)]\!](\mathcal{M}) \oplus [\![\mathsf{c}(4)]\!](\mathcal{M})$$

In the natural number semiring, $\oplus$ is normal addition ($+$).

$$[\![\mathtt{c}(1)]\!](\mathcal{M}) + [\![\mathtt{c}(2)]\!](\mathcal{M}) + [\![\mathtt{c}(3)]\!](\mathcal{M}) + [\![\mathtt{c}(4)]\!](\mathcal{M})$$

Each atomic expression evaluates to 1 or 0 depending on whether it is `true` of `false`, respectively, in the structure $\mathcal{M}$.

$$0 + 1 + 0 + 1 = 2$$

This is how weighted logics can evaluate to values other than `true` or `false`. The reader can easily verify that the same procedure will correctly evaluate the structure of the string $abba$ to 0.

The second example shows how to count the length of a string. Again, we use the natural number semiring.

$$\texttt{length} \stackrel{\text{def}}{=} \exists x[1] \tag{4.2}$$

The 1 in the equation is a valid term because 1 is a natural number! Recall it was mentioned that elements of the semirings are allowed to be atomic terms. If we consider string $acbc$ again and its structure $\mathcal{M}$, we can see how `length` is evaluated. The existential quantifier in `length` expands to a series of disjunctions, one for each position in the string.

$$[\![\ \texttt{length}\ ]\!](\mathcal{M}) = [\![\bigvee_{x \in \{1,2,3,4\}} 1]\!](\mathcal{M}) = [\![1 \vee 1 \vee 1 \vee 1]\!](\mathcal{M})$$

Again, in the natural number semiring, $\vee$ is interpreted as normal addition, and so $[\![1 \vee 1 \vee 1 \vee 1]\!](\mathcal{M})$ evaluates to $1 + 1 + 1 + 1 = 4$.

The next example implements a unigram distribution over a three letter alphabet. For this we will use the real interval semiring.

$$\texttt{U} \stackrel{\text{def}}{=} \forall x \Big[\big(\mathtt{a}(x) \wedge 0.4\big) \vee \big(\mathtt{b}(x) \wedge 0.4\big) \vee \big(\mathtt{c}(x) \wedge 0.2\big) \Big] \tag{4.3}$$

This equation essentially assigns the probabilities of 0.4 to occurrences of `a` and `b` and a probability of 0.2 to an occurrence of `c`. Below is the evaluation of `U` with respect to the structure $\mathcal{M}$ of the string $acbc$. The universal quantifier will expand to a series of conjunctions of terms. In this example, those terms themselves are compositions of subterms. Since $\otimes$ and $\oplus$ are interpreted as normal multiplication and addition respectively in

the real interval semiring, the term $\big(\mathtt{a}(x) \wedge 0.4\big) \vee \big(\mathtt{b}(x) \wedge 0.4\big) \vee \big(\mathtt{c}(x) \wedge 0.2\big)$ will be interpreted as $\big(\mathtt{a}(x) \times 0.4\big) + \big(\mathtt{b}(x) \times 0.4\big) + \big(\mathtt{c}(x) \times 0.2\big)$ for each x in the domain. Consequently, when evaluating $U(\mathcal{M}_{acbc})$, it first expands as follows.

$$\begin{aligned}
[\![\ \mathtt{U}\ ]\!](\mathcal{M}) \;=\; & \Big(\big(\mathtt{a}(1) \times 0.4\big) + \big(\mathtt{b}(1) \times 0.4\big) + \big(\mathtt{c}(1) \times 0.2\big)\Big) \\
& \times \Big(\big(\mathtt{a}(2) \times 0.4\big) + \big(\mathtt{b}(2) \times 0.4\big) + \big(\mathtt{c}(2) \times 0.2\big)\Big) \\
& \times \Big(\big(\mathtt{a}(3) \times 0.4\big) + \big(\mathtt{b}(3) \times 0.4\big) + \big(\mathtt{c}(3) \times 0.2\big)\Big) \\
& \times \Big(\big(\mathtt{a}(4) \times 0.4\big) + \big(\mathtt{b}(4) \times 0.4\big) + \big(\mathtt{c}(4) \times 0.2\big)\Big)
\end{aligned}$$

The subterms within the term $\big(\mathtt{a}(x) \wedge 0.4\big) \vee \big(\mathtt{b}(x) \wedge 0.4\big) \vee \big(\mathtt{c}(x) \wedge 0.2\big)$ are mutually exclusive. Position x must satisfy exactly one of a, b, and c. As a result, two of the subterms will evaluate to zero. Consequently, the evaluation will continue as follows.

$$\begin{aligned}
[\![\ \mathtt{U}\ ]\!](\mathcal{M}) \;&=\; (1 \times 0.4 + 0 + 0) \\
&\quad \times (0 + 0 + 1 \times 0.2) \\
&\quad \times (0 + 1 \times 0.4 + 0) \\
&\quad \times (0 + 0 + 1 \times 0.2) \\
&=\; 0.0064
\end{aligned}$$

Other probability distributions over sequences can be expressed in similar ways.

Our final example makes use of the language semiring to express an optional post-nasal voicing generalization. The equation below identifies post-nasal voicing environments. For simplicity we assume the successor model with letters, and we limit the alphabet to the symbols $\{a, n, d, t\}$.

$$\mathtt{NT} \stackrel{\text{def}}{=} \exists x, y \Big[x \lhd y \wedge \mathtt{n}(x) \wedge \mathtt{d}(y) \Big] \tag{4.4}$$

Given the Boolean semiring, the sentence NT would evaluate to true given the structure of the string $anda$ and to false given the structure of the string $anta$.

However, we now want to consider the finite language semiring FIN. The $\otimes$ operation is now language concatenation. Given two sets of strings X and Y, their concatenation is $XY = \{xy \mid x \in X, y \in Y\}$. Note that the empty set acts as 0 here and the set containing only the empty string acts as

1, the identity. In other words, for all finite languages X, $X\varnothing = \varnothing X = \varnothing$ and $X\{\lambda\} = \{\lambda\}X = X$. The $\oplus$ operation is now union. We have seen in the previous example that by multiplying the base terms with elements of the semiring we can associate different weights to different symbols. The same approach is in the next equation, where the substring nd is ultimately associated with the finite language $\{nd, nt\}$.

$$\texttt{NT} \stackrel{\text{def}}{=} \exists x, y \Big[x \lhd y \wedge \texttt{n}(x) \wedge \{n\} \wedge \texttt{d}(y) \wedge \{d, t\} \Big] \tag{4.5}$$

To see how this works, let's evaluate `NT` on the structure of the string nd. Since the domain of this structure has two elements, the existential quantifier expands to four disjunctive terms which correspond to the (x, y) pairs $(1, 1), (1, 2), (2, 1)$, and $(2, 2)$. Each disjunctive term is the conjunction of three subterms. The first subterm is $x \lhd y$. This only evaluates to `true` for (x, y) pair $(1, 2)$. The others evaluate to `false`.

Consider one of the `false` cases, say $x = 1$ and $y = 1$. Since 1 is not the successor of itself, the subterm $x \lhd y$ evaluates to `false`. False terms are interpreted as 0, which in the finite language semiring corresponds to the emptyset. So here $[\![x \lhd y]\!] = \varnothing$. Since conjunction is interpreted as language concatenation, we are 'multiplying' the other subterms by the emptyset. Consequently, this entire disjunctive term evaluates to $\varnothing$. Likewise, the other `false` cases evaluate to $\varnothing$.

Let's move on to the one `true` case when $x = 1$ and $y = 2$. Now $x \lhd y$ evaluates to `true` which is interpreted as the multiplicative identity $\{\lambda\}$. The other subterms evaluate as follows. Since position 1 is a n, $[\![\texttt{n}(x) \times \{n\}]\!] = \{\lambda\}\{n\} = \{n\}$. And since position 2 is a d, $[\![\texttt{d}(y) \times \{d, t\}]\!] = \{\lambda\}\{d, t\} = \{d, t\}$. These three subterms multiply together: $\{\lambda\}\{n\}\{d, t\} = \{nd, nt\}$. This disjunctive term this evaluates to $\{nd, nt\}$.

Finally, we combine all the disjunctive terms together with $\oplus$, which in this semiring is union. We have $\varnothing \cup \{nd, nt\} \cup \varnothing \cup \varnothing$, which of course equals $\{nd, nt\}$. To sum up we have shown when `NT` in Equation 4.5 is applied to the structure of the string nd, it evaluates to the set $\{nd, nt\}$.

We are not yet done. Any string without a nd substring given to Equation 4.5 will evaluate to the empty set. This is because every disjunctive term will evaluate to the empty set since none of its subterms will be be true. So to allow the process to apply to any word, it is important that we embed Equation 4.5 into a larger expression.

How do we accomplish this? The next equation establishes basic faithfulness (the identity function).

$$\texttt{id} \quad \stackrel{\text{def}}{=} \quad \forall x \Big[\big(\texttt{a}(x) \wedge \{a\}\big) \vee \big(\texttt{n}(x) \wedge \{n\}\big) \vee \big(\texttt{d}(x) \wedge \{d\}\big) \vee \big(\texttt{t}(x) \wedge \{t\}\big) \Big] \tag{4.6}$$

The idea is to combine Equation 4.6 with Equation 4.5 to achieve the desired outcome. Here is one way to do this.

$$\begin{aligned} \texttt{NT} \quad \stackrel{\text{def}}{=} \quad \forall x \Big[& \big(\texttt{a}(x) \wedge \{a\}\big) \vee \big(\texttt{n}(x) \wedge \{n\}\big) \vee \big(\texttt{t}(x) \wedge \{t\}\big) \\ & \vee \big(\texttt{d}(x) \wedge \big(\exists y[y \lhd x \wedge \texttt{n}(y)]\big) \wedge \{d,t\}\big) \\ & \vee \big(\texttt{d}(x) \wedge \big(\exists y[y \lhd x \wedge \neg\texttt{n}(y)]\big) \wedge \{d\}\big) \Big] \end{aligned} \tag{4.7}$$

The idea is to again to make use of mutually exclusive conditions for each position. In this case there are five. If a position satisfies `a`, `n`, or `t`, it invariably surfaces faithfully. If a position satisfies `d` then it depends on whether a previous position exists which satisfies `n`. If so, then the fourth disjunct will evaluate to $\{d,t\}$ and the last one to $\varnothing$. If not, the fourth disjunct will evaluate to $\varnothing$ and the last one $\{d\}$.

This last example is especially interesting because it provides another way to express transformations with logical formula other than the Courcellian approach introduced in chapter 3, which is used throughout the remainder of this book. The approach to string-to-string functions using weighted logics over a string or language based semiring (like FIN), to my knowledge, has not been studied in any more detail.

A basic idea that informed the examples here has been to use the logical language to identify substructures and to then multiply them by elements of the appropriate semiring. In this way, the outputs are always some kind of sum of the relevant weighted substructures.

Finally, it is also worth observing that given two semirings A and B, a new semiring can be constructed whose elements belong to the cross-product $A \times B$. For example, we could combine the Finite Language semiring with the real interval semiring to express probabilities over the output variations.

4.3 Conclusion

Weighted logics allow one to express linguistic generalizations beyond binarity. There are some technical differences in the ways these logics are defined. Negation only applies to the base cases. Conjunction and disjunction are interpreted as semiring multiplication $\otimes$ and addition $\oplus$. There are *many* semirings (Golan, 1999), including ones for strings and formal languages. While there is much here to explore, the remainder of this book focuses on the use of logic and model theory as described in previous chapters. This chapter is presented to lay to rest any doubts about the efficacy that formal logic brings to non-binary generalizations.

Chapter 5

Below First Order Logic

JEFFREY HEINZ AND JAMES ROGERS

This chapter continues the line of thinking developed in Chapter 2. There it was shown how the choice of constraint definition language (CDL) provides a theory of possible constraints. It was also shown that from a model-theoretic perspective, choice of constraint definition language includes choosing an explicit representation and logical formalism. It was also argued there that if theorists wish to posit a CDL which can express both local and long-distance constraints of the kind found in phonology, but cannot express generalizations like EVEN-N, then, of the CDLs considered, First-Order (FO) logic with precedence would be the best choice.

Another way to motivate First-Order logic with precedence is that, given the CDLs considered so far, it was the *least* class of constraints that included both the local and long-distance style constraints. The idea that "Everything should be made as simple as possible, but no simpler" is at the heart of scientific thinking.[1] We are interested in the *minimally necessary* computational machinery to account for the variety of generalizations observed across the world's languages (cf. the Minimalist Program (Chomsky, 1995)).

One clue that FO is more expressive than necessary, is that it is straightforward to define constraints that are sensitive to the number of occurrences of complex structures in a word. Readers may recall that this counting is

[1] This expression is typically attributed Einstein but it seems it was sharpened after the fact (Robinson, 2018).

in fact present in the abstract characterization of "FO with successor" in Theorem 1.

For example, Equation 2.11 gave a definition for the constraint *NT. It is very easy to write a similar constraint that only penalizes words with three NT sequences but not two as shown below.

$$\begin{aligned} \texttt{*3NT} \stackrel{\text{def}}{=} \neg(\exists x_1, x_2, x_3, x_4, x_5, x_6)\Big[& \\ & \big(x_1 \lhd x_2 \wedge \texttt{nasal}(x_1) \wedge \texttt{voiceless}(x_2)\big) \\ \wedge & \big(x_3 \lhd x_4 \wedge \texttt{nasal}(x_3) \wedge \texttt{voiceless}(x_4)\big) \\ \wedge & \big(x_5 \lhd x_6 \wedge \texttt{nasal}(x_5) \wedge \texttt{voiceless}(x_6)\big) \\ \wedge & \big(x_1 \neq x_3 \wedge x_1 \neq x_5 \wedge x_3 \neq x_5\big)\Big] \end{aligned} \tag{5.1}$$

Note we use $x \neq y$ as shorthand for $\neg(x = y)$. According to this constraint hypothetical words like *kampantasakanka* are ill-formed, but words like *kampantasakaka* are well-formed. Is there a principled way to eliminate this kind of counting from the CDLs?

There is. **Propositional logic** is a logical system that is weaker than FO. In this section we motivate and define a propositional-style logic along the lines developed by Rogers and Lambert (2019b). We do this for both the successor and the precedence models of strings.

The resulting CDLs do not have the ability to count in the manner above. More generally, the abstract characterizations of the resulting CDLs corresponds to a particular type of memory model, the so-called "Testable" classes (McNaughton and Papert, 1971; Simon, 1975), with clear cognitive implications (Rogers and Pullum, 2011; Rogers *et al.*, 2013). We return to these broader issues after introducing propositional logic.

5.1 Propositional Logic with Factors

Sentences of propositional logic are Boolean combinations of **atomic propositions**. The Boolean connectives, presented in Table 2.2, are the symbols: $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation), $\rightarrow$ (implication), and $\leftrightarrow$ (biconditional).[2] Classically, the atomic propositions can be anything from

[2]In technical presentations of propositional logic, only some of these are presented as fundamental and the remainder are derived from those.

sentences like "All men are mortal" to "The sample contained chlorine." The truth of any sentence in propositional logic can be computed from the truth values of the atomic propositions along with the standard ways the Boolean connectives are interpreted. Good introductions to propositional logic include Keisler and Robbin (1996) and Hedman (2004).

Additionally, one way to interpret the meaning of a sentence ϕ in propositional logic is as the set of worlds in the universe for which ϕ would evaluate to `true`. In our context, the universe is the set of possible strings Σ^* and each string in Σ^* is a "world" in this universe. Thus, just as with sentences of FO and MSO logic, each propositional sentence ϕ will pick out some set of strings in Σ^*, which are those words for which ϕ can be said to be true of.

What are the atomic propositions in this universe of strings? Following Rogers and Lambert (2019b), we present atomic propositions based on the notion of **containment**. They are sentences of the form "Words contain S", where S is a model-theoretic **connected** structure. Consequently the proposition S will be `true` of any string w whose model M_w **contains** S. In this case, we say that S **is a factor of** M_w.

In order to precisely define the **factor** relation, we must introduce the meanings of *connected* and *contains*. The formal details are given in Chapter 6 and are illustrated below with examples.

As an example, consider the successor model with features and the structure with domain $D = \{1, 2\}$, the successor relation given by $\{(1, 2)\}$, with `nasal`$=\{1\}$, `voiceless`$=\{2\}$, and with all other unary relations denoting phonological features equal to the empty set. This structure, which we denote `NT`, represents a nasal immediately succeeded by a voiceless segment. It is shown in Figure 5.1.

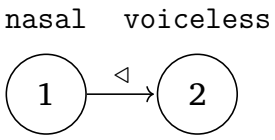


Figure 5.1: The factor NT

Compare this structure with $\mathcal{M}_{sans}$ in the successor model with features presented in Figure 2.2. We can say that the structure `NT` is a factor of

$\mathcal{M}_{sans}$ because $\mathcal{M}_{sans}$ contains the structure NT. This is because we can find elements in the domain of $\mathcal{M}_{sans}$ – namely elements 3 and 4 – which match the elements 1 and 2. By "match", we mean that the relations held by 1 and 2 in NT hold for 3 and 4 in $\mathcal{M}_{sans}$, respectively. What relations are held by 1 and 2 in NT? 1 satisfies the unary relation nasal and 2 satisfies the unary relation voiceless. Additionally, 2 is the successor of 1. We can likewise see that in $\mathcal{M}_{sans}$, 3 satisfies the unary relation nasal, 4 satisfies the unary relation voiceless, and 4 is the successor of 3. For these reasons, we can conclude that $\mathcal{M}_{sans}$ *contains* the structure NT.

However, containment alone is insufficient to define the factor relation. Let's consider another structure N,T. Like NT, this structure has domain $D = \{1, 2\}$ with nasal = {1}, voiceless = {2}, and with all other unary relations denoting phonological features equal to the empty set. Furthermore, no successor relation holds between these two elements. The model $\mathcal{M}_{sans}$ also contains this structure because we can find elements in $\mathcal{M}_{sans}$ which match the elements in N,T. In fact, successor structures of words like *donut* and *ten* also contain the structure N,T.

We choose to eliminate the possibility of such disconnected structures, by requiring the atoms of our propositional logic to be **connected** structures. Informally, a structure is connected if any two elements in a domain can be connected by a *series* of relations that chain the two elements together. For example, let the structure CCC be defined to have domain $D = \{1, 2, 3\}$, to have cons = {1,2,3} with all other unary relations denoting phonological features equal to the empty set, and to have the successor relation given by $\{(1, 2), (2, 3)$. There are three pairs of domain elements: (1,2), (2,3), and (1,3). Clearly pairs (1,2) and (2,3) are connected pairs since they are connected by the successor relation. Pair (1,3) is also connected, however, via the series of successor relations that connects 1 to 2 and then 2 to 3. Formal details are given in Chapter 6 (see also Rogers and Lambert (2019b)).

We refer to connected structures as *factors*.[3] We observe that models of every string in Σ^* is a factor because each is a connected structure. Furthermore, we can now understand why NT *is a factor of* $\mathcal{M}_{sans}$. It is because NT is a connected structure contained within $\mathcal{M}_{sans}$. If a factor S

[3]We avoid the term substructure since it has a distinct meaning in model theory; see Hedman (2004) for instance.

is contained within a structure $\mathcal{M}$, we write $S \sqsubseteq \mathcal{M}$. Hence, $\texttt{NT} \sqsubseteq \mathcal{M}_{sans}$.

At last we can specify the atoms of the propositional logic we introduce. The atoms are factors. Every connected structure contained in some string in Σ^* is an atom. Thus, to decide whether a model of a string $\mathcal{M}$ satisfies a sentence of propositional logic with a structure S as an atom, we will need to decide whether S is a factor of $\mathcal{M}$ as shown in Equation 5.2

$$\mathcal{M} \models \mathcal{S} \text{ iff } \mathcal{S} \sqsubseteq \mathcal{M} \tag{5.2}$$

The remainder of the logic is defined like every other propositional logic. Sentences of propositional logic combine atomic propositions with the Boolean connectives ($\wedge$ conjunction, $\vee$ disjunction, $\neg$ negation, $\rightarrow$ implication, and $\leftrightarrow$ biconditional), and these combinations have their usual meanings. The language associated with a propositional sentence ϕ is also defined in the usual manner.

$$L(\phi) = \{w \in \Sigma^* \mid \mathcal{M}_w \models \phi\} \tag{5.3}$$

As was the case with FO and MSO logics, this propositional-style logic we have introduced depends on a model signature. This is because what the atomic propositions — the connected structures — depend is on the model signature.

5.2 Examples of Propositional Logic with Factors

In this section we discuss the CDLs: PROP($\lhd$), PROP($<$), and PROP($\lhd, \rtimes, \ltimes$), each with and without features. These refer to propositional logical languages defined with factors from the model signatures with successor, precedence, and successor with word boundaries respectively. It can be shown that each of these CDLs is unable to define a constraint that penalizes words with three NT sequences but not two (regardless of whether or not features are used).

Each of these CDLs has a very similar characterization as expressed in the following theorem. We identify the **size of a factor** with the size of its domain.

Theorem 4 (Characterization of PROP-definable constraints)**.** *A constraint is PROP-definable with model signature $\mathfrak{R}$ if and only if there is a number k such that for any two strings w and v, whenever the $\mathfrak{R}$-structures of these two strings have the same factors up to size k under the given model, then either both w and v violate the constraint or neither does.*

That the theorem is true is not hard to see. If two strings w, v have exactly the same factors up to size k then their structures satisfy exactly the same set of atomic propositions. Since the truth of any propositional formula ϕ depends only on the truth or falsity of its atomic propositions, it must be the case that either both $\mathcal{M}_w \models \phi$ and $\mathcal{M}_v \models \phi$ or neither $\mathcal{M}_w$ nor $\mathcal{M}_v$ satisfy ϕ.

Significant literature exists on the classes of formal languages definable with some of these CDLs. In particular the class of formal languages definable with PROP($\lhd$) are Strongly Locally Testable (Beauquier and Pin, 1991). A constraint like *NT is thus not only FO-definable with successor but it is PROP-definable with successor.

To be explicit, if the alphabet is the one used in Chapter 2 ({a, b, d, e, g, i, k, l, m, n, o, p, r, s, t, u, z}) then *NT can be expressed as shown in Equation 5.4 below.

$$*\texttt{NT} \stackrel{\text{def}}{=} \neg\texttt{mk} \wedge \neg\texttt{mp} \wedge \neg\texttt{ms} \wedge \neg\texttt{mt} \wedge \neg\texttt{nk} \wedge \neg\texttt{np} \wedge \neg\texttt{ns} \wedge \neg\texttt{nt} \tag{5.4}$$

In Equation 5.4, the sequence ab represents the factor where the first element is a and the second is b. On the other hand, if we are using features, then *NT can be expressed as shown in Equation 5.5 below.

$$*NT \stackrel{\text{def}}{=} \neg\texttt{NT} \tag{5.5}$$

where NT represents the factor shown in Figure 5.1. We conclude that *NT is definable with PROP($\lhd$) (with or without features) and is therefore a Strongly Locally Testable constraint.

We can also show that the constraint which is violated by three NT sequences but not two is not definable in this way. To show this, we use Theorem 4. Let us call this particular constraint *3NT. If this constraint was definable with PROP($\lhd$), then according to this theorem there would be some maximum factor size k such that for any two strings w and v, whenever the model structures of these two strings have the same factors up to size k under the given model, then either both w and v violate the

constraint or neither does. Consequently, we can show a constraint is not PROP($\triangleleft$) by showing there exists, for any k, two strings with the same set of factors, but one obeys the constraint and one does not.

Pick an arbitrary k and consider the strings $w = a^k nta^k nta^k$ and $v = a^k nta^k nta^k nta^k$. Clearly w obeys *3NT but v does not. And yet these strings have the same set of k-factors: $\{a^k, a^{k-1}n, a^{k-2}nt, a^{k-3}nta, \ldots, nta^{k-2}\}$. Since we can always find a pair of strings that *3NT distinguishes for any k, there is no maximum k such that strings with the same k-factors either both obey or both violate the constraint. It follows from Theorem 4 that *3NT is not definable in PROP($\triangleleft$).

Another class of constraints that has been well studied is definable with PROP($\triangleleft, \rtimes, \ltimes$). This model extends the successor model with left and right word boundaries. The signature of this model is $\{(\mathtt{b})_{b\in\Sigma}\triangleleft, \rtimes, \ltimes, \}$ where symbols $\rtimes$ and $\ltimes$ denote unary relations which are interpreted as the left and right word boundaries, respectively. The model-theoretic representation of a string $w = b_1 b_2 \ldots b_n$ is presented in Table 5.1.

D	$\stackrel{\text{def}}{=}$	$\{0, 1, 2, \ldots n+1\}$
a	$\stackrel{\text{def}}{=}$	$\{i \in D \mid b_i = b\}$ for each unary relation a
$\triangleleft$	$\stackrel{\text{def}}{=}$	$\{(i, i+1) \subseteq D \times D\}$
$\rtimes$	$\stackrel{\text{def}}{=}$	$\{0\}$
$\ltimes$	$\stackrel{\text{def}}{=}$	$\{n+1\}$

Table 5.1: The successor model for words with word boundaries $w = b_1 b_2 \ldots b_n$.

For example, Figure 5.2 shows the structure the successor model with word boundaries assigns to the string *sans*. Under this model, factors can

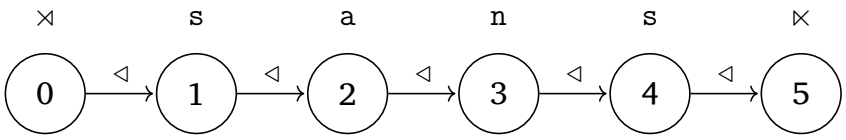


Figure 5.2: A graphical depiction of the successor model with word boundaries of the word *sans*.

distinguish structures at left and right word boundaries from ones that

are not at these boundaries. The class of languages definable with this model and propositional logic is exactly the Locally Testable languages (McNaughton and Papert, 1971; Rogers and Pullum, 2011). The Locally Testable Languages are known to properly include the Strongly Locally Testable languages. Similar arguments to the ones presented above will also show that *3NT is not Locally Testable for any factor size k.

We next address the question of whether features increase or decrease the definable constraints. We observe that two strings with the same factors in a model signature with letters PROP($(\mathtt{b})_{b\in\Sigma}, \lhd, \rtimes, \ltimes$, letters) will also have the same factors in a model signature with features PROP(feat, $\lhd, \rtimes, \ltimes$) and vice versa. So the expressivity of PROP(feat, $\lhd, \rtimes, \ltimes$) and PROP($\lhd, \rtimes, \ltimes$, letters) are the same (the Locally Testable class) and thus no arguments based on expressivity can be used to distinguish these CDLs. However, it is of course the case that *the way* certain sets of strings can be expressed within these logical languages will be different, and arguments for one or the other CDL could be made on such grounds.

The class of formal languages definable with PROP($<$) are Piecewise Testable (Simon, 1975). The constraint *N..L is PROP-definable with precedence with and without features as shown in Figure 5.3 below.

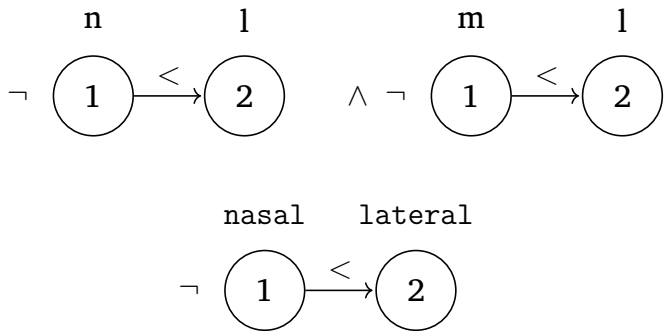


Figure 5.3: The factor NL with letters (above) and features (below)

To summarize this section, a propositional logic whose atomic propositions correspond to factors interpreted in terms of containment provides CDLs that are less powerful than corresponding FO ones. Figure 5.4 illustrates the situation with the constraints discussed in Chapter 2.

	$\lhd$, features	$<$, features
MSO	*N..L, EVEN-N	EVEN-N
FO		
PROP	*NT	*N..L

Figure 5.4: Classifying the constraints *NT, *N..L, and EVEN-N.

5.3 Conjunctions of Negative Literals

The constraints presented above all have the same form. That is, they are the *conjunctions of negative literals*. A literal is an atomic proposition. If P is a literal then its negative literal is simply $\neg P$. In the propositional logic with factors introduced above, conjunctions of negative literals simply mean an expression of propositional logic of the following form.

$$\neg X_1 \wedge \neg X_2 \wedge \ldots \wedge \neg X_n$$

Such an expression simply means "Words are well-formed provided they don't contain X_1 and don't contain X_2 and ...don't contain X_n."

Constraints that can be defined with the logical language CNL($\lhd, \rtimes, \ltimes$) correspond exactly to the class of Strictly Local Languages (McNaughton and Papert, 1971; Rogers and Pullum, 2011). This class of languages has as its defining property Suffix Substitution Closure.

Theorem 5 (Characterization of CNL($\lhd, \rtimes, \ltimes$) constraints)**.** *A constraint is CNL($\lhd, \rtimes, \ltimes$) definable if and only if there is a number k such that for any strings $u_1, v_1, u_2, v_2 \in \Sigma^*$ and for any string x of length $k-1$, whenever u_1xv_1 and u_2xv_2 obey the constraint, it is the case that the string u_1xv_2.*

For example, in the case of *NT, it will turn out that $k = 2$. Since both strings $minato$ and $pungu$ obey the *NT constraint, and since both share a sequence of length $k-1$ (here this is n), then we can identify $u_1 = mi$, $v_1 = ato$, $u_2 = pu$, $v_2 = gu$, and $x = n$. Hence we have $u_1xv_1 = minato$ and $u_2xv_2 = pungu$ and we satisfy the antecedent condition in the statement of the theorem. It follows that the string $u_1xv_2 = mingu$ must also be a string that obeys *NT. And in fact it does. This is true for *all* such strings u_1, v_1, u_2, v_2 and x.

Suffix Substitution Closure (SSC) is an abstract property that holds of any Strictly Local language regardless of the intensional description of the formal language. We could use any of the logical languages discussed so far to define the set of strings which do not violate the constraint *NT. We write a finite-state acceptor or use some other grammatical formalism. The SSC tell us something about the *shape* of a formal language in the same way that having 4 sides and 4 right angles tells us that the shape of a polygon is a rectangle. No intensional description required.

Suffix Substitution Closure can be used to show that certain constraints are NOT Strictly Local (and therefore NOT definable with CNL($\lhd, \rtimes, \ltimes$) by finding, for any k, two strings u_1xv_1 and u_2xv_2 with x the length of $k-1$, which obey the constraint but where u_1xv_2 does not.

Here is an example, consider the formula ϕ in PROP($\lhd, \rtimes, \ltimes$) defined in Equation 5.6.

$$\phi = \mathsf{a} \tag{5.6}$$

This constraint says words must contain the letter a. We can use Suffix Substitution Closure to show that this constraint is not definable with CDL($\lhd, \rtimes, \ltimes$). Fix k. Consider the strings $cc^{k-1}a$ and $ac^{k-1}c$. Both of these obey the constraint since they both contain a. However, when we set $u_1 = c$, $x = c^{k-1}$, $v_1 = a$, $u_2 = a$, and $v_2 = c$ we can see that the substituting the suffix yields $u_1xv_2 = cc^{k-1}c$, which clearly violates the constraint since it contains no a.

The formula in Equation 5.7 provides another example.

$$\phi = \neg\mathsf{a} \rightarrow \neg\mathsf{b} \tag{5.7}$$

A word w obeys this constraint provided the sentence "if w does not contain a then w does not contain b" is true. In other words, words without as must also be without bs. Again, pick a k. Consider the strings $cc^{k-1}c$ and $ac^{k-1}b$. Both of these obey the constraint. The first one obeys it because it contains neither as nor bs. The second one obeys it because it contains an a, and so the antecedent in the conditional is not met. However, when we set $u_1 = c$, $x = c^{k-1}$, $v_1 = c$, $u_2 = a$, and $v_2 = b$ we can see that substituting the suffix yields $u_1xv_2 = cc^{k-1}b$, which clearly violates the constraint since it contains no a but does contain b.

If we change the model signature, other classes of languages are obtained. For example, the constraints definable with CNL($<$) correspond exactly to the Strictly Piecewise Languages (Rogers *et al.*, 2010, 2013). The

constraint *N..L is definable with CNL($<$). This class of languages has as its defining property Subsequence Closure.

Theorem 6 (Characterization of CNL($<$) constraints)**.** *A constraint is CNL($<$) definable if and only if for any string x which obeys the constraint, every subsequence of x also obeys the constraint.*

Like with Suffix Substitution Closure, this is a property of Strictly Piecewise languages independent of the grammatical formalism. Subsequence Closure gives us another kind of shape in the space of formal languages.

The conclusion that we come to is that if we are only interested in defining constraints like *NT and *N..L, a minimally expressive constraint language that does the job is to have constraints drawn from CNL($\lhd$) and CNL($<$). Figure 5.5 illustrates. This is more or less the position adopted by

MSO	*N..L, EVEN-N	EVEN-N
FO		
Prop		
CNL	*NT	*N..L
	$\lhd$, features	$<$, features

Figure 5.5: Classifying the constraints *NT, *N..L, and EVEN-N.

Heinz (2010). A key difference between then and now is that the logical and model-theoretic presentation allows us to more precisely understand the nature of the restrictions on what makes a possible constraint. This is partly because the relationships between the different logical formalisms (MSO, FO, Prop, CNL) are well understood, and partly because we also understand the consequences of certain representational choices.

An important line of research has also examined constraints like *N..L using autosegmental representations, invoking the concept of a phonological tier. Readers are referred to **TODO:add tier refs here...**

5.4 Discussion

When more constraints are examined in more languages, it almost certainly reveals that things may not be as simple as this presentation suggests. But

this discussion was not so much about the correctness of this particular conclusion, as it was to emphasize a way to proceed with analysis.

We seek to formalize linguistic generalizations to help us understand them. Expressing these constraints in a logical language does this in spades. It requires us to be explicit about representations. It requires us to be explicit about the logical formalism. When we combine a model-theoretic representation with a logic, whether it MSO, FO, Propositional, or some fragment thereof like CNL, we have created a Constraint Definition Language, which gives us a class of patterns.

That class of patterns can be studied, and situated with respect to other classes of patterns. Humboldt is famous for having said that language makes "infinite employment of finite means" (von Humboldt, 1999, p. 91), but he also said that to do linguistic typology one needs to have two encyclopedias (Frans Planc, p.c.) One of these encyclopedias is an "Encyclopedia of Types," by which he meant the collection of linguistic generalizations that we go out and find in the world. The other is an "Encyclopedia of Categories," by which he meant some systematic way of putting classifying those types. Different logical languages, parameterized by logical power on the one hand, and model-theoretic representation on the other, provide an unparalleled Encyclopedia of Categories with which we can study linguistic generalizations.

Another consideration is learning. We can ask whether the constraints definable with a particular CDL can be learned, under different definitions of what learning means. It is known that when the maximum factor size is specified to some k, that CNL constraints are efficiently learnable under different definitions of learning. The class of PROP constraints similarly constrained is also learnable, but generally not feasibly. See Lambert *et al.* (2021) for details.

This chapter explored logics weaker than First Order as they could be applied to constraints. What about transformations? This question is more open. It is not straightforward how to synthesize the approach taken in this chapter, which uses containment and propositional logic, with the Courcellian logical transformations explained in Chapter 3. On the other hand, in Chapter 22, Chandlee and Lindell present a significant result by establishing an equivalence between Input Strictly Local functions (Chandlee and Heinz, 2018) and a weaker fragment of First Order logic known as Quantifier Free logic. Another approach, not pursued in this book, utilizes algebraic properties of the transformations to explore weaker

variants (Lambert, 2022; Lambert and Heinz, 2023).

5.5 Summary

In this chapter, we showed how propositional logics can be used to express constraints using the notion of structural containment. It was important that our structures be connected; and we introduced the term *factor* to talk about such connected structures. We observed that many local and long-distance phonotactic constraints belong to a fragment of such a propositional logical language, which is the conjunctions of negative literals. We concluded that there are many logical languages which can be defined and studied to classify phonological constraints.

possible revision: return to the shape metaphor introduced with SSC and subsequence closure. Extract and put in its own short section which also mentions the characterizations of Star Free, from the previous chapter, and LT and PT from this one.

Chapter 6

Formal Presentation of Model Theory and Logic

JEFFREY HEINZ

This chapter presents formal definitions of the syntax and semantics of three logical formalisms discussed in earlier chapters. It draws from Enderton (2001); Courcelle (1994); Engelfriet and Hoogeboom (2001); Hedman (2004) and Courcelle and Engelfriet (2012). The organization of this chapter follows the order of the material in part I of this book. First, relational models, model signatures, and structures are defined. Then the syntax and semantics of MSO logic and FO logic are presented. Next the formulas needed for Courcellian logical transductions where the words are represented with model-theoretic relational structures are presented and it is explained how they are interpreted. The following section defines semirings, and the syntax and semantics of weighted logic. The last section defines connected structures, factors, and the syntax and semantics of a propositional logic whose atomic propositions are connected structures found within words.

6.1 Relational Models and Signatures

A n-ary **relation** R is a relation of arity n. This means it expresses a relation among n different elements. So if D is the domain of elements then R is a

subset of

$$D^n = \underbrace{D \times D \times \ldots \times D}_{n \text{ times}} .$$

For example, a unary relation is a subset of D and a binary relation is a subset of $D \times D$. The arity of a relation R is denoted $\rho(R)$.

A **signature** is a *finite number* of relations, denoted $\mathfrak{R}$. The relations in $\mathfrak{R}$ can be of various arities. Formally, if $n \in \mathbb{N}$ is the number of relations in the signature, let

$$\mathfrak{R} = \langle R_1, \ldots R_n \rangle \text{ such that for all } 1 \leq i \leq n,\ \rho(R_i) > 0.$$

In words, $\mathfrak{R}$ is a tuple of n relations, and R_i is a $\rho(R_i)$-ary relation. A signature can be thought of as a way to define a class of logically possible structures. It can be thought of as expressing the *type* of representations under consideration.

A **relational structure** of type $\mathfrak{R}$, also called a $\mathfrak{R}$-structure, is a tuple $\langle \mathcal{D} \mid (\mathcal{R})_{\mathcal{R} \in \mathfrak{R}} \rangle$. Relational structures are representations of the information that is immediately accessible about an object. The object can be identified as a set elements of a domain with certain relationships which exist among those elements. Since the objects we consider have only finitely many domain elements, these structures are called **finite relational structures**.

If the analyst has a class of objects in mind (for example words) then it is important to ensure that each unique object has some model and that distinct objects have distinct models.

As an example, consider conventional word models. Fix an alphabet Σ. Then a conventional word model has $|\Sigma|$ unary relations, one for each letter of the alphabet, and one binary relation, which is the ordering relation. The two models only differ in the ordering relation. For successor-structures, we require $\lhd = \{(i, i+1) \mid i, i+1 \in D\}$ but for precedence-structures, we require $< = \{(i, j) \mid i, j \in D, i < j\}$.

6.2 MSO Logic for relational models

The difference between MSO and FO logic has to do with **quantification**. Both logics make use of **variables**. MSO makes use of two kinds of variables: variables that range over individual elements of the domain and variables that range over sets of individual elements of the domain. The former are

denoted with lowercase letters such as x, y, z and the latter with uppercase letters X, Y, Z. We denote these two countable sets of variables with V_x and V_X respectively. While MSO uses both kinds of variables, FO logic only uses V_x. Therefore FO logic is literally those formulas of MSO logic *without* quantification over sets of individual domain elements.

If $\rho(R) = 1$, and x stands in the R relation in some domain, we write $R(x)$. Similarly, if $\rho(R) = 2$, and x_1 stands in the R relation to x_2 in some domain, we write $R(x_1, x_2)$. Generally, if $\rho(R) = n$, and the elements $x_1, x_2, \ldots x_n$ stand in the R relation in some domain, we write $R(x_1, x_2, \ldots x_n)$. When $\rho(R)$ is not explicit, we use $\vec{x}$ to mean a tuple of $\rho(R)$ variables and write $R(\vec{x})$ to mean R holds for the tuple of elements in $\vec{x}$. In the notation $R(\vec{x})$, it is understood that $\rho(R) = |\vec{x}|$.

6.2.1 Syntax of MSO logic

This sections defines the syntax of MSO logic.

Definition 1 (Formulas of MSO logic)**.** Fix a signature $\mathfrak{R}$. The formulas of MSO($\mathfrak{R}$) are defined inductively as follows.

The base cases.

For all variables $x, y \in V_x = \{x_0, x_1, \ldots\}$, $X \in V_X = \{X_0, X_1, \ldots\}$, the following are formulas of MSO logic.

(B1)	$x = y$	(**equality**)
(B2)	$x \in X$	(**membership**)
(B3)	$R(\vec{x})$ for each $R \in \mathfrak{R}$	(**atomic relational formulas**)

The inductive cases.

If φ, ψ are formulas of MSO logic, then so are

(I1) $(\neg\varphi)$ **(negation)**
(I2) $(\varphi \vee \psi)$ **(disjunction)**
(I3) $(\varphi \wedge \psi)$ **(conjunction)**
(I4) $(\varphi \rightarrow \psi)$ **(implication)**
(I5) $(\varphi \leftrightarrow \psi)$ **(biconditional)**
(I6) $(\exists x)[\varphi]$ **(existential quantification for individuals)**
(I7) $(\exists X)[\varphi]$ **(existential quantification for sets of individuals)**
(I8) $(\forall x)[\varphi]$ **(universal quantification for individuals)**
(I9) $(\forall X)[\varphi]$ **(universal quantification for sets of individuals)**

Nothing else is a formula of MSO logic.

It is possible to define a MSO logic with some subset of the above inductive cases (for example negation, disjunction, and existential quantification) and to derive the remainder. The above definition attempts to strike a balance between austere minimality and some utility.

6.2.2 Semantics of MSO logic

The **free** variables of a formula φ are those variables in φ that are not quantified. A formula is a **sentence** if none of its variables are free. Only sentences can be interpreted.

If a $\mathfrak{R}$-structure $\mathcal{M}$ **satisfies**, or **models**, a sentence $\varphi \in \text{MSO}(\mathfrak{R})$, one writes $\mathcal{M} \models \varphi$. If Ω is a class of objects (like Σ^*) and $\mathfrak{R}$ is a signature for representing elements of Ω then the **extension** of φ is denoted $[\![\varphi]\!]$ and equals $\{\omega \in \Omega \mid \mathcal{M}_\omega \models \varphi\}$.

It will also be useful to think of the interpretation of φ as a function that maps relational structures to the set $\{\texttt{true}, \texttt{false}\}$. Since $[\![\varphi]\!]$ denotes a set, this function is essentially that set's **indicator function**. Instead of introducing new notation for this indicator function, I will reuse the $[\![\varphi]\!]$ notation. So while $[\![\varphi]\!]$ designates a set, $[\![\varphi]\!](\mathcal{M})$ denotes a function which takes an $\mathfrak{R}$-structure $\mathcal{M}$ and returns a truth value. Whether $[\![\varphi]\!]$ is being interpreted as a set or as a function should be clear from context.

In order to evaluate $[\![\varphi]\!](\mathcal{M})$—that is, in order to decide whether $\mathcal{M} \models \varphi$—variables must be assigned values. For this reason, the function $[\![\varphi]\!]$ actually takes two arguments: one is the $\mathfrak{R}$-structure $\mathcal{M}$ and one is the *assignment function*. The assignment function $\mathbb{S}$ maps individual variables (like x) to individual elements of domain D and maps set-of-individual

variables (like X) to sets of individuals (so subsets of D). Formally, $\mathbb{S} : (V_x \to D) \cup (V_X \to \wp(D))$. The assignment function $\mathbb{S}$ may be partial, even empty. The empty assignment is denoted $\mathbb{S}_0$.

We evaluate $[\![\varphi]\!](\mathcal{M}, \mathbb{S}_0)$. Throughout the evaluation, the assignment function $\mathbb{S}$ gets updated. The notation $\mathbb{S}[x \mapsto e]$ updates the assignment function to bind element e to variable x. Similarly, the notation $\mathbb{S}[X \mapsto S]$ updates the assignment function to bind the set of elements S to variable X. Then whether $\mathcal{M} \models \varphi$ can be determined inductively by the below definition.

Definition 2 (Interpreting sentences of MSO logic)**.** Fix a signature $\mathfrak{R}$.

The base cases.

(B1)	$[\![x = y]\!](\mathcal{M}, \mathbb{S})$	$\leftrightarrow$	$\mathbb{S}(x) = \mathbb{S}(y)$
(B2)	$[\![x \in X]\!](\mathcal{M}, \mathbb{S})$	$\leftrightarrow$	$\mathbb{S}(x) \in \mathbb{S}(X)$
(B3)	For each $R \in \mathfrak{R}$, $[\![R(\vec{x})]\!](\mathcal{M}, \mathbb{S})$	$\leftrightarrow$	$\mathbb{S}(\vec{x}) \in R$

To clarify the notation in (B3): if $\vec{x} = (x_1, x_2, \ldots x_n)$ then $\mathbb{S}(\vec{x}) = (\mathbb{S}(x_1), \mathbb{S}(x_2), \ldots \mathbb{S}(x_n))$.

The inductive cases.

(I1)	$[\![(\neg\varphi)]\!](\mathcal{M}, \mathbb{S})$	$\leftrightarrow$	$\neg[\![\varphi]\!](\mathcal{M}, \mathbb{S})$
(I2)	$[\![(\varphi \vee \psi)]\!](\mathcal{M}, \mathbb{S})$	$\leftrightarrow$	$[\![\varphi]\!](\mathcal{M}, \mathbb{S}) \vee [\![\psi]\!](\mathcal{M}, \mathbb{S})$
(I3)	$[\![(\varphi \wedge \psi)]\!](\mathcal{M}, \mathbb{S})$	$\leftrightarrow$	$[\![\varphi]\!](\mathcal{M}, \mathbb{S}) \wedge [\![\psi]\!](\mathcal{M}, \mathbb{S})$
(I4)	$[\![(\varphi \to \psi)]\!](\mathcal{M}, \mathbb{S})$	$\leftrightarrow$	$[\![\varphi]\!](\mathcal{M}, \mathbb{S}) \to [\![\psi]\!](\mathcal{M}, \mathbb{S})$
(I5)	$[\![(\varphi \leftrightarrow \psi)]\!](\mathcal{M}, \mathbb{S})$	$\leftrightarrow$	$[\![\varphi]\!](\mathcal{M}, \mathbb{S}) \leftrightarrow [\![\psi]\!](\mathcal{M}, \mathbb{S})$
(I6)	$[\![(\exists x)[\varphi]]\!](\mathcal{M}, \mathbb{S})$	$\leftrightarrow$	$(\bigvee_{e \in D} [\![\varphi]\!](\mathcal{M}, \mathbb{S}[x \mapsto e])$
(I7)	$[\![(\exists X)[\varphi]]\!](\mathcal{M}, \mathbb{S})$	$\leftrightarrow$	$(\bigvee_{S \subseteq D} [\![\varphi]\!](\mathcal{M}, \mathbb{S}[X \mapsto S])$
(I8)	$[\![(\forall x)[\varphi]]\!](\mathcal{M}, \mathbb{S})$	$\leftrightarrow$	$(\bigwedge_{e \in D} [\![\varphi]\!](\mathcal{M}, \mathbb{S}[x \mapsto e])$
(I9)	$[\![(\forall X)[\varphi]]\!](\mathcal{M}, \mathbb{S})$	$\leftrightarrow$	$(\bigwedge_{S \subseteq D} [\![\varphi]\!](\mathcal{M}, \mathbb{S}[X \mapsto S])$

6.3 FO Logic

FO($\mathfrak{R}$) is defined as all the formulas of MSO($\mathfrak{R}$) logic which include no quantification over sets of individuals. In other words, there are no sentences which include variables from V_X and so cases B2, I7, and I9 never occur. In all other respects, sentences of FO logic are interpreted the same way as above.

6.4 Courcellian Logical Transformations

Next we define transductions from $\mathfrak{R}_A$-structures to $\mathfrak{R}_B$-structures.

A deterministic MSO-definable transduction τ from $\mathfrak{R}_A$-structures to $\mathfrak{R}_B$-structures is specified by the following formulas.

1. a **domain formula** $\varphi_d \in$ MSO($\mathfrak{R}_A$) with no free variables;
2. a nonempty **copy set** $C \subset \mathbb{N}$ of finite cardinality;
3. for each $c \in C$, a **licensing formula** $\varphi^c_\ell(x) \in$ MSO($\mathfrak{R}_A$) with one free variable; and
4. for each $R_B \in \mathfrak{R}_B$ with $\rho(R_B) = n$ and $\vec{c} \in C^n$, there is a **relational formula** $\varphi^{\vec{c}}_{R_B}(\vec{x}) \in$ MSO($\mathfrak{R}$) with n free variables (note $|\vec{c}| = |\vec{x}| = n$).

It follows that defining τ requires the following formulas to be defined.

- one domain formula
- $|C|$ licensing formulas
- $\sum_{R_B \in \mathfrak{R}_B} |C|^{\rho(R_B)}$ relational formulas. To explain why, observe that $|C|$ relational formulas will need to be defined for each unary relation in $\mathfrak{R}_B$; $|C|^2$ relational formulas will need to be defined for each binary relation in $\mathfrak{R}_B$; and generally $|C|^n$ relational formulas will need to be defined for each n-ary relation in $\mathfrak{R}_B$.

Next we define how the above formulas provide a $\mathfrak{R}_B$-structures from a given $\mathfrak{R}_A$-structure $\mathcal{M} = \langle D_A \mid (R)_{R \in \mathfrak{R}_A} \rangle$.

1. If $\mathcal{M} \models \varphi_d$ then $\tau($$\mathcal{M}$ is defined. Otherwise $\tau(\mathcal{M})$ is undefined.

2. If $\tau(\mathcal{M})$ is defined then it equals the $\mathfrak{R}_B$-structure $\langle D_B \mid (R)_{R \in \mathfrak{R}_B} \rangle$ where

   - $D_B = \{(e, c) \mid e \in D_A, c \in C, \mathcal{M} \models \varphi^c_\ell(x)\}$
   - For each $R \in \mathfrak{R}_B$ with $\rho(R) = n$,
     and for each $\langle (x_1, c_1), \ldots, (x_n, c_n) \rangle \in (D_B)^n$,
     it is the case that $\langle (x_1, c_1), \ldots, (x_n, c_n) \rangle \in R_B$ iff $\mathcal{M} \models \varphi^{\vec{c}}_{R_B}(\vec{x})$
     where $\vec{c} = \langle c_1, \ldots, c_n \rangle$ and $\vec{x} = \langle x_1, \ldots, x_n \rangle$.

Consequently, the following conclusions stand.

- For any element element e in D_A of the $\mathfrak{R}_A$-structure and for any element $c \in C$, the pair (e, c) exists in the domain of the $\mathfrak{R}_B$-structure $\tau(\mathcal{M})$ if and only if $\varphi^c_\ell(e)$ is true.
- For any unary relation $R \in \mathfrak{R}_B$, $e \in D_A$, and $c \in C$, $R(e, c)$ holds if and only if $\mathcal{M} \models \varphi^c_R(e)$ and $(e, c) \in D_B$.
- For any binary relation $R \in \mathfrak{R}_B$, $e_1, e_2 \in D_A$, and $c_1, c_2 \in C$, $R((e_1, c_1), (e_2, c_2))$ holds if and only if $\mathcal{M} \models \varphi^{c_1,c_2}_{R^\diamond}(e_1, e_2)$ and $(e_1, c_1), (e_2, c_2) \in D_B$.

6.5 Weighted Monadic Second Order Logic

This section formalizes the concepts that were introduced in Chapter 4.

6.5.1 Semirings

We have seen how we can use logic to describe functions $f : \Sigma^* \to \{\texttt{true}, \texttt{false}\}$. Weighted logics allow one to describe functions with different co-domains, including $\mathbb{N}, [0, 1], \Delta^*$ and so on. Crucially, the co-domain is a mathematical object known as a **semiring**. We basically follow the presentation by Droste and Gastin (2009).[1]

[1] An important difference is I have kept equality, which they omit. One reason to omit equality is that it may not be decidable for an arbitrary semiring whether two of its elements are equal. For example, in the real interval, most real numbers are not even computable. Nevertheless, equality is assumed here.

A semiring is a set S with two binary operations $\oplus, \otimes$, called 'addition/plus' and 'multiplication/times', and with elements 1 and 0 with the following properties satisfied for all $x, y, z \in S$:

(P1)	$x \oplus y, x \otimes y \in S$	(closure under $\oplus$ and $\otimes$)
(P2)	$x \oplus y = y \oplus x$	($\oplus$ is commutative)
(P3)	$0 \oplus x = x \oplus 0 = x$	(0 is the identity for $\oplus$)
(P4)	$1 \otimes x = x \otimes 1 = x$	(1 is the identity for $\otimes$)
(P5)	$0 \otimes x = x \otimes 0 = 0$	(0 is an annihilator for $\otimes$)
(P6)	$x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$	($\otimes$ right distributes over $\oplus$)

Below are some examples of semirings.

Name	S	$\oplus$	$\otimes$	0	1
Boolean	$\{\texttt{true}, \texttt{false}\}$	$\vee$	$\wedge$	`false`	`true`
Natural	$\mathbb{N}$	$+$	$\times$	0	1
Viterbi	$[0, 1]$	`max`	$\times$	0	1
Language	$\wp(\Sigma^*)$	$\cup$	$\cdot$	$\emptyset$	$\{\lambda\}$

Previously we could understand existential quantification as disjunction over the elements in the domain whereas universal quantification is a conjunction of the elements in the domain. With WMSO, existential quantification combines the elements of the domain with $\oplus$ whereas universal quantification combines them with $\otimes$.

6.5.2 Syntax of Weighted MSO Logic

Definition 3 (Formulas of WMSO logic)**.** Fix a signature $\mathfrak{R}$ and a semiring S. The formulas of WMSO($S, \mathfrak{R}$) are defined inductively as follows.

The base cases.

For all variables $x, y \in \{x_0, x_1, \ldots\}$, $X \in \{X_0, X_1, \ldots\}$, and for all $R \in \mathbb{M}$ the following are formulas of MSO logic.

(B1)	s, for each $s \in S$	**(atomic semiring element)**
(B2)	$x = y$	**(equality)**
(B3)	$x \neq y$	**(non-equality)**
(B4)	$x \in X$	**(membership)**
(B5)	$x \notin X$	**(non-membership)**
(B6)	$R(\vec{x})$, for each $R \in \mathbb{M}$	**(positive relational atom)**
(B7)	$\neg R(\vec{x})$, for each $R \in \mathbb{M}$	**(negative relational atom)**

As before, it is understood that the $|\vec{x}| = \rho(R)$. So if R is a unary relation, then $\vec{x} = (x)$. If R is a binary relation, then $\vec{x} = (x, y)$, and so on.

The inductive cases.

If φ, ψ are formulas of MSO logic, then so are

(I1)	$(\varphi \vee \psi)$	**(disjunction)**
(I2)	$(\varphi \wedge \psi)$	**(conjunction)**
(I3)	$(\exists x)[\varphi]$	**(existential quantification for individuals)**
(I4)	$(\exists X)[\varphi]$	**(existential quantification for sets of individuals)**
(I5)	$(\forall x)[\varphi]$	**(universal quantification for individuals)**
(I6)	$(\forall X)[\varphi]$	**(universal quantification for sets of individuals)**

Nothing else is a formula of weighted MSO logic. Note that negation is only present in the base cases.

6.5.3 Semantics of Weighted MSO Logic

Let Ω be a class of objects (like Σ^*) and let S be a semiring. Let $\mathfrak{R}$ denote a signature for representing elements of Ω. Let φ be a sentence of WMSO($S, \mathfrak{R}$). Then $[\![\varphi]\!]$ denotes a function with domain Ω and co-domain S. Formally, $[\![\varphi]\!] : \Omega \to S$.

As before, interpreting φ requires an assignment function $\mathbb{S}$. We write $[\![\varphi]\!](\mathcal{M}, \mathbb{S})$ to express the value in S that φ assigns to $\mathcal{M}$.

Definition 4 (Interpreting formulas of WMSO logic)**.** Fix a signature $\mathfrak{R}$ and semiring S. Let D be the domain of the input $\mathfrak{R}$-structure $\mathcal{M}$.

The base cases.

(B1)	$[\![s]\!](\mathcal{M},\mathbb{S})$	$\stackrel{\text{def}}{=}$	s
(B2)	$[\![(x=y)]\!](\mathcal{M},\mathbb{S})$	$\stackrel{\text{def}}{=}$	1 iff $\mathbb{S}(x)=\mathbb{S}(y)$, 0 otherwise
(B3)	$[\![x\neq y]\!](\mathcal{M},\mathbb{S})$	$\stackrel{\text{def}}{=}$	0 iff $\mathbb{S}(x)=\mathbb{S}(y)$, 1 otherwise
(B4)	$[\![x\in X]\!](\mathcal{M},\mathbb{S})$	$\stackrel{\text{def}}{=}$	1 iff $\mathbb{S}(x)\in\mathbb{S}(X)$, 0 otherwise
(B5)	$[\![x\notin X]\!](\mathcal{M},\mathbb{S})$	$\stackrel{\text{def}}{=}$	0 iff $\mathbb{S}(x)\in\mathbb{S}(X)$, 1 otherwise
(B6)	$[\![R(\vec{x})]\!](\mathcal{M},\mathbb{S})$	$\stackrel{\text{def}}{=}$	1 iff $R(\mathbb{S}(\vec{x}))$, 0 otherwise
(B7)	$[\![\neg R(\vec{x})]\!](\mathcal{M},\mathbb{S})$	$\stackrel{\text{def}}{=}$	0 iff $R(\mathbb{S}(\vec{x}))$, 1 otherwise

The inductive cases.

(I1)	$[\![(\varphi\vee\psi)]\!](\mathcal{M},\mathbb{S})$	$\stackrel{\text{def}}{=}$	$[\![\varphi]\!](\mathcal{M},\mathbb{S})\oplus[\![\varphi]\!](\mathcal{M},\mathbb{S})$
(I2)	$[\![(\varphi\wedge\psi)]\!](\mathcal{M},\mathbb{S})$	$\stackrel{\text{def}}{=}$	$[\![\varphi]\!](\mathcal{M},\mathbb{S})\otimes[\![\varphi]\!](\mathcal{M},\mathbb{S})$
(I3)	$[\![(\exists x)[\varphi]]\!](\mathcal{M},\mathbb{S})$	$\stackrel{\text{def}}{=}$	$\bigoplus_{e\in D}[\![\varphi]\!](\mathbb{S}[x\mapsto e],w)$
(I4)	$[\![(\exists X)[\varphi]]\!](\mathcal{M},\mathbb{S})$	$\stackrel{\text{def}}{=}$	$\bigoplus_{E\in D}[\![\varphi]\!](\mathbb{S}[X\mapsto E],w)$
(I5)	$[\![(\forall x)[\varphi]]\!](\mathcal{M},\mathbb{S})$	$\stackrel{\text{def}}{=}$	$\bigotimes_{e\in D}[\![\varphi]\!](\mathbb{S}[x\mapsto e],w)$
(I6)	$[\![(\forall X)[\varphi]]\!](\mathcal{M},\mathbb{S})$	$\stackrel{\text{def}}{=}$	$\bigotimes_{E\in D}[\![\varphi]\!](\mathbb{S}[X\mapsto E],w)$

Since multiplication is not necessarily commutative, the order in which it occurs matters. When there is universal quantification over individuals ($\forall x$), the multiplication is done according to the natural order. This means that if the elements of D are natural numbers then they are multiplied according to the order of natural numbers.

When there is universal quantification over sets of individuals, an order over the subsets of the domain must be assumed. One way to order finite subsets of natural numbers is to order them according to the length-lexicographic order of their list-representations. A list-representation of a finite subset of natural numbers is just the list of numbers in ascending order.

Finally, since addition is necessarily commutative (unlike multiplication), we do not worry about the order of the computation for existential quantification.

6.6 Propositional Logic

This section defines a logical language using propositional logic and $\mathfrak{R}$-structures.

We begin with what is meant by **connected relational structure** with a signature $\mathfrak{R}$. For each $\mathfrak{R}$-structure $\mathcal{M}$ with $\mathfrak{R} = \{R_1, \ldots R_n\}$ let the binary relation C be defined as follows.

$$\begin{aligned} C \stackrel{\text{def}}{=} \big\{&(x,y) \in D \times D \mid \\ &\exists i \in \{1 \ldots n\}, R_i \in \mathfrak{R} \\ &\exists k \in \mathbb{N}\ [\rho(R_i) = k], \\ &\exists (x_1 \ldots x_k) \in R_i, \\ &\exists s,t \in \{1 \ldots k\}, x = x_s, y = x_t \big\} \end{aligned}$$

Further, let C^* denote the transitive closure of C. This means C^* is the least set which contains C and for which it is the case that whenever $(x,y) \in C^*$ and $(y,z) \in C^*$ then $(x,z) \in C^*$ too. Then a structure A is **connected** whenever, for all x, y in the domain of A, it holds that $(x,y) \in C^*$.

As an example, consider the structure M_{abbcc} in the conventional successor model. This is a connected structure because the the successor relation chains together any two elements. For instance, that domain elements 1 and 4 are connected is witnessed by these elements of the successor relation $(1,2),(2,3),(3,4)$. In fact, the structure of every string under every model discussed is connected under this definition.

What is an example of an unconnected structure? Under the signature $\langle \lhd, a, b, c \rangle$, consider the structure $A = \{\{1,2\} \mid \emptyset, \{1\}, \{2\}, \emptyset\}$. This structure contains two elements (one is labeled *a* and one is labeled *b*) but they are not connected by any series of relations.

Next we discuss what it means for one structure to be a **restriction** of another. Let A, B both be $\mathfrak{R}$-structures. A **is a restriction of** structure B if $D_A \subseteq D_B$ and for each m-ary relation R, we have $(x_1 \ldots x_m) \in R_A$ if and only if $(x_1 \ldots x_m) \in R_B$ and $x_1, \ldots, x_m \in D_A$. So A is essentially what is left of B after B is stripped of elements and relations which are not wholly within the domain of A.

Finally, we say structure A **is contained by** B structure if A is isomorphic to a restriction of B. Whenever A is contained by B and A is a connected structure, we also say A is a **factor** of B (denoted $A \sqsubseteq B$).

For each string $w \in \Sigma^*$, let $F(\mathcal{M}_w)$ denote the set of factors of the structure $\mathcal{M}_w$ and let $F_k(\mathcal{M}_w)$ be set of factors whose size is less than or equal to k (recall that the size of a structure is equal to the cardinality of its domain). Formally, $F(w) = \{\mathcal{S} \sqsubseteq M_w\}$ and $F_k(w) = \{\mathcal{S} \sqsubseteq M_w \mid |\mathcal{S}| \leq k\}$. Finally, we lift the definition of F and F_k to sets of strings as follows.

$$F(S) = \bigcup_{w \in S} F(\mathcal{M}_w) \tag{6.1}$$

$$F_k(S) = \bigcup_{w \in S} F_k(\mathcal{M}_w) \tag{6.2}$$

6.6.1 Syntax of Propositional Logic

We can now define sentences of propositional logic as follows.

Definition 5 (Propositional Logic with Literal Factors)**.** Fix a signature $\mathfrak{R}$.

The base case.

(B1) For all factors f in $F(\Sigma^*)$, f is a sentence of PROP($\mathfrak{R}$).

The inductive cases.

If φ, ψ are formulas of PROP($\mathfrak{R}$), then so are

(I1)	$\neg\varphi$	**(negation)**
(I2)	$(\varphi \vee \psi)$	**(disjunction)**
(I3)	$(\varphi \wedge \psi)$	**(conjunction)**
(I4)	$(\varphi \rightarrow \psi)$	**(implication)**
(I5)	$(\varphi \leftrightarrow \psi)$	**(biconditional)**

Nothing else is a formula of Propositional logic.

As with non-weighted MSO logic, for a sentence φ belonging to PROP($\mathfrak{R}$) and a $\mathfrak{R}$-structure $\mathcal{M}$, we say $\mathcal{M} \models \varphi$ if φ is true of $\mathcal{M}$. If Ω is a class of objects (like Σ^*) and $\mathfrak{R}$ is a signature for representing elements of Ω then the *extension* of φ is denoted $[\![\varphi]\!]$ and equals $\{\omega \in \Omega \mid \mathcal{M}_\omega \models \varphi\}$.

6.6.2 Semantics of Propositional Logic

As with the non-weighted case before, we conceive of $[\![\varphi]\!]$ as an indicator function which maps relational structures to the set $\{\texttt{true}, \texttt{false}\}$.

Definition 6 (Intepreting Sentences of Propositional Logic with Literal Factors)**.** Fix a signature $\mathfrak{R}$.

The base case.

(B1) For all factors f in $F(\Sigma^*)$, $[\![f]\!](\mathcal{M}) \stackrel{\text{def}}{=} f \sqsubseteq \mathcal{M}$.

The inductive cases.

If φ, ψ are formulas of PROP($\mathfrak{R}$), then so are

$$\begin{array}{lrcl}
\text{(I1)} & [\![(\neg\varphi)]\!](\mathcal{M}) & \leftrightarrow & \neg[\![\varphi]\!](\mathcal{M}) \\
\text{(I2)} & [\![(\varphi \vee \psi)]\!](\mathcal{M}) & \leftrightarrow & [\![\varphi]\!](\mathcal{M}) \vee [\![\psi]\!](\mathcal{M}) \\
\text{(I3)} & [\![(\varphi \wedge \psi)]\!](\mathcal{M}) & \leftrightarrow & [\![\varphi]\!](\mathcal{M}) \wedge [\![\psi]\!](\mathcal{M}) \\
\text{(I4)} & [\![(\varphi \rightarrow \psi)]\!](\mathcal{M}) & \leftrightarrow & [\![\varphi]\!](\mathcal{M}) \rightarrow [\![\psi]\!](\mathcal{M}) \\
\text{(I5)} & [\![(\varphi \leftrightarrow \psi)]\!](\mathcal{M}) & \leftrightarrow & [\![\varphi]\!](\mathcal{M}) \leftrightarrow [\![\psi]\!](\mathcal{M})
\end{array}$$

Part II

Case Studies

Chapter 7

Regressive voicing assimilation in Russian obstruent clusters

HOSSEP DOLATIAN

7.1 Introduction

Russian is known for having the common phonological process of voicing assimilation in obstruent clusters. In brief, obstruents clusters in lexical items can have heterogeneous voicing in the underlying representation (UR): /at valnʲi/ 'from wave.' But in the surface representation (SR), the clusters agree in voice: [ad valnʲi]. This chapter illustrate how this local phonological process can be described and formalized with the logical methods in Part I of this book.

7.2 General description on the data

Table 7.1 lists prepositional phrases made up of a preposition and a single-word object. We look at three prepositions: obstruent final /at, bʲiz/, and vowel-final /u/. Data and transcriptions are taken from Halle and Clements (1983, 109).

First Segment of X	'from X'	'without X'	'next to X'	Gloss for X
voiced sonorant	at rózi͡ts	bʲiz rózi	u rózi	'rose'
	at mʲɛni	bʲiz mʲɛni	u mʲɛni	'change'
	at luni	bʲiz luni	u luni	'moon'
	at áli	bʲiz áli	u áli	'Ala' (name)
	at iri	bʲiz iri	u iri	'Ira' (name)
voiced obstruent	ad valni	bʲiz valni	u valni	'wave'
	ad baradi	bʲiz baradí	u baradí	'beard'
	ad galavi	bʲiz galavi	u galavi	'head'
voiceless obstruent	at pʲiti	bʲis pʲiti	u pʲiti	'heel'
	at sʲistrí	bʲis sʲistrí	u sʲistrí	'sister'
	at karovi	bʲis karóvi	u karóvi	'cow'

Table 7.1: Data set for voicing assimilation of Russian obstruent clusters

The vowel-final preposition 'next to' has a single surface form [u]. It is reasonable to propose a faithful UR /u/. In contrast, the obstruent-final prepositions vary in their pronunciation. The preposition 'from' is [ad] before voiced obstruents, [at] elsewhere. The preposition 'without' is [bʲis] before voiceless obstruents, and [bʲiz] elsewhere. This variation is best explained by positing the URs /at, bʲiz/ and a phonological process of regressive voicing assimilation.

Generalization 7.1. Voicing assimilation in obstruents. Obstruent become voiceless before voiceless obstruents. Obstruents becomes voiced before voiced obstruents.

Table 7.2 illustrates how this generalization correctly accounts for the systematic variation in the pronunciations of the prepositions in Table 7.1.

	'from wave'	'from heel'	'without wave'	'without heel'
Input UR	/at valni/	/at pʲiti/	/bʲiz valni/	/bʲiz pʲiti/
Assimilation	ad valni			bʲis pʲiti
Output SR	[ad valni]	[at pʲiti]	[bʲiz valni]	[bʲis pʲiti]

Table 7.2: Application of regressive voicing assimilation

This generalization can be expressed with an SPE-style rule using so-called alpha notation: [-son] $\longrightarrow$ [αvoice] / $\_\_$ [-son, αvoice] . In Optimality Theory, the markedness constraint *[-son,αvoice][-son,$-\alpha$voice] would outrank the constraint IDENT(VOICE). Additional constraints would be required to make sure that the second obstruent in the sequence stays faithful. In the remainder of this chapter, I formalize this generalization with model theory and logical transductions.

7.3 Logical formalization of assimilation

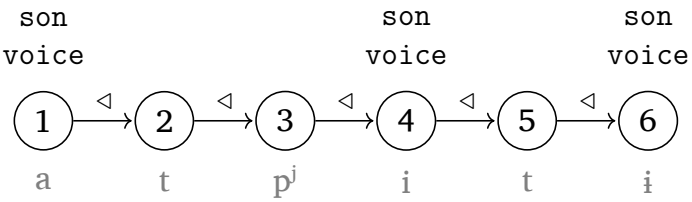
This section provides an analysis using First Order logic and model-theoretic representations of regressive voicing assimilation in Russian obstruent clusters. It is important to be clear both about the representations and the transformations.

7.3.1 Representations

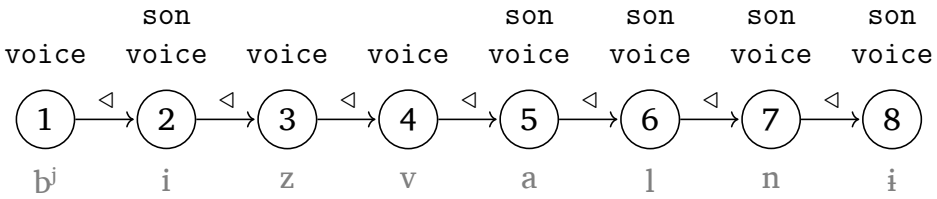
Voicing assimilation has two key properties: a) it targets the features [$\pm$voice] and [$-$son], b) it references adjacent segments. Therefore the input and output word models must be able to reference these properties.

For both the underlying representations and surface representations, I essentially adopt a successor word model with features (see 2.5). There are several different feature systems that make the appropriate featural distinctions. Here I adopt a simple system. In particular, I assume that features are privative, not binary. Thus, the word model uses the unary relations $\texttt{voice}(x)$ and $\texttt{son}(x)$. A segment x is voiced whenever $\texttt{voice}(x)$ evaluates to `true`. A segment x is voiceless whenever $\texttt{voice}(x)$ evaluates to `false`, meaning that $\neg\texttt{voice}(x)$ is `true`. Similarly, a segment x is an obstruent whenever $\neg\texttt{son}(x)$ is `true`. Besides these features, the word model must also have other features to distinguish all possible segments in Russian, but they don't play role in assimilation. This set of privative features, which includes `voice` and `son`, is denoted $\mathcal{F}$. To the order segments, the word model uses the binary successor relation (`succ`). The above information is summarized in the model's signature below.

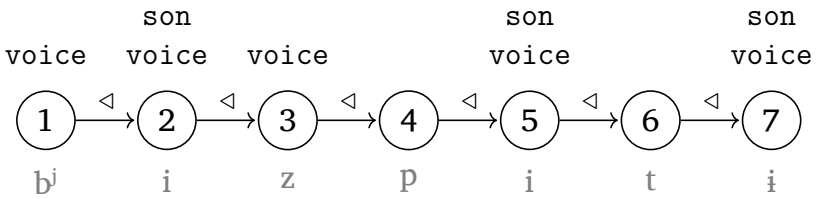
$$\mathfrak{R} = \{\triangleleft\} \cup \mathcal{F}$$



(a) UR /at pʲitɨ/ ‘from heel’



(b) UR /bʲiz valnɨ/ ‘without wave’



(c) UR /bʲiz pʲitɨ/ ‘without heel’

Figure 7.1: Word models of three underlying representations

Figure 7.1 presents word models for the examples /bʲiz valnɨ/ ‘from heel,’ /bʲiz pʲitɨ/ ‘without wave,’ /bʲiz pʲitɨ/ ‘without heel.’ Following convention, the domain elements are non-negative integers. Each segment is represented by privative phonological features and only the features `voice` and `son` are shown in Figure 7.1. To facilitate interpretability, segments are shown in gray below their corresponding domain elements. Adjacent pairs of segments stand in the successor relationship, represented by arrows labeled with $\triangleleft$.

7.3.2 A Logical Transduction for Voice Assimilation

As explained in Chapter 3, defining logical transformation requires defining the copy set, a domain formula, a licensing formula, and formulas for the relations in the model signature of the surface representation. All of these formulas are to defined with a logical language in terms of the relations of the model signature of the underlying representations. This analysis uses First Order logic with the $\mathfrak{R}$-structures defined in the previous sections.

The domain formula is defined to be `true` so that transformation applies to every input. Since voicing assimilation does not change the number of segments between the input and output, a copy set C of cardinality 1 suffices, and the licensing formula is defined to be `true`.

The key parts of the transformation lie defining the unary relations for the output structure. The only change that takes place regards the feature `voice`. All features $\texttt{f} \in \mathcal{F}$ except for this one are faithful in the surface form. Nor is there any repositioning of the segments. If a segment x precedes a segment y in the input, then it also does so in the output. These faithful aspects of the transformation are captured with the equations below.

$$\phi_{\texttt{f}}(x) \stackrel{\text{def}}{=} \quad \texttt{f}(x) \text{ where } \texttt{f} \in \mathcal{F}, \texttt{f} \neq \texttt{voice} \tag{7.1}$$

$$\phi_{\texttt{succ}}(x,y) \stackrel{\text{def}}{=} \quad \texttt{succ}(x,y) \tag{7.2}$$

Note that the first equation is a template for all features other than voice. For example, that equation means that $\phi_{\texttt{son}}(x) = \texttt{son}(x)$ and similarly for the other features in $\mathcal{F}$.

Regarding voicing, a sound x change its voicing quality based on whether a) it is an obstruent, and b) it precedes another obstruent. To make our formalization easier to follow, we define the following predicates that capture whether a segment x is a voiceless or voiced, and whether x precedes a voiceless obstruent or voiced obstruent y.

$$\texttt{vd\_obst}\,(x) \stackrel{\text{def}}{=} \quad \neg\texttt{son}(x) \wedge \texttt{voice}(x) \tag{7.3}$$

$$\texttt{vless\_obst}\,(x) \stackrel{\text{def}}{=} \quad \neg\texttt{son}(x) \wedge \neg\texttt{voice}(x) \tag{7.4}$$

$$\texttt{pre\_vd\_obst}\,(x) \stackrel{\text{def}}{=} \quad \exists y[\,\texttt{vd\_obst}\,(y) \wedge x \lhd y] \tag{7.5}$$

$$\texttt{pre\_vless\_obst}\,(x) \stackrel{\text{def}}{=} \quad \exists y[\,\texttt{vless\_obst}\,(y) \wedge x \lhd y] \tag{7.6}$$

For any given segment x, it will surface as [+voice] if and only if any of the following conditions are satisfied:

(a) it is voiced sonorant underlyingly,

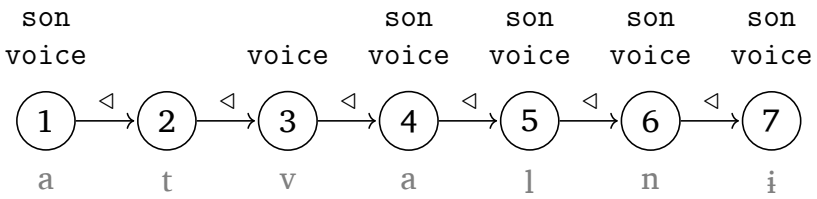
(b) it is a voiced obstruent that does not precede a voiceless obstruent, or

(c) it is a voiceless obstruent that precedes a voiced obstruent.

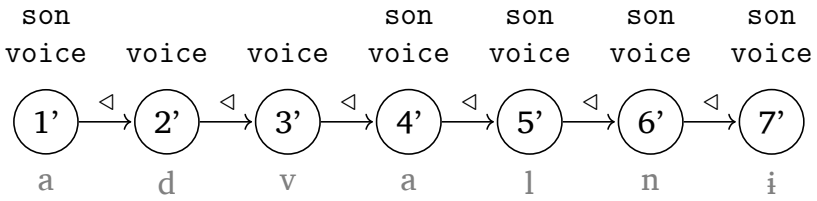
These three conditions are captured via disjunction in the output formula below.

$$\begin{aligned}\phi_{\texttt{voice}}(x) \stackrel{\text{def}}{=} & [\texttt{voice}(x) \wedge \texttt{son}(x)] \\ & \vee [\ \texttt{vd\_obst}\ (x) \wedge \neg\ \texttt{pre\_vless\_obst}\ (x)] \\ & \vee [\ \texttt{vless\_obst}\ (x) \wedge\ \texttt{pre\_vd\_obst}\ (x)]\end{aligned} \tag{7.7}$$

We illustrate the application of this formula with the UR /at valnɨ/ 'from wave' and the SR [ad valnɨ]. We show the word models of the UR and SR in Figure 7.2. To facilitate discussion, output domain elements are marked with an apostrophe.



(a) Input UR /at valnɨ/



(b) Output SR /ad valnɨ/

Figure 7.2: Input UR and output SR as word models 'from wave'

The relevant segment is the /t/ at $x = 2$. It surfaces as voiced [d] at index 2'. It is underlyingly voiceless, meaning $\texttt{voice}(2)$ is `false`. Thus the first two disjuncts in $\phi_{\texttt{voice}}(2)$ are `false`. However, this segment is a voiceless obstruent that precedes a voiced obstruent. Thus the third disjunct in $\phi_{\texttt{voice}}(2)$ is `true`. Thus this segment gains the feature [+voice] in the output. Table 7.3 shows the truth value of each relevant input formula and predicate.

	x						
Formulas	1	2	3	4	5	6	7
$\texttt{voice}(\texttt{x})$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$
$\texttt{son}(x)$	$\top$	$\bot$	$\bot$	$\top$	$\top$	$\top$	$\top$
$\texttt{vd\_obst}\,(x)$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$
$\texttt{vless\_obst}\,(x)$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\texttt{pre\_vd\_obst}\,(x)$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\texttt{pre\_vless\_obst}\,(x)$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\phi_{\texttt{voice}}(x)$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
$\texttt{voice}(x) \wedge \texttt{son}(x)$	$\top$	$\bot$	$\bot$	$\top$	$\top$	$\top$	$\top$
$\texttt{vd\_obst}\,(x) \wedge \neg\, \texttt{pre\_vless\_obst}\,(x)$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$
$\texttt{vless\_obst}\,(x) \wedge\, \texttt{pre\_vd\_obst}\,(x)$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

Table 7.3: Truth values for the transformation of /at valnɨ/ to [ad valnɨ] 'from heel.'

Note how the segment /v/ at index 3 surfaces as voiced [v] at index 3' because it is a voiced obstruent and does not precede a voiceless obstruent (condition (c) above). Thus this segment satisfies the second disjunct of our voicing formula.

Interested readers are encouraged to see how these formulas derive the correct output structures for the other words in Figure 7.1.

7.4 Discussion

The data considered does not show how Russian treats a cluster of more than two obstruents, e.g. /atbta/. There are Russian words with triple obstruent clusters such as [fspyat] 'back' and [vzdrógnut] 'to shutter.' These obstruent clusters, and all others that I am aware of, agree in voicing.

This fact has been used as evidence for a phonotactic constraint like *[-son,αvoice][-son,$-\alpha$voice] used in an OT analysis. In terms of First Order logic and the $\mathfrak{R}$-structures employed here, this constraint can be expressed with the formula below.

$$\texttt{Agree} \stackrel{\text{def}}{=} \neg\exists x, y\Big[\big(\neg\texttt{son}(x) \wedge \neg\texttt{son}(y) \wedge x \lhd y\big) \rightarrow \big(\texttt{voice}(x) \wedge \texttt{voice}(y)\big) \vee \big(\neg\texttt{voice}(x) \wedge \neg\texttt{voice}(y)\big)\Big] \tag{7.8}$$

In words, this formula says if there are two adjacent obstruents then they are either both voiced or both voiceless. There is no evidence that I am aware of that this constraint is violated in any surface forms in Russian.

It may be seem surprising then, that the logical transduction in the previous section maps hypothetical inputs like /atbta/ to surface forms which violate this constraint, such as [adpta]. Readers should verify that the logical transduction presented will map the $\mathfrak{R}$-structure for /atbta/ to the $\mathfrak{R}$-structure for [adpta]. One may wonder whether how this prediction compares to one given by Optimality Theory where a sufficiently highly ranked AGREE constraint ensures such clusters are not derived from underlying forms. The idea that grammars should map any hypothetical underlying forms to well-formed surface forms has been dubbed 'Richness of the Base' (see discussion by Kager (1999)), and has been proposed as a desiderata of phonological theories.

This line of reasoning, however, is not without its flaws. One problem is there is no evidence how Russian speakers would actually pronounce underlying /atpkza/ or /atbgza/ because such URs do not exist, a fact which could be accounted for with a theory that allows constraints like AGREE to apply to the underlying forms of lexical items. Also, it is well known that ranking AGREE over IDENT gives rise to majority rules effects, which have been argued to be problematic on typological (Baković, 2000), computational (Heinz and Lai, 2013), and psycholinguistic grounds (Finley, 2008). Another analysis, available to Optimality Theory, is to run around

the Majority Rules issue by enforcing *spans* of adjacent obstruents to agree with the rightmost obstruent in the span, which is consistent with the regressive nature of the assimilation in Russian.[1]

This analysis is also available to the logical approach, and could be accomplished using First Order logic with word models inlcuding general precedence ($<$) but not with words models only including successor ($\lhd$).[2] In this analysis, the predicates $\texttt{pre\_vd\_obst}(x)$ and $\texttt{pre\_vless\_obst}(x)$ would have to be redefined to be true whenever x is followed by a sequence of obstruents, of which the rightmost one is a voiceless or voiced obstruent, respectively. In particular, the term $x \lhd y$ in those definitions would be replaced by the term below.

$$x < y \wedge \neg\exists z[\neg\texttt{son}(z) \wedge y \lhd z] \wedge \forall z[x < z < y \wedge \neg\texttt{son}(z)] \tag{7.9}$$

In words, this formula says that position x comes before position y, that the next position after y is not an obstruent, and that all the positions between x and y are obstruents. Since y is an obstruent (given the rest of the definitions of $\texttt{pre\_vd\_obst}(x)$ and $\texttt{pre\_vless\_obst}(x)$), this formula ensure that x is to the immediate left of a span of obstruents of which y is the rightmost one.

7.5 Conclusion

Voicing assimilation is a common phonological process that targets obstruent clusters. This chapter analyzed and formalized an example of regressive voicing assimilation in Russian with FO($\lhd, \mathcal{F}$) where $\mathcal{F}$ is a collection of privative features distinguishing speech segments in Russian.

[1] Another plausible approach is to appeal to different prioritizations of faithfulness to segments based on whether make up roots or affixes. However, since this does not address the questions raised by Richness of the Base it will not be further discussed, except to say that such distinctions can also be treated with modoel theory and logic. See (Dolatian, 2020a) for example.

[2] Technically, this could be accomplished with successor ($\lhd$) provided the size of spans of obstruent clusters is bounded in length. However, this situation is also against the spirit of the rich base.

Chapter 8

Saltation in Polish

AMANDA PAYNE

This chapter provides a logical analysis of the phonological process of velar palatalization in Polish, which is one of the processes in the world's languages that has been identified as saltatory. Hayes and White (2015) define **saltation**, so-named because it is a phonological 'leap', in the following way:

> Let A, B, and C be phonological segments. Suppose that for every feature for which A and C have the same value, B likewise has that value; but that B differs from both A and C. If in some context A alternates with C, but B remains invariant, then the alternation A $\sim$ C is a saltation.

Consequently saltatory alternations are cases when a sound alternates with a sound less similar to it than 'necessary,' as the following example will make clear. The alternation of interest in this chapter comes from the data in Łubowicz (2002), and readers are referred to Rubach (1984) and Gussmann (2007) for additional details.

Polish has several palatalization processes affecting obstruents (Gussmann, 2007). This chapter focuses exclusively on the process whereby velar obstruents become alveolar[1] before front vowels. It affects /k/ and /x/ fairly straightforwardly, as seen in (1) and (2).

[1]Rubach (1984) describes these consonants as post-alveolar but Jassem (2003) and Gussmann (2007) describe them as alveolar. We follow these later descriptions.

1. /krok + it͡ɕ/ → krot͡ʃ + it͡ɕ ‘to step’
2. /strax + it͡ɕ/ → straʃ + it͡ɕ ‘to frighten’

However, the velar /g/, in addition to undergoing palatalization, also undergoes spirantization in the same environment, as seen in (3).

3. /vag + it͡ɕ/ → vaʒ + it͡ɕ (*vad͡ʒ + it͡ɕ) ‘to weigh’

Note also that the alveolar /d͡ʒ/ does exist in the pre-front vowel environment when it is there underlyingly, as in (4).

4. /bri̵d͡ʒ + i̵k + i̵/ → bri̵d͡ʒ + ek ‘bridge (dim)’

Examining this alternation in the light of Hayes and White’s definition makes clear it is an example of saltation. The segments /g/ and /ʒ/ share every manner of articulation except continuancy and timing of the release. (They also differ in place of articulation.) The segment /d͡ʒ/ differs from /ʒ/ only in that articulating the former requires completely blocking the airflow in the oral cavity. The segment /d͡ʒ/ differs from /g/ in place of articulation as well as the timing of the release. Consequently the /g/ → [ʒ] alternation before front vowels constitutes an example of saltation since /d͡ʒ/ remains invariant in that context. This alternation is thus unexpected, in the sense that [d͡ʒ] is featurally and therefore phonetically more similar to /g/ than [ʒ] is. If phonological alternations are minimal repairs to marked structures, a standard hypothesis of Optimality Theory, it is unclear why /g/ maps to [ʒ] before front vowels instead of [d͡ʒ].

8.1 Representations

To model this alternation, we will use word models consisting of sets of features and the binary successor relation ($\lhd$) (see Chapter 2.5). In order to proceed, we must determine which features are relevant for the language and alternation. Like the preceding chapter, this chapter also assumes the features are privative. Consequently a segment x is voiced if $\texttt{voice}(x)$ evaluates to `true` and is voiceless if $\texttt{voice}(x)$ evaluates to `false`.

Since the alternation changes velar obstruents to alveolar ones before front vowels, the features `sonorant`, `dorsal`, `coronal`, `vocalic` and `front` are relevant. The alternation also affects manner features like `continuant`

and del_rel (delayed release) depending in part on the feature voice. Thus all of these features are included in the set $\mathcal{F}$ of unary relations in the signature of the models for both the underlying and surface forms.

Table 8.1 shows the features for the velar and alveolar obstruents in Polish. Finally, the set $\mathcal{F}$ also includes other features relevant to the

segment	place	manner	voicing
k	dorsal		
x	dorsal	continuant	
g	dorsal		voice
t͡ʃ	coronal	del_rel	
ʃ	coronal	continuant	
d͡ʒ	coronal	del_rel	voice
ʒ	coronal	continuant	voice

Table 8.1: Features for the relevant velar and alveolar obstruents

language but not necessarily relevant to this particular alternation. The variable f will be used to represent these other features in $\mathcal{F}$ that have not been made explicit above.

8.2 Logical Transduction

Next I define a logical transduction using First Order logic over the structures given in the previous section. For this, we need the following formulas.

- A domain formula, $\phi_{\texttt{dom}}$. In this case it is simply $\phi_{\texttt{dom}} \stackrel{\text{def}}{=} \texttt{true}$.
- A copy set C which will determine the limit of the size of the surface forms in terms of copies of the underlying forms. Here, since there is no insertion of segments from input to output, I let $C \stackrel{\text{def}}{=} \{1\}$.
- A licensing formula $\phi_{\texttt{license}}(x)$, which determines whether the copy of element x is licensed in the domain of the output structure of the surface form. Here, since there is no deletion of segments from input to output, I let $\phi_{\texttt{license}}(x) \stackrel{\text{def}}{=} \texttt{true}$.

- The binary relation formula $\phi_{\lhd}$, where $\phi_{\lhd}(x,y) \stackrel{\text{def}}{=} x \lhd y$.

- Unary relation formulas for each feature in $\mathcal{F}$. Many of these show no change between the underlying and surface forms, while others will describe the palatalization and spirantization found in the Polish data. The remainder of this section details these formulas.

Velar palatalization does not affect the features `sonorant`, `vocalic`, `front`, and `voice`. It follows that these features are always faithful between input and output structures.

$$\phi_{\texttt{sonorant}}(x) \stackrel{\text{def}}{=} \texttt{sonorant}(x) \tag{8.5}$$

$$\phi_{\texttt{vocalic}}(x) \stackrel{\text{def}}{=} \texttt{vocalic}(x) \tag{8.6}$$

$$\phi_{\texttt{front}}(x) \stackrel{\text{def}}{=} \texttt{front}(x) \tag{8.7}$$

$$\phi_{\texttt{voice}}(x) \stackrel{\text{def}}{=} \texttt{voice}(x) \tag{8.8}$$

The features `dorsal`, `coronal`, `continuant`, and `del_rel` can change depending on their context. All other features f which are in $\mathcal{F}$ are also defined to be faithful to the input as shown below.

$$\phi_{\texttt{f}}(x) \stackrel{\text{def}}{=} \texttt{f}(x) \tag{8.9}$$

Next I turn to the features `dorsal`, `coronal`, `continuant`, and `del_rel`. It will be helpful to identify when a segment x is an obstruent before a front vowel.

$$\texttt{before\_front\_vowel}\,(x) \stackrel{\text{def}}{=} \neg\texttt{sonorant}(x) \wedge \exists y(x \lhd y \wedge \texttt{front}(y) \wedge \texttt{vocalic}(y)) \tag{8.10}$$

Underlying velar obstruents remain velar in the surface form, unless they are before front vowels.

$$\phi_{\texttt{dorsal}}(x) \stackrel{\text{def}}{=} \texttt{dorsal}(x) \wedge \neg\,\texttt{before\_front\_vowel}\,(x) \tag{8.11}$$

Underlying alveolar segments remain alveolar in the surface form, plus

velar obstruents become alveolar before front vowels.[2]

$$\phi_{\texttt{coronal}}(x) \stackrel{\text{def}}{=} \texttt{coronal}(x) \vee (\texttt{dorsal}(x) \wedge \texttt{before\_front\_vowel}\,(x)) \tag{8.12}$$

Underlying continuants remain continuants, plus the voiced velar stop becomes a fricative before front vowels.

$$\phi_{\texttt{continuant}}(x) \stackrel{\text{def}}{=} \texttt{continuant}(x) \vee (\texttt{dorsal}(x) \wedge \texttt{voice}(x) \wedge \texttt{before\_front\_vowel}\,(x)) \tag{8.13}$$

Underlying segments with a delayed release remain delayed release, plus the voiceless velar stop becomes an affricate when before front vowels.

$$\phi_{\texttt{del\_rel}}(x) \stackrel{\text{def}}{=} \texttt{del\_rel}(x) \vee \big(\texttt{dorsal}(x) \wedge \neg\texttt{voice}(x) \wedge \neg\texttt{continuant}(x) \wedge \texttt{before\_front\_vowel}\,(x)\big) \tag{8.14}$$

I illustrate how this transduction works for the the four hypothetical examples below.

5. /gi/ → [ʒi]
6. /ki/ → [t͡ʃi]
7. /xi/ → [ʃi]
8. /d͡ʒi/ → [d͡ʒ]

Consider first the alternation in (5). Here, /gi/ undergoes a becomes [ʒi] as shown in Figure 8.1. The truth values for the relevant formulas are displayed in Table 8.2.

In (6), the underlying representation /ki/ becomes [t͡ʃi] as shown in Figure 8.2. Note that the voiceless velar [k] takes on the feature `del_rel`, as displayed in Table 8.3.

Next we turn to the alternation in (7). Here, /xi/ undergoes a typical palatalization, becoming [ʃi] (Figure 8.3). The evaluation of the logical formulas is shown in Table 8.4.

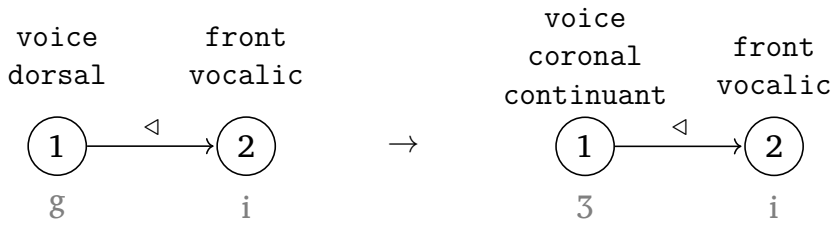


Figure 8.1: Word models for the input /gi/ and output [ʒi].

	x	
Formulas	1	2
$\texttt{voice}(x)$	$\top$	$\top$
$\texttt{sonorant}(x)$	$\bot$	$\top$
$\texttt{continuant}(x)$	$\bot$	$\top$
$\texttt{vocalic}(x)$	$\bot$	$\top$
$\texttt{front}(x)$	$\bot$	$\top$
$\texttt{before\_front\_vowel}\,(x)$	$\top$	$\bot$
$\phi_{\texttt{continuant}}(x)$	$\top$	$\top$
$\phi_{\texttt{dorsal}}(x)$	$\bot$	$\bot$
$\phi_{\texttt{coronal}}(x)$	$\top$	$\bot$
$\phi_{\texttt{del\_rel}}(x)$	$\bot$	$\bot$

Table 8.2: Truth values for the transformation of /gi/ to [ʒi] in Polish.

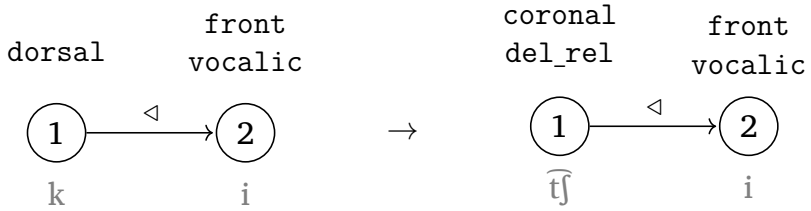


Figure 8.2: Word models for the input /ki/ and output [t͡ʃi].

Finally, consider the hypothetical form in (8), /d͡ʒi/, which surfaces

	x	
Formulas	1	2
$\texttt{voice}(x)$	$\bot$	$\top$
$\texttt{sonorant}(x)$	$\bot$	$\top$
$\texttt{continuant}(x)$	$\bot$	$\top$
$\texttt{vocalic}(x)$	$\bot$	$\top$
$\texttt{front}(x)$	$\bot$	$\top$
$\texttt{before\_front\_vowel}(x)$	$\top$	$\bot$
$\phi_{\texttt{continuant}}(x)$	$\bot$	$\top$
$\phi_{\texttt{dorsal}}(x)$	$\bot$	$\bot$
$\phi_{\texttt{coronal}}(x)$	$\top$	$\bot$
$\phi_{\texttt{del\_rel}}(x)$	$\top$	$\bot$

Table 8.3: Truth values for the transformation of /ki/ to [t͡ʃi].

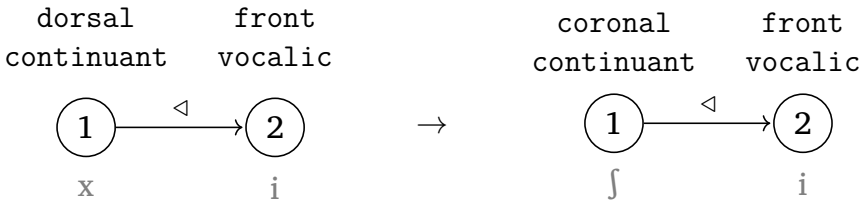


Figure 8.3: Word models for the input /xi/ and output [ʃi].

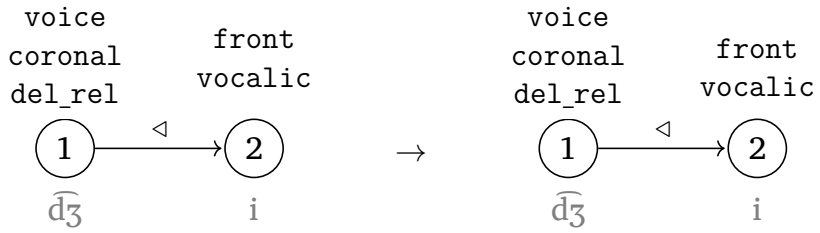


Figure 8.4: Word model for the input /d͡ʒi/ and the output [d͡ʒi].

faithfully as the output [d͡ʒi] (Figures 8.4). The evaluation of the logical

[2]There are palatalization processes affecting alveolar obstruents in the language

	x	
Formulas	1	2
$\texttt{voice}(x)$	$\bot$	$\top$
$\texttt{sonorant}(x)$	$\bot$	$\top$
$\texttt{continuant}(x)$	$\top$	$\top$
$\texttt{vocalic}(x)$	$\bot$	$\top$
$\texttt{front}(x)$	$\bot$	$\top$
$\texttt{before\_front\_vowel}(x)$	$\top$	$\bot$
$\phi_{\texttt{continuant}}(x)$	$\top$	$\top$
$\phi_{\texttt{dorsal}}(x)$	$\bot$	$\bot$
$\phi_{\texttt{coronal}}(x)$	$\top$	$\bot$
$\phi_{\texttt{del\_rel}}(x)$	$\bot$	$\bot$

Table 8.4: Truth values for the transformation of /xi/ to [ʃi].

formulae is shown in Table 8.5.

When full words are considered, the process behaves exactly the same. For instance, consider the saltative alternation shown in (3) where /va**g** + i͡tɕ/ → [va**ʒ** + i͡tɕ].

The structure of the underlying form /vagi͡tɕ/ is shown in Figure 8.5.

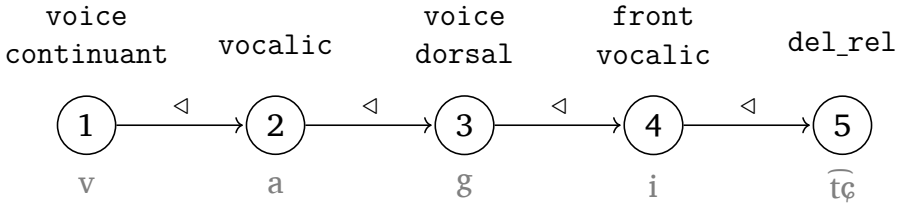


Figure 8.5: Word model for the input /vagi͡tɕ/

If we apply each of the equations (5)–(10) to the input, we get a model for the output form, [vaʒi͡tɕ], as shown in Figure 8.6. Table 8.6 illustrates the evaluation of the formulas. Note the changes which have taken place in segment 3, which changes from a velar to an alveolar fricative.

(Rubach, 1984; Gussmann, 2007), but they are not described in this chapter.

	x	
Formulas	1	2
$\texttt{voice}(x)$	$\top$	$\top$
$\texttt{sonorant}(x)$	$\bot$	$\top$
$\texttt{continuant}(x)$	$\bot$	$\top$
$\texttt{vocalic}(x)$	$\bot$	$\top$
$\texttt{front}(x)$	$\bot$	$\top$
$\texttt{before\_front\_vowel}(x)$	$\top$	$\bot$
$\phi_{\texttt{continuant}}(x)$	$\bot$	$\top$
$\phi_{\texttt{dorsal}}(x)$	$\bot$	$\bot$
$\phi_{\texttt{coronal}}(x)$	$\top$	$\bot$
$\phi_{\texttt{del\_rel}}(x)$	$\top$	$\bot$

Table 8.5: Truth values for the transformation of /d͡ʒi/ to [d͡ʒi].

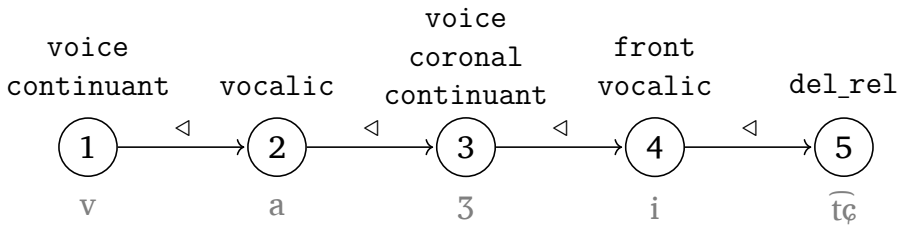


Figure 8.6: Word model for the output [vaʒit͡ɕ]

8.3 Discussion

Although saltation has been considered an unusual (even unexpected) phonological process, it is nevertheless one which exists in multiple human languages. Optimality Theory requires additional machinery to account for saltation, like constraint conjunction. Even so, these analyses of the pattern may overgenerate (Hayes and White, 2015). On the other hand, the logical transduction presented here is quite simple, and similar in form to the models used for non-saltatory processes. It follows that the logical transduction presented here does not account for the typological or learning differences attributed to saltative processes. Such differences

	x				
Formulas	1	2	3	4	5
$\texttt{voice}(x)$	$\top$	$\top$	$\top$	$\top$	$\bot$
$\texttt{sonorant}(x)$	$\bot$	$\top$	$\bot$	$\top$	$\bot$
$\texttt{continuant}(x)$	$\top$	$\top$	$\bot$	$\top$	$\bot$
$\texttt{vocalic}(x)$	$\bot$	$\top$	$\bot$	$\top$	$\bot$
$\texttt{front}(x)$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$
$\texttt{before\_front\_vowel}(x)$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$
$\phi_{\texttt{continuant}}(x)$	$\top$	$\top$	$\top$	$\top$	$\bot$
$\phi_{\texttt{dorsal}}(x)$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\phi_{\texttt{coronal}}(x)$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$
$\phi_{\texttt{del\_rel}}(x)$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$

Table 8.6: Truth values for the transformation of /vagit͡ɕ/ to [vaʒit͡ɕ] ‘to weigh’.

could be accounted for by other factors, such as the P-map (White, 2017), suitably formalized.

In addition, the logic-based approach forces researchers to attend the predictions and patterning made at the level of individual phonological features. This provides another view into phonological systems, where a crucial question becomes “What are the necessary and sufficient conditions for a position to have a particular property (like continuancy)?”

8.4 Conclusion

This chapter analyzed velar palatalization in Polish with a logical transduction applying over word models. Polish velar palatalization is a saltative process, meaning that an underlying sound undergoes a change to another sound which is, impressionistically, more different than it ‘needs’ to be to avoid being marked. The logical transduction and word models shown here formalize the process without recourse to additional machinery.

Chapter 9

Palatalization and Harmony in Lamba

HYUN JIN HWANGBO

This chapter provides a computational analysis of the following phonological processes of Lamba: vowel harmony, palatalization, and nasalization. The first part of the chapter presents the data and key phonological generalizations. The data comes from Kenstowicz and Kisseberth (1979, 71-72). The second part formalizes these generalizations in First Order logic using model-theoretic representations, which include phonological features and the successor relation. Two equivalent logical transductions are presented, one of which introduces a technique to simulate the serial derivation of a traditional rule-based analysis.

9.1 Data and phonology of Lamba

This section argues for particular underlying representations and phonological processes in Lamba.

9.1.1 Vowel Harmony

The first data set in Table 9.1 illustrates vowel harmony. The neuter and applied forms show alternations between [-ika]∼[-eka] and [-ila]∼[-ela], respectively. As Table 9.1 demonstrates, the suffixes surface as [-ika] and

[-ila] when a vowel in the stem is one of [i, u, a] while the suffixes surface as [-eka] and [-ela] when a vowel in the stem is either [e] or [o]. For example, the suffix surfaces as [-ika] in [t͡ʃitika] ‘do (neuter)’ whereas the suffix surfaces [-eka] in [t͡ʃeteka] ‘spy (neuter).’

Past	Passive	Neuter	Applied	Reciprocal	Gloss
t͡ʃita	t͡ʃitwa	t͡ʃitika	t͡ʃitila	t͡ʃitana	‘do’
tula	tulwa	tulika	tulia	tulana	‘dig’
t͡ʃeta	t͡ʃetwa	t͡ʃeteka	t͡ʃetela	t͡ʃetana	‘spy’
soŋka	soŋkwa	soŋkeka	soŋkela	soŋkana	‘pay tax’
seka	sekwa	sekeka	sekela	sekana	‘laugh at’
poka	pokwa	pokeka	pokela	pokana	‘recieve’
pata	patwa	patika	patila	patana	‘scold’

Table 9.1: Vowel Harmony in neuter and applied suffixes in Lamba.

The past, passive and reciprocal suffixes show no alternations so it follows their underlying forms are /-a, -wa, -ana/ respectively. For the underlying forms of the neuter and the applied suffixes, there are two hypotheses. The first hypothesis posits underlying forms /-eka/ and /-ela/, respectively, and a phonological process transforms them to [-ika] and [-ila] conditioned on the vowels [i, u, a]. However, this analysis cannot be seen as a natural height assimilation because the conditioning vowels have different heights. The second hypothesis posits the underlying forms as /-ika/ and /-ila/, respectively, and a phonological process transforming them to [-eka] and [-ela] conditioned on the vowels [e, o]. Such a process can be understood as a height assimilation, where the high front vowel /i/ lowers to the mid vowel [e] when the previous vowel is also mid. This generalization is stated in Gen 9.1.

Generalization 9.1. Vowel Harmony:
Front high vowels /i/ become mid [e] if the previous vowel is a mid vowel /e, o/.

9.1.2 Palatalization

The second data set in Table 9.2 shows alternations between [s]∼[ʃ] and [k]∼[t͡ʃ] before the high front vowel [i] (e.g., [fisa]∼[fiʃika] ‘hide’ and

[fuka]∼[fut͡ʃika] ‘creep’).

Past	Passive	Neuter	Applied	Reciprocal	Gloss
fisa	fiswa	fiʃika	fiʃila	fisana	‘hide’
lasa	laswa	laʃika	laʃila	lasana	‘wound’
masa	maswa	maʃika	maʃila	masana	‘plaster’
ʃika	ʃikwa	ʃit͡ʃika	ʃit͡ʃila	ʃikana	‘bury’
fuka	fukwa	fut͡ʃika	fut͡ʃila	fukana	‘creep’
kaka	kakwa	kat͡ʃika	kat͡ʃila	kakana	‘tie’

Table 9.2: Palatalization in Lamba before the high front vowel.

The underlying forms of stems exhibiting these alternations contain /s/ and /k/ since this would imply a predictable and phonetically natural palatalization process before the high front vowel. If the underlying forms of such stems contained /ʃ/ and /t͡ʃ/ that would imply a process of depalatalization before any vowels except the high front vowel, which is generally considered to be an unnatural class. The generalization of palatalization is stated in Gen 9.2.

Generalization 9.2. Palatalization:
Underlying /s, k/ surface as [ʃ, t͡ʃ] respectively before the high front /i/.

Now consider the words in Table 9.3, where the stems end in a mid vowel and /s/. No alternation of [s]∼[ʃ] is observed, presumably because Vowel Harmony (Gen 9.1) removes the front high vowel which normally conditions palatalization (Gen 9.2). In other words, in these examples, vowel harmony can be said to bleed palatalization.

Past	Passive	Neuter	Applied	Reciprocal	Gloss
t͡ʃesa	t͡ʃeswa	t͡ʃeseka	t͡ʃesela	t͡ʃesana	‘cut’
kosa	koswa	koseka	kosela	kosana	‘be strong’

Table 9.3: Vowel harmony and palatalization in Lamba

In grammars with serially ordered rules, this kind of interaction can be modeled by ordering vowel harmony before palatalization. In grammars

with ranked constraints, this kind of interaction can be modeled by ranking the constraints responsible for vowel harmony above the ones responsible for palatalization.

Generalization 9.3. Relative Priority of Palatalization and Vowel Harmony:
Palatalization has less priority than vowel harmony.

Here, "priority" means "ordered earlier" in grammars with rules and "ranked higher" in grammars with constraints.

Another way to capture the environments where palatalization does not take place is to revise the palatalization generalization to specify that only [s] or [k] sounds which do not follow mid vowels palatalize (Gen 9.4).

Generalization 9.4. Palatalization (alternate):
Underlying /s, k/ segments that are not proceeded by mid vowels surface as [ʃ, t͡ʃ] before the high front /i/.

This generalization effectively introduces the vowel harmony environment into palatalization. While this introduces some redundancy, it has the advantage of being a true statement.

Summarizing, it seems clear there is a palatalization process affecting /s, k/ in Lamba. There are at least two ways to analyze it, however. One way is with a broad generalization (Gen 9.2) which has lower priority with respect to vowel harmony as expressed in Gen 9.3. Another way is to provide a narrower generalization (Gen 9.4), which recapitulates to some extent an aspect of the environment relevant to vowel harmony. Both analyses are considered in the computational analysis later in this chapter.

9.1.3 Nasalization

The final data set in Table 9.4 presents two additional surface variants of the applied suffix: [-ina]∼[-ena]. These variants only occur in stems ending with a nasal [m, n, ɲ, ŋ].

Past	Passive	Neuter	Applied	Reciprocal	Gloss
ima	imwa	imika	imi<u>n</u>a	imana	‘rise’
puma	pumwa	pumika	pumi<u>n</u>a	pumana	‘flog’
mena	menwa	meneka	mene<u>n</u>a	menana	‘grow’
fweɲa	fweɲwa	fweɲeka	fweɲe<u>n</u>a	fweɲana	‘scratch’
pona	ponwa	poneka	pone<u>n</u>a	ponana	‘fall’
ŋaŋa	ŋaŋwa	ŋaŋika	ŋaŋi<u>n</u>a	ŋaŋana	‘snigger’

Table 9.4: Nasalization of the lateral when the preceding consonant is nasal.

The word [soŋk-ela] ‘pay tax (applied)’ shown in Table 9.1 makes clear that the nasal cannot be any preceding consonant but must be the one previous to the lateral.

Recall the applied suffix was originally analyzed as underlying /-ila/. Together, with the data in Table 9.4 this would imply a nasalization process, by which a lateral becomes a nasal stop if the previous consonant is a nasal. In addition, this approach accounts for the variation between [-ina]∼[-ena] by the aforementioned process of vowel harmony. In other words, vowel harmony and nasalization operate independently.

The alternative hypothesis that the underlying form of the applied suffix is /-ina/ would instead imply a lateralization process, whereby /n/ surfaces as [l] if the previous consonant is not a nasal. This analysis is less natural because it is neither an example of assimilation nor dissimilation.

Therefore, the best analysis is the first one considered which continues to adopt the underlying form of the applied suffix as /-ila/, and introduce a nasalization process. This process is expressed in Gen 9.5.

Generalization 9.5. Nasalization:
Lateral /l/ becomes a nasal [n] if the previous consonant is nasal.

9.1.4 Summary

So far, I have justified underlying representations and identified three phonological processes in Lamba. First, high front vowels lower to mid when the previous vowel is mid. Second, palatalization of /s, k/ occur before high front vowels as long as the /s, k/ are not preceded by mid

vowels. Alternatively, this second generalization could be expressed more simply as "/s, k/ palatalize before high front vowels" as long as it is also stated that palatalization is bled by vowel harmony. Third, laterals become nasal if the previous consonant is nasal.

9.2 Computational formalization of Lamba

The rest of the chapter presents model theoretic representations and two logical transductions which give computational treatments of the phonological analyses above. The representations use phonological features and the successor relation. First Order logic is also used. Thus the analyses are presented in the logical language FO(features,$\triangleleft$).

The two transductions are the same when it comes to nasalization and vowel harmony. The first transduction is faithful to the alternate palatalization generalization (Gen 9.4). The second is more faithful to the broader Palatalization generalization (Gen 9.2) and the generalization that vowel harmony takes priority over palatalization (Gen 9.3). This second analysis introduces a technique that allows one to simulate rule ordering with logical transductions. The two transductions are equivalent, however, in that they map the same inputs to the same outputs.

9.2.1 Representations

The word models for the underlying and surface representations are the same. They contain the binary relation successor ($\triangleleft$) and unary relations for phonological features. Figure 9.1 presents the segments present in the words in the Lamba datasets.

Table 9.5 presents a complete set of features sufficient for distinguishing the surface inventory and for the present analysis. The features are privative and not binary. For example, a segment x is continuant whenever $\texttt{stop}(x)$ evaluates to `false`. A segment x is non-continuant whenever $\texttt{stop}(x)$ evaluates to `true`. Similarly, a segment x is a vowel provided $\texttt{voc}(x)$ is `true` and non-vocalic when $\texttt{voc}(x)$ is `false`. While additional features could be included to be more faithful to the eventual phonetic implementation (such as features for voice and consonantal), they are omitted here because they are, strictly speaking, not necessary to account for the generalizations provided in the previous section.

	Labial	Coronal	Velar
oral stop	p	t	k
affricate		t͡ʃ	
fricative	f	s ʃ	
nasal stop	m	n ɲ	ŋ
lateral		l	
approximant	w		

	Front	Back
high	i	u
mid	e	o
low	a	

Figure 9.1: Consonantal and Vowel inventories of Lamba.

The signature for both the underlying and surface forms are shown in Equation 9.1.

$$\mathfrak{R} = \{\texttt{voc}, \texttt{low}, \texttt{high}, \texttt{back}, \texttt{delrel}, \texttt{stop}, \texttt{nas}, \texttt{lat}, \texttt{appr}, \texttt{lab}, \texttt{cor}, \texttt{dist}, \texttt{dor}, \lhd\} \tag{9.1}$$

All representations used throughout are these $\mathfrak{R}$-structures.

9.2.2 A Logical Transduction

The logical transduction in this section presents formulas which capture the generalizations of nasalization, vowel harmony, and the alternate palatalization generalization (Gen 9.4), in that order.

The domain formula, copy set, licensing formula, and successor formula are defined as in (9.2), (9.3), (9.4), and (9.5), respectively.

$$\phi_{\texttt{domain}} \stackrel{\text{def}}{=} \texttt{true} \tag{9.2}$$

$$C \stackrel{\text{def}}{=} \{1\} \tag{9.3}$$

$$\phi_{\texttt{license}} \stackrel{\text{def}}{=} \texttt{true} \tag{9.4}$$

$$\phi_{\lhd}(x, y) \stackrel{\text{def}}{=} x \lhd y \tag{9.5}$$

This ensures that the function applies to every input, that there is no deletion, epenthesis, or rearrangement of positions in the input. The portion of Lamba phonology studied here shows all changes are featural changes, to be captured with the unary relations in $\mathfrak{R}$.

In particular, nasalization implicates changes to the features nasal, lateral, and continuant; vowel harmony implicates changes to the feature

vocalic	voc	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{\text{a,e,i,o,u}\}\}$
low	low	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i = \text{a}\}$
high	high	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{\text{i,u}\}\}$
back	back	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{\text{o,u}\}\}$
delayed release	delrel	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i = \widehat{\text{tʃ}}\}$
non-continuant	stop	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{\text{p,t,}\widehat{\text{tʃ}}\text{,k,m,n,ŋ,ɲ}\}\}$
nasal	nas	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{\text{m,n,ŋ,ɲ}\}\}$
lateral	lat	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i = \text{l}\}$
approximant	appr	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i = \text{w}\}$
labial	lab	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{\text{p,f,m,w}\}\}$
coronal	cor	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{\text{t,}\widehat{\text{tʃ}}\text{,s,ʃ,n,ɲ,l}\}\}$
distributed	dist	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{\widehat{\text{tʃ}}\text{,ʃ,ɲ}\}\}$
dorsal	dor	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i \in \{\text{k,ŋ}\}\}$

Table 9.5: Features and their interpretations for words $w = a_1 \dots a_n$ with domain $D = \{1, 2, \dots n\}$.

high; and palatalization implicates changes to the features dorsal, coronal, distributed, and delayed release. Unary relations not corresponding to these features are never changed. This last fact is expressed in (9.6) below.

$$\forall \texttt{f} \in \{\texttt{voc}, \texttt{low}, \texttt{back}, \texttt{appr}, \texttt{lab}\} : \quad \phi_{\texttt{f}} \stackrel{\text{def}}{=} \texttt{f}(x) \tag{9.6}$$

Nasalization

Nasals on the surface form are derived from underlyingly nasals or from an underlyingly lateral whose previous consonant is a nasal (Gen 9.5). In all the examples provided, the previous consonant is always separated from the lateral by exactly one vowel. Therefore, this analysis adapts that generalization to one where laterals following a nasal-vowel sequence become nasalized.[1] The user-defined predicate in (9.7) is a short hand for a lateral in this environment. It follows that positions in the output structure

[1] If one wanted to describe the "previous consonant is a nasal" condition using First Order logic, one would have to add general precedence to the signature $\mathfrak{R}$. Then one

which are nasals, laterals, and stops are defined with the predicates shown below.

$$\texttt{NVL}\,(x) \stackrel{\text{def}}{=} \texttt{lat}(x) \wedge (\exists y, z)[\texttt{nas}(y) \wedge \texttt{voc}(z) \wedge y \lhd z \lhd x] \tag{9.7}$$

$$\phi_{\texttt{nas}}(x) \stackrel{\text{def}}{=} \texttt{nas}(x) \vee \texttt{NVL}\,(x) \tag{9.8}$$

$$\phi_{\texttt{lat}}(x) \stackrel{\text{def}}{=} \texttt{lat}(x) \wedge \neg\, \texttt{NVL}\,(x) \tag{9.9}$$

$$\phi_{\texttt{stop}}(x) \stackrel{\text{def}}{=} \texttt{stop}(x) \vee \texttt{NVL}\,(x) \tag{9.10}$$

Vowel Harmony

Next consider vowel harmony. High vowels surface as high only when they are underlyingly high and when they do not follow a mid-vowel-consonant sequence. A user-defined predicate of the environment where a mid vowel followed by a consonant and a high vowel is in (9.12). Since mid vowels are neither high nor low, they are represented as negation of both high and low as shown in (9.11). Thus, whether a vowel is high or not on the surface can be defined as in (9.13).

$$\texttt{MidV}\,(x) \stackrel{\text{def}}{=} \texttt{voc}(x) \wedge \neg\texttt{high}(x) \wedge \neg\texttt{low}(x) \tag{9.11}$$

$$\texttt{MidCHi}\,(x) \stackrel{\text{def}}{=} \texttt{high}(x) \wedge (\exists y, z)[\,\texttt{MidV}\,(y) \wedge \neg\texttt{voc}(z) \wedge y \lhd z \lhd x] \tag{9.12}$$

$$\phi_{\texttt{high}}(x) \stackrel{\text{def}}{=} \texttt{high}(x) \wedge \neg\,\texttt{MidCHi}\,(x) \tag{9.13}$$

The formulas provided so far correctly predict the output as [ponena] as shown in Table 9.6. Figure 9.2 visualizes the $\mathfrak{R}$-structures of the input /ponila/ and the output [ponena] 'fall (applied).' Observe the changes on the second domain element in the output structure — namely, the deletion of `high` and `lateral`, and the inclusion of `nasal` and `stop`.

---

could define `previous_cons_is_nas(x)` as $\exists y[\texttt{nas}(y) \wedge \forall z[y < z < x \rightarrow \texttt{voc}(z)]]$. This would make different predictions for hypothetical URs like /pemoila/.

	/p	o	n	i	l	a/
			x			
Formulas	1	2	3	4	5	6
$\texttt{nas}(x)$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$
$\texttt{lat}(x)$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$
$\texttt{stop}(x)$	$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$
$\texttt{NVL}\ (x)$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$
$\phi_{\texttt{nas}}(x)$	$\bot$	$\bot$	$\top$	$\bot$	$\top$	$\bot$
$\phi_{\texttt{lat}}(x)$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\phi_{\texttt{stop}}(x)$	$\top$	$\bot$	$\top$	$\bot$	$\top$	$\bot$
$\texttt{high}(x)$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$
$\texttt{MidV}\ (x)$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$
$\texttt{MidCHi}\ (x)$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$
$\phi_{\texttt{high}}(x)$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

Table 9.6: Truth table for the map /ponila/ to [ponena] ‘fall (applied)’.

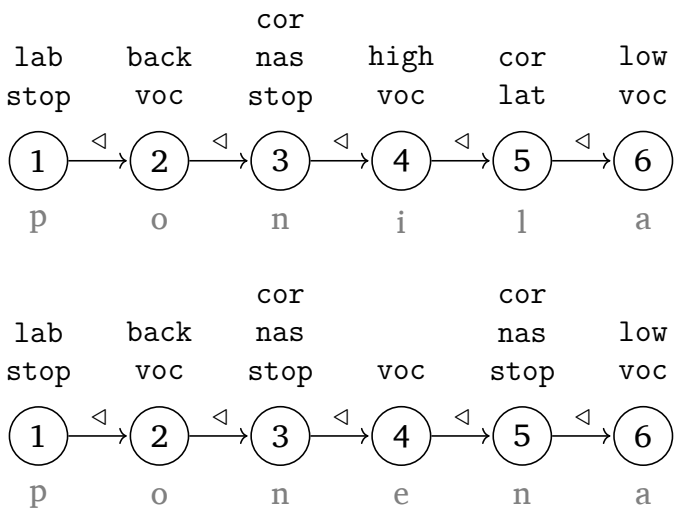


Figure 9.2: Illustrating the input and output structures corresponding to underlying /ponila/ surfacing as [ponena] ‘fall (applied)’.

Palatalization

The alternate generalization of palatalization (Gen 9.4) states that an /s/ after a non-mid vowel and before a high front vowel [i] surfaces as [ʃ], otherwise as [s]. Likewise, a dorsal sound /k/ after a non-mid vowel and before a high front vowel surfaces as [t͡ʃ], and otherwise [k]. Together, these changes affect the features distributed, delayed release, coronal and dorsal. However, it will be helpful first to specify the environment where palatalization occurs, which is shown in (9.17).

$$\texttt{s}\,(x) \stackrel{\text{def}}{=} \texttt{cor}(x) \wedge \neg\texttt{stop}(x) \wedge \neg\texttt{dist}(x) \wedge \neg\texttt{lat}(x) \wedge \neg\texttt{nas}(x) \tag{9.14}$$

$$\texttt{k}\,(x) \stackrel{\text{def}}{=} \texttt{dor}(x) \wedge \neg\texttt{nas}(x) \tag{9.15}$$

$$\texttt{i}\,(x) \stackrel{\text{def}}{=} \texttt{high}(x) \wedge \neg\texttt{back}(x) \tag{9.16}$$

$$\texttt{pal\_env}\,(x) \stackrel{\text{def}}{=} (\exists y, z)[z \lhd x \lhd y \wedge \neg\,\texttt{MidV}\,(z) \wedge\ \texttt{i}\,(y)] \tag{9.17}$$

When /s/ surfaces as [ʃ] the only feature that changes is distributed. When /k/ surfaces as [t͡ʃ] the features coronal, dorsal, and distributed change. The formula for the features distributed, coronal and dorsal are in (9.18),(9.19) and (9.20), respectively.

$$\phi_{\texttt{dist}}(x) \stackrel{\text{def}}{=} \texttt{dist}(x) \vee (\ \texttt{pal\_env}\ (x) \wedge (\ \texttt{s}\ (x) \vee\ \texttt{k}\ (x))) \tag{9.18}$$

$$\phi_{\texttt{dor}}(x) \stackrel{\text{def}}{=} \texttt{dor}(x) \wedge (\ \texttt{k}\ (x) \rightarrow \neg\ \texttt{pal\_env}\ (x)) \tag{9.19}$$

$$\phi_{\texttt{cor}}(x) \stackrel{\text{def}}{=} \texttt{cor}(x) \vee (\ \texttt{k}\ (x) \wedge\ \texttt{pal\_env}\ (x)) \tag{9.20}$$

Figure 9.3 visualizes the $\mathfrak{R}$-structures of the input /fisila/ 'hide' and its output [fiʃila]. Observe the inclusion of `distributed` on the second domain element in the output structure.

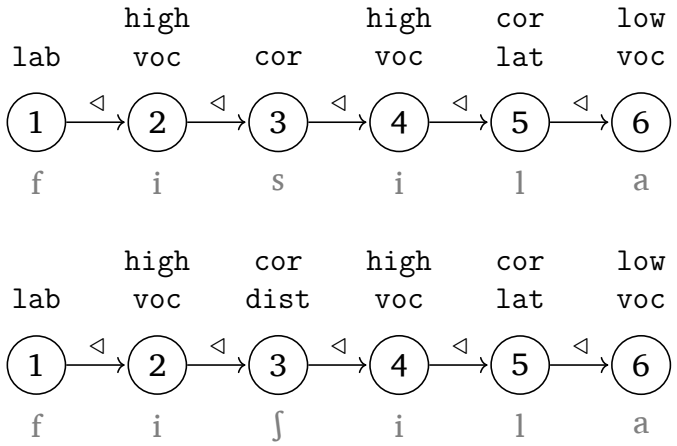


Figure 9.3: Illustrating the input and output structures corresponding to underlying /fisila/ surfacing as [fiʃila] 'hide (applied).'

Lastly, Figure 9.4 shows the input and output structures of the mapping /kosila/ $\rightarrow$ [kosela] 'be strong (applied).' Domain element 4 will not surface high due to (9.13). In addition, domain element 3 does not satisfy $\phi_{\texttt{dist}}$ because its preceding vowel is mid (see formulas 9.18 and 9.17).

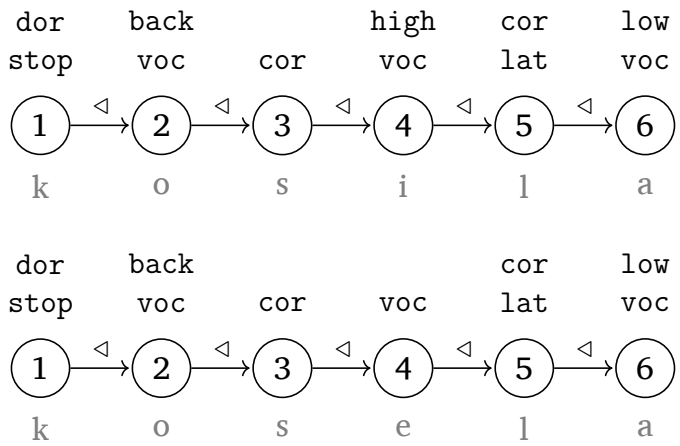


Figure 9.4: Illustrating the input and output structures corresponding to underlying /kosila/ surfacing as [kosela] 'hide (applied).'

This concludes the first analysis of nasalization, vowel harmony, and palatalization, effectively incorporating the vowel harmony environment

into palatalization.

9.2.3 Another Logical Transduction

The rest of this chapter provides another computational analysis akin to the use of serially ordered rules. Recall that some may argue that the correct account of palatalization is not the narrower generalization made in Gen 9.4, but the broader one in Gen 9.2. The scope of this broader generalization is contained by the application of vowel harmony (Gen 9.1) *before* palatalization.

One way to accomplish this is simply to write logical transductions for each of the three phonological processes and compose them. This takes advantage of the fact that FO($\lhd$) is closed under composition (Courcelle and Engelfriet, 2012). However, in this analysis, I pursue a different route, and instead use the copyset to *simulate* this analysis. Each process occurs on a single copy, whose formulas only refer to the previous copy. Finally, only the last copy is licensed, which realizes the surface form. Essentially, the copies act as intermediate representations, with the last copy corresponding to the surface form.

The word models here have the signature as in (9.1) and are thus the same $\mathfrak{R}$-structures as presented previously. As with the transduction the domain formula asserts the transduction applies to any $\mathfrak{R}$-structures representing underlying forms.

$$\phi_{\texttt{domain}} \stackrel{\text{def}}{=} \texttt{true} \tag{9.21}$$

$$C \stackrel{\text{def}}{=} \{1,2,3\} \tag{9.22}$$

$$\phi^1_{\texttt{license}} \stackrel{\text{def}}{=} \texttt{false} \tag{9.23}$$

$$\phi^2_{\texttt{license}} \stackrel{\text{def}}{=} \texttt{false} \tag{9.24}$$

$$\phi^3_{\texttt{license}} \stackrel{\text{def}}{=} \texttt{true} \tag{9.25}$$

$$\phi^{(i,j)}_{\lhd}(x,y) \stackrel{\text{def}}{=} \begin{cases} x \lhd y, i = j \\ \texttt{false}, i \neq j \end{cases} \tag{9.26}$$

The copy set has cardinality three, reflecting the three processes. Nasalization will be defined on the first copy, vowel harmony on the second copy,

and palatalization on the third copy.[2] Because the copy set is of size three, there are three licensing functions, one for each copy. Of these, only the elements of third copy are licensed, the others are not part of the output structure.

There are also a number of formulas for the successor relation. Recall from Chapter 3 that $\phi_{\triangleleft}^{(i,j)}(x,y)$ is read as "the ith copy of x stands in the successor relation to the jth copy of y." Formula 9.26 is a shorthand for each formula needed to define the successor relation where $i,j \in C$. When $i = j$, ith copy of x will stands in the successor relation to the jth copy of y if and only if $x \triangleleft y$ in the output. In all other cases, the ith copy of x will not stand in the successor relation to the jth copy of y. The above formulas essentially structure the output as shown in Figure 9.5, with the unary relations defined below filling in the properties of each element.

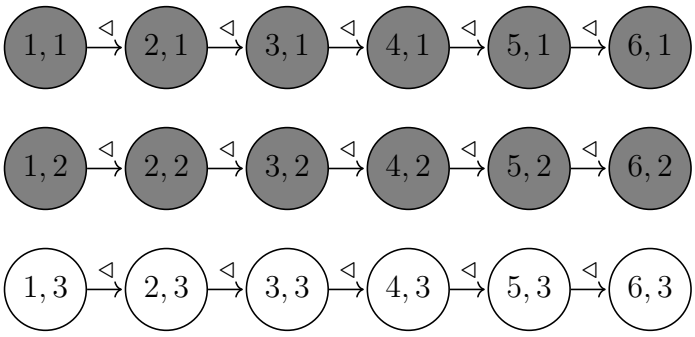


Figure 9.5: The effect of the copyset and the licensing and successor formulas on the output structure of an input word of length six. Unlicensed domain elements are grayed out. Element (x,y) refers to the yth copy of x. Unary relations are omitted.

The formulas for the phonological features on the first copy are defined

[2]Since nasalization and vowel harmony are independent non-interacting processes, they could both occur in a single copy. However, the separation of one copy per process emphasizes the technique being introduced.

below. These are defined the same way as in the previous section.

$$\phi^1_{\texttt{nas}}(x) \stackrel{\text{def}}{=} \texttt{nas}(x) \vee \texttt{NVL}\,(x) \tag{9.27}$$

$$\phi^1_{\texttt{lat}}(x) \stackrel{\text{def}}{=} \texttt{lat}(x) \wedge \neg\, \texttt{NVL}\,(x) \tag{9.28}$$

$$\phi^1_{\texttt{stop}}(x) \stackrel{\text{def}}{=} \texttt{stop}(x) \vee \texttt{NVL}\,(x) \tag{9.29}$$

All other features f are defined with $\phi^1_{\texttt{f}}(x) \stackrel{\text{def}}{=} \texttt{f}(x)$. For example, $\phi^1_{\texttt{high}}(x) \stackrel{\text{def}}{=} \texttt{high}(x)$ and $\phi^1_{\texttt{dist}}(x) \stackrel{\text{def}}{=} \texttt{dist}(x)$. This effectively realizes the output of the nasalization process on the first copy.

The second copy realizes vowel harmony. To simulate the effects of rule ordering, for each feature f, $\phi^2_{\texttt{f}}(x)$ will not reference the input directly, but instead the unary relations on the first copy. This takes advantage of the fact that formulas like $\phi^1_{\texttt{f}}(x)$ are essentially like user-defined predicates available for use.

To illustrate, consider that in vowel harmony, only the feature high changes. Therefore, for all other features f are defined with $\phi^2_{\texttt{f}}(x) \stackrel{\text{def}}{=} \texttt{f}^1(x)$. For instance, $\phi^2_{\texttt{nas}}(x) \stackrel{\text{def}}{=} \texttt{nas}^1(x)$, which effectively propagates any nasalization in the first copy to the second copy. The formulas for vowel harmony are shown below and it will be instructive to compare them to the formulas in (9.11), (9.12), and (9.13).

$$\texttt{MidV2}\,(x) \stackrel{\text{def}}{=} \phi^1_{\texttt{voc}}(x) \wedge \neg\phi^1_{\texttt{high}}(x) \wedge \neg\phi^1_{\texttt{low}}(x) \tag{9.30}$$

$$\texttt{MidCHi2}\,(x) \stackrel{\text{def}}{=} \phi^1_{\texttt{high}}(x) \wedge (\exists y, z)[\,\texttt{MidV2}\,(y) \wedge \phi^1_{\texttt{cons}}(z) \wedge \phi^{(1,1)}_{\lhd}(y,z) \wedge \phi^{(1,1)}_{\lhd}(z,x)] \tag{9.31}$$

$$\phi^2_{\texttt{high}}(x) \stackrel{\text{def}}{=} \texttt{voc}^1(x) \wedge \texttt{high}^1(x) \wedge \neg\,\texttt{MidCHi2}\,(x) \tag{9.32}$$

Finally we turn to palatalization. This transduction models the generalization in Gen 9.2, where /s, k/ palatalize provided they precede a high front vowel. This generalization does not refer to the quality of the vowel

before /s, k/ as was the case with the alternate generalization in Gen 9.4.

$$\texttt{s2}\,(x) \stackrel{\text{def}}{=} \phi^2_{\texttt{cor}}(x) \wedge \neg\phi^2_{\texttt{stop}}(x) \wedge \neg\phi^2_{\texttt{dist}}(x) \wedge \neg\phi^2_{\texttt{lat}}(x) \wedge \neg\phi^2_{\texttt{nas}}(x) \tag{9.33}$$

$$\texttt{k2}\,(x) \stackrel{\text{def}}{=} \phi^2_{\texttt{dor}}(x) \wedge \neg\phi^2_{\texttt{nas}}(x) \tag{9.34}$$

$$\texttt{i2}\,(x) \stackrel{\text{def}}{=} \phi^2_{\texttt{high}}(x) \wedge \neg\phi^2_{\texttt{back}}(x) \tag{9.35}$$

$$\texttt{pal\_env2}\,(x) \stackrel{\text{def}}{=} (\exists y)[\,\texttt{i2}\,(y) \wedge \phi^{(2,2)}_{\triangleleft}(x,y)] \tag{9.36}$$

The formulas for the features distributed, coronal and dorsal are shown below. Formula 9.39 for example says, "The third copy of x will be coronal provided the second copy of x is coronal OR the second copy of x is the velar stop that is in the palatal environment." Again, it is instructive to compare these formulas to the ones in (9.18), (9.19), and (9.20).

$$\phi^3_{\texttt{dist}}(x) \stackrel{\text{def}}{=} \phi^2_{\texttt{dist}}(x) \vee (\,\texttt{pal\_env2}\,(x) \wedge (\,\texttt{s2}\,(x) \vee\, \texttt{k2}\,(x))) \tag{9.37}$$

$$\phi^3_{\texttt{dor}}(x) \stackrel{\text{def}}{=} \phi^2_{\texttt{dor}}(x) \wedge (\,\texttt{k2}\,(x) \rightarrow \neg\,\texttt{pal\_env2}\,(x)) \tag{9.38}$$

$$\phi^3_{\texttt{cor}}(x) \stackrel{\text{def}}{=} \phi^2_{\texttt{cor}}(x) \vee (\,\texttt{k2}\,(x) \wedge\, \texttt{pal\_env2}\,(x)) \tag{9.39}$$

Figure 9.6 visualizes how this logical transduction uses three copies to construct the output structure for the input /ponila/ 'fall (applied)', whose input structure is the same as the one shown in Figure 9.2.

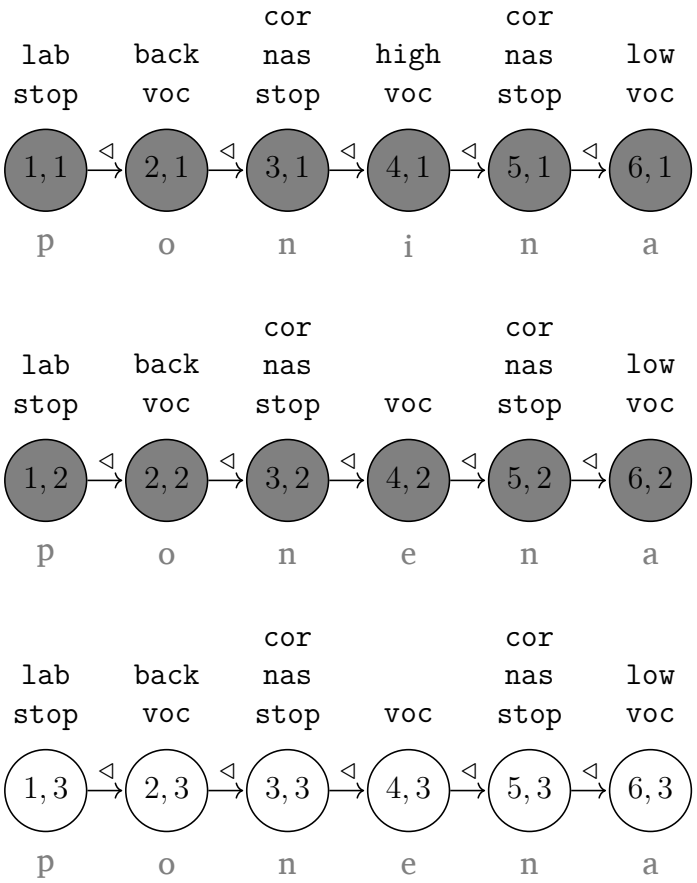


Figure 9.6: The first, second and third copies illustrating the logical transduction which maps /ponila/ to [ponena] 'fall (applied).'

The next example is the word /fisila/ 'hide (applied),' whose input is visualized in Figure 9.3. Figure 9.7 shows the first, second and third copies in the construction of the output structure.

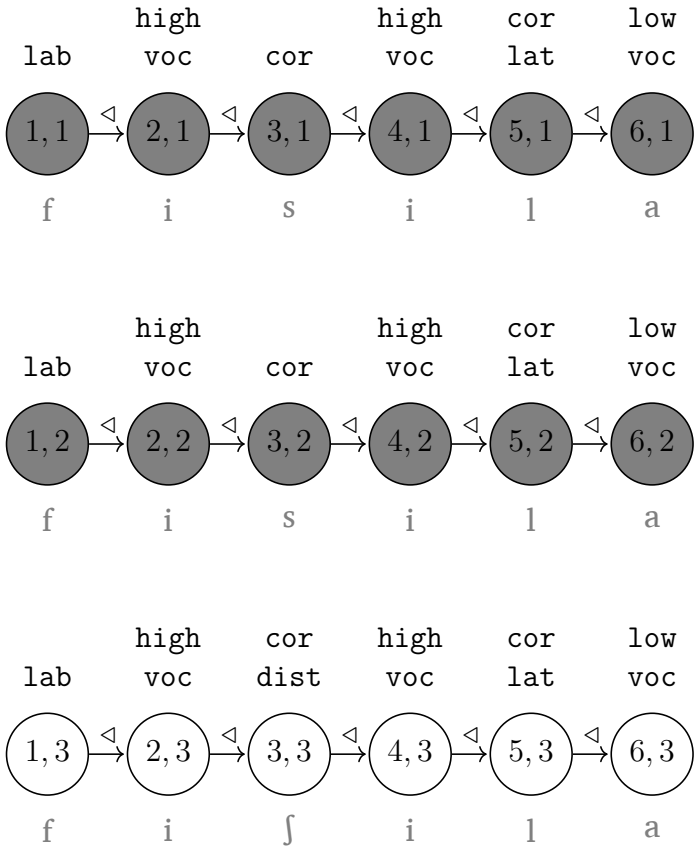


Figure 9.7: The first, second and third copies illustrating the logical transduction which maps /fisila/ to [fiʃila] 'hide.'

The last example considers the input /kosila/ 'be strong (applied).' Figure 9.8 shows the first, second and third copies in the construction of the output structure. Since nasalization does not apply, the first copy is faithful to the input structure. Observe that palatalization does not apply in node (3,3) because the environment for palatalization considers the second copy, and not the input structure directly (see formulas 9.36 and 9.35).

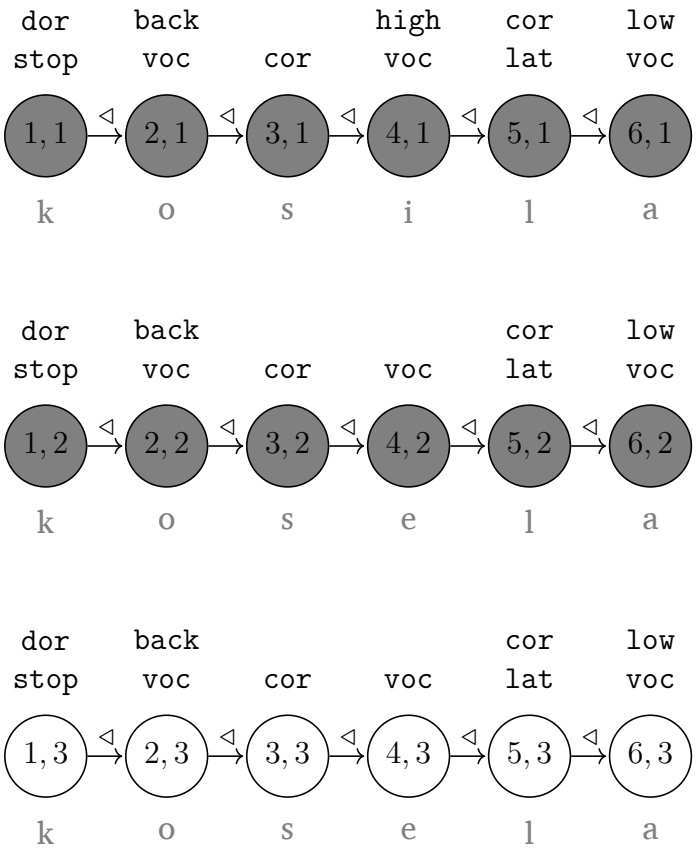


Figure 9.8: The first, second and third copies illustrating the logical transduction which maps /kosila/ to [kosela] 'be strong (applied).'

9.3 Conclusion

To sum up, this chapter provides a phonological analysis of variations of the pronunciations of the neuter and applied verbal forms in Lamba. Three phonological processes, vowel harmony, palatalization, and nasalization, were identified, with vowel harmony taking priority over palatalization. This analysis was formalized using model theoretic representations and formulas of First Order logic.

The interaction between vowel harmony and palatalization can be expressed in different ways. One way, which keeps the three processes independent of each other, essentially includes the triggering environment

for vowel harmony in the palatalization process as a blocking environment. Another approach allows palatalization to apply more broadly, but only after the application of vowel harmony. This latter approach is typical to serial rule-based analyses, where it would be said that the rule for vowel harmony bleeds the rule for palatalization.

This serial nature of this latter approach can be simulated with a single logical transduction by way of the copy set. It was shown how the intermediate representations can be associated with different copies, all of but one of which do not satisfy the licensing formulas. The elements of the copy which is licensed correspond to the output of the transduction (and thus to the output of the SPE-style grammar). In this way, this analysis illustrates a general strategy for converting phonological grammars of ordered rules into a single logical transduction.

Chapter 10

Obstruent devoicing and g-deletion in Turkish

KRISTINA STROTHER-GARCIA

10.1 Introduction

There are two alternations involving obstruents in Turkish: a widely-attested process of obstruent devoicing, and a much more restricted process of velar stop deletion, which I analyze as *g*-deletion. This chapter reviews the basic facts concerning these alternations and presents a logical transduction modeling these processes.

The two obstruent alternations described above are illustrated by the following data from Eliasson (1985), with the transcriptions adapted to the International Phonetic Alphabet.

Table 10.1: Obstruent alternations in Turkish

	Singular	**Plural**	**3rd Person Poss.**	**Gloss**
a.	kitap	kitap-lar	kitab-i	'book'
	paket	paket-ler	paked-i	'package'
	renk	renk-ler	reng-i	'color'
b.	sap	sap-lar	sap-i	'stalk'
	at	at-lar	at-i	'horse'
	ek	ek-ler	ek-i	'joint'
c.	t͡ʃilek	t͡ʃilek-lar	t͡ʃile-i	'strawberry'
	gœk	gœk-ler	gœ-y	'heaven'

Comparing the forms in Table 10.1(a) to those in 10.1(b), it appears that voiceless obstruents alternate with their voiced counterparts in coda position (e.g., [ki.tap] and [ki.tap.lar] vs. [ki.ta.bi] 'book'). These alternations are easily accounted for if the final obstruents of the roots in Table 10.1(a) are underlyingly voiced, and are subject to a process of obstruent coda devoicing. These data are not compatible with an obstruent voicing process because it would incorrectly generate forms such as *[sab-i] 'stalk' and *[ad-i] 'horse' in Table 10.1(b).

Additionally, there is a [k∼ ∅] alternation in Table 10.1(c) e.g. [t͡ʃilek] and [t͡ʃileklar] vs. [t͡ʃile-i] 'strawberry'. Regardless of whether the final velar obstruent in such roots are underlyingly voiced or voiceless, they are expected to surface faithfully in the 3rd person possessive form because they are syllabified as onsets: *[t͡ʃi.le.gi, t͡ʃi.le.ki]. Because [k] surfaces in the 3rd person possessive form of 'joint' [e.ki] in Table 10.1(c), [k] is clearly not banned in this environment.[1] In contrast, [g] never surfaces intervocalically.[2]

Taken together, these facts provide evidence that the roots in Table 10.1(c) end in /g/, even though this phoneme never surfaces fully faithfully in this position: /t͡ʃileg, t͡ʃileglar, t͡ʃilegi/. When it surfaces in coda position, it is devoiced: [t͡ʃilek, t͡ʃileklar]. When it surfaces intervocalically, it deletes: [t͡ʃilei]. By comparison, note that the syllable-final /g/ in

[1] Eliasson (1985), following Zimmer, argues for underlying /k/ and a k-deletion process which does not apply to monosyllabic stems.

[2] Note [g] does contrast with [k] in in Turkish in other positions such as word-initially.

[reng] 'color' does not delete because it follows a consonant.

I acknowledge that this phonological analysis omits some aspects of the phonology of Turkish. For instance, Eliasson (1985) presents additional data indicating that *g*-deletion is also conditioned by word size, syllable weight, and morphological properties. In addition, other phonological processes that apply to these forms, such as vowel harmony (compare 3rd person possessive [reng-i] 'color' to [gœ-y] 'heaven'), are not analyzed in this chapter. Nonetheless, the simplifications made here help illustrate how logical transductions operate, especially when the phonological generalizations involve deletion and syllable structure.

10.2 Computational Analysis

This section describes a logical transduction that models both obstruent devoicing and *g*-deletion simultaneously.

10.2.1 Representations

This analysis assumes that phonological features are the primitive representational units of phonological words, upon which phonological processes operate. Therefore, phonological features make up the unary relations in the word models, at both the underlying and surface levels. In particular, I make use of privative (rather than binary) features. Consequently, there is no [−voice]; rather, a voiceless segment x satisfies the formula $\neg\texttt{voice}(x)$.

The relational signature of the word models is in Equation 10.1 below.

$$\mathfrak{R} = \{<, \texttt{voc}, \texttt{voi}, \texttt{son}, \texttt{dor}\} \cup F \tag{10.1}$$

For the order relation, I use general precedence $(<)$ because, as explained below, it simplifies the logical transduction since deletion is involved. For features, the following abbreviations have been used.

- `voc` is short for 'vocalic'
- `voi` is short for 'voice'
- `son` is short for 'sonorant'
- `dor` is short for 'dorsal'
- F is the set containing all the other phonological features for Turkish which are otherwise irrelevant to this analysis.

10.2.2 Logical formalization

This section provides an analysis obstruent devoicing and *g*-deletion in terms of First Order logic and the $\mathfrak{R}$-structures defined in the previous section.

The domain formula is set to true so that the transduction applies to all ($\mathfrak{R}$-structures of) underlying forms. Since no epenthesis is present in the fragment of Turkish phonology under consideration, the copy set is set to {1}.

$$\phi_{\texttt{dom}} \stackrel{\text{def}}{=} \texttt{true} \tag{10.2}$$

$$C \stackrel{\text{def}}{=} \{1\} \tag{10.3}$$

Licensing and precedence formulas

The licensing formula, and the formula for precedence are provided next. They are implicated by the *g*-deletion process. In particular, since an intervocalic /g/ deletes, it must not be licensed by $\phi_{\texttt{license}}$.

It can be useful to define predicates that correspond to the traditional notion of phonemes. I provide such a predicate for /g/ below.

$$\texttt{g}\,(x) \stackrel{\text{def}}{=} \texttt{voi}(x) \wedge \texttt{dor}(x) \wedge \neg\texttt{son}(x) \tag{10.4}$$

Note that some properties typically associated with /g/ are left out (e.g., continuancy) because they are not relevant to the processes studied here. Next, I identify which positions correspond to intervocalic /g/.

$$\texttt{VgV}\,(x) \stackrel{\text{def}}{=} \texttt{g}\,(x) \wedge (\exists w, y)[\texttt{voc}(w) \wedge \texttt{voc}(y) \wedge w \lhd x \lhd y] \tag{10.5}$$

The predicate $\texttt{VgV}\,(x)$ is true of all intervocalic /g/ positions in the input structure.

Finally, the licensing formula can be stated as follows.

$$\phi_{\texttt{license}}(x) = \neg\,\texttt{VgV}\,(x) \tag{10.6}$$

In other words, all positions not corresponding to intervocalic /g/ are licensed in the surface structure. Positions corresponding to intervocalic /g/, however, are not, and they will be deleted.

To illustrate, consider the truth table in Table 10.2 illustrating the 3rd person possessive form of 'heaven': /gœgy/ $\rightarrow$ [gœy].[3]

	g	œ	g	y
		x		
Formulas	1	2	3	4
$\texttt{voc}(\texttt{x})$	$\bot$	$\top$	$\bot$	$\top$
$\texttt{g}\,(x)$	$\top$	$\bot$	$\top$	$\bot$
$\texttt{VgV}\,(x)$	$\bot$	$\bot$	$\top$	$\bot$
$\phi_{\texttt{license}}(x)$	$\top$	$\top$	$\bot$	$\top$

Table 10.2: Truth values for the transformation of /gœgy/ $\rightarrow$ [gœy] 'heaven.'

Clearly position 3 is the only one satisfying $\texttt{VgV}$, and thus is the only position not satisfying $\phi_{\texttt{license}}$.

The order relation $<$ in the surface structures are defined next.

$$\phi_{<}(x,y) \stackrel{\text{def}}{=} \quad x < y \tag{10.7}$$

This says, "position x precedes position y in the output if and only if position x precedes position y in the input." It follows from this definition that $\phi_{<}(2,3)$ and $\phi_{<}(3,4)$ are true for the $\mathfrak{R}$-structure representing /gœgy/. However, recall from Chapters 3 and 6 that the elements of the relations in the output structure are only the ones whose components are licensed. In other words, since position 3 is not licensed in the output structure, (2,3) and (3,4) will not belong to the precedence relation $(<)$ in the output structure.

This "generate and filter" strategy can help simplify the formulas needed to define a logical transduction, but care must be taken when using it. For instance, if the signature included the successor relation $(\lhd)$ instead of the precedence relation $(<)$, the "generate and filter" strategy may fail. This is because if one defines the $\phi_{\lhd}$ faithfully then when the elements of the

[3] I assume the the 3rd person possessive suffix is /y/ in this example. As mentioned, the chapter is ignoring vowel harmony.

relation containing unlicensed components are filtered out, one can be left with a structure that does not correspond to any string.

For concreteness, consider the hypothetical formula for $\phi_{\triangleleft}$ below, which maintains that the successor relation in the output is faithful that in the input. (I am also presuming the input and output signatures include $(\triangleleft)$ but not $(<)$.)

$$\phi_{\triangleleft}(x,y) \stackrel{\text{def}}{=} \quad x \triangleleft y \tag{10.8}$$

While (1,2), (2,3), and (3,4) satisfy $\phi_{\triangleleft}$, the successor relation in the output structure only contains (1,2) since the others contain the unlicensed component 3. Consequently, this structure does not order position 4 with respect to positions 1 and 2 and it is not **connected** (cf. Chapter 5). This structure is not a string.

One can use the successor relation, but one has to make sure to "stitch" the licensed positions together. The definition below is one way to accomplish this for the Turkish pattern described here.

$$\phi_{\texttt{succ}}(x,y) \stackrel{\text{def}}{=} \quad (x \triangleleft y \vee (\exists z[x \triangleleft z \triangleleft y \wedge \texttt{VgV}\,(z)]) \tag{10.9}$$

Formula 10.9 is a disjunction of two terms. The first terms says position y succeeds x in the output structure provided "y succeeds x in the input structure." The second term says position y succeeds x in the output structure provided "there is another position z between y and x that is an intervocalic /g/." In this case, since z will be deleted, position y be the successor of x in the output form. This first dijunct ensures (1,2), (2,3), and (3,4) satisfy $\phi_{\triangleleft}$, and the second disjunct ensures (2,4) satisfies $\phi_{\triangleleft}$. Since (1,2) and (2,4) are the only ones which do not contain an unlicensed component, they make the successor relation in the output structure. This is illustrated in Figure 10.1.

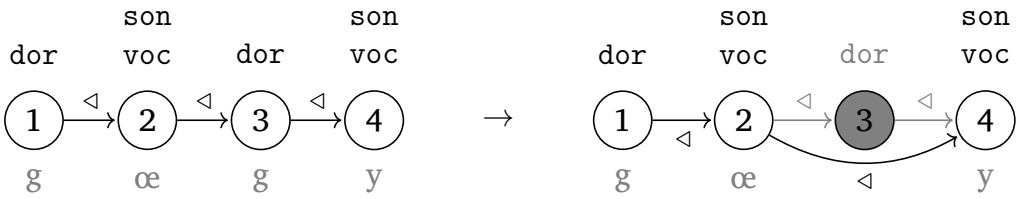


Figure 10.1: Visualization of /gœg-y/→[gœ-y]. The only unary relations shown are `dor`, `son`, and `voc`. Unlicensed domain elements, and their associated relations, are grayed out.

This issue directly relates to order-preservation, which was discussed in Chapter 3.6. The transduction described here is in fact order-preserving. In Chapter 3.6, a general solution to the problem of "stitching" together the successor relation was provided, but it depended on the the precedence relation. I hope this concrete example of the differences between the successor and precedence relations when it comes to deletion has been instructive and helps elucidate why knowing whether a transduction is order-preserving or not is useful.

It should also be clearer now why I prefer using a signature with general precedence instead of one with successor. Given that the logical transductions are defined in a way to filter out elements of relations with unlicensed components, the formula for $\phi_<$ in formula 10.7 is simpler than the one for successor in formula 10.9. This is simply because with general precedence, one does not have to worry about stitching the successor relation together to maintain a connected structure.

Formulas for the features

The featural make-up for each segment in the string is determined by the unary relations in the $\mathfrak{R}$-signature. When mapping (the $\mathfrak{R}$-structure of) any Turkish word's underlying form to (the $\mathfrak{R}$-structure of) its surface form, most unary relations are preserved. In particular, only the relation `voi` changes. For simplicity, I assume that all features not relevant to the present analyses are preserved as well as indicated by the formula below.

$$\phi_{\mathtt{f}}(x) \stackrel{\text{def}}{=} \quad \mathtt{f}(x) \text{ for all } f \in F \cup \{\mathtt{voc}, \mathtt{son}, \mathtt{dor}\} \tag{10.10}$$

Obstruent devoicing occurs only in coda position, so it will be useful to define predicates that identify parts of the syllable. I assume that all vowels (and only vowels) are syllabic nuclei. Onsets and codas can be identified using the following predicates.

$$\mathtt{onset}\,(x) \stackrel{\text{def}}{=} \quad \neg\mathtt{voc}(x) \wedge (\exists y)[\mathtt{voc}(y) \wedge x \lhd y) \tag{10.11}$$

$$\mathtt{coda}\,(x) \stackrel{\text{def}}{=} \quad \neg\mathtt{voc}(x) \wedge \neg\,\mathtt{onset}\,(x) \tag{10.12}$$

Formula (10.11) encodes the generalization that prevocalic consonants (and glides) are onsets. Whereas complex onsets are not allowed in Turkish, complex codas are. Two-segment codas are attested in the present data,

and there is no evidence for the limitations on complex coda formation. Lacking such evidence, I assume that any non-vowel (consonant or glide) that is not an onset must belong to a coda. This is formalized in (10.12), although I make no distinction here between the first and subsequent segments in a complex coda. Finally, observe that the successor predicate is not a member of the signature primitive, but is instead a user-defined precicate. I use the definition on page 53 in Chapter 2.

Next, the environment where devoicing occurs is defined.

$$\texttt{codaD}\,(x) \stackrel{\text{def}}{=} \quad \texttt{coda}\,(x) \wedge \texttt{voi}(x) \wedge \neg\texttt{son}(x) \tag{10.13}$$

The predicate $\texttt{codaD}\,(x)$ is true for any voiced obstruent in coda position. This structure is banned in surface forms, meaning it cannot appear in any output $\mathfrak{R}$-structure.

To repair a voiced obstruent in coda, its corresponding output position must not satisfy $\phi_{\texttt{voi}}$(x).

$$\phi_{\texttt{voi}}(x) \stackrel{\text{def}}{=} \quad \texttt{voi}(x) \wedge \neg\,\texttt{codaD}\,(x) \tag{10.14}$$

Formula 10.14 prevents voiced obstruents in coda position from satisfy the feature `voi` in surface forms. As such, they will be devoiced.

To illustrate, consider the truth table in Table 10.3 and Figure 10.2 illustrating the singular form of 'color': /reng/ $\rightarrow$ [renk].

	r	e	n	g
			x	
Formulas	1	2	3	4
voi(x)	$\top$	$\top$	$\top$	$\top$
son(x)	$\top$	$\top$	$\top$	$\bot$
voc(x)	$\bot$	$\top$	$\bot$	$\bot$
onset (x)	$\top$	$\bot$	$\bot$	$\bot$
coda (x)	$\bot$	$\bot$	$\top$	$\top$
codaD (x)	$\bot$	$\bot$	$\bot$	$\top$
$\phi_{\texttt{voi}}(x)$	$\top$	$\top$	$\top$	$\bot$

Table 10.3: Truth values for the transformation of /reng/ $\rightarrow$ [renk] 'color.'

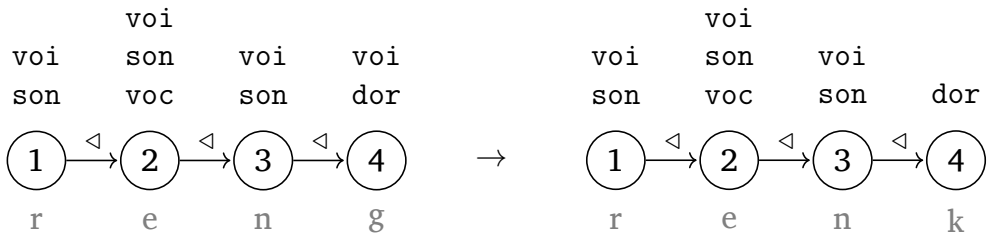


Figure 10.2: /reng/→[renk]

Observe that $\phi_{\texttt{voi}}(4)$ is false because position 4 satisfies `codaD`. All positions x not satisfying `codaD` are faithful to whether x satisfies $\texttt{voi}(x)$ in the input structure.

10.3 Conclusion

As shown in the previous section, the logical transduction proposed here accurately predicts the surface forms attested in Table 10.1. This section also provided an example of how the licensing formula can be used to model deletion. It emphasized the fact that when segments delete, the order relation needs to be defined in a way that is coherent. This is easy in the case for the general precedence relation, but more care needs to be taken for the successor relation.

This analysis made several simplifying assumptions in its account. Despite them, it is clear that a logical transduction expressed in FO($<$) is capable of modeling multiple phonological processes simultaneously to account for aspects of Turkish phonology. Future work could expand the analysis here to account for vowel harmony, and to reflect generalizations that take into account morphological information, word size, and syllable weight.

Chapter 11

Cluster Reduction in Tibetan

Jeffrey Heinz and Kristina Strother-Garcia

This chapter provides a computational analysis of the variation in the pronunciation of some Tibetan numbers.[1] In Tibetan, the names for certain numbers greater than ten are formed by concatenating the word for a number less than ten with the word that means 'ten.' We refer to these combinations as "compound numerals." In particular, numbers from eleven to nineteen are formed by prefixing [d͡ʒu] 'ten' to the number being added to ten. For example, [d͡ʒuŋa] 'fifteen' has the number [ŋa] 'five' as the first element of the compound, so it can be analyzed as [d͡ʒu-ŋa], literally meaning 'ten-five.' To form multiples of ten, the root meaning 'ten' is concatenated as a suffix to the number being multiplied by ten, as in [ŋabd͡ʒu] 'fifty' (literally 'five-ten'). These examples show that there is some variation in pronunciation across different compound numerals, as there is a [b] in the compound numeral for 'fifty' but not for 'fifteen.' Other such consonant∼zero alternations exist, and they are underlined in Table 11.1. Alternations with zero are typically accounted for by either attributing a process of epenethesis or deletion to the phonology.

[1] The "Tibetan Numeral Problem" is presented in Halle and Clements (1983, p. 105) and Odden (2014, p. 112).

x		$10 + x$		$10 \times x$	
d͡ʒig	‘one’	d͡ʒugd͡ʒig	‘eleven’		
ʃi	‘four’	d͡ʒubʃi	‘fourteen’	ʃibd͡ʒu	‘forty’
ŋa	‘five’	d͡ʒuŋa	‘fifteen’	gubd͡ʒu	‘ninety
gu	‘nine’	d͡ʒurgu	‘nineteen’	ŋabd͡ʒu	‘fifty’
d͡ʒu	‘ten’				

Table 11.1: Tibetan numbers and compound numerals.

11.1 Phonological Analysis

Consider first an approach based on epenthesis. We first observe that [b] follows the first root and precedes [d͡ʒu] in all multiples of ten greater than ten: [ʃibd͡ʒu] ‘forty,’ [ŋabd͡ʒu] ‘fifty,’ and [gubd͡ʒu] ‘ninety.’ In each case, [b] is preceded by a vowel and followed by the affricate [d͡ʒ]—there is no more specific generalization we can make about the environment of this [b]. If this [b] does not appear in the underlying form of either the root meaning ‘five’ or the root meaning ‘ten,’ one could propose a phonological process which epenthesizes [b] between a vowel and the affricate [d͡ʒ]. However, this process would incorrectly predict the surface form [*d͡ʒubd͡ʒig] instead of [d͡ʒugd͡ʒig] ‘eleven.’

Furthermore, as Table 11.1 shows, the roots used to form other numbers in the teens vary in pronunciation as well. ‘Fourteen’ is [d͡ʒubʃi] rather than *[d͡ʒuʃi] and ‘nineteen’ is [d͡ʒurgu] rather than *[d͡ʒugu]. This variation is not easily explained by a single process of epenthesis because the consonants that appear in the compound numerals are not predictable based on the environments. Although one may be able to conceive of multiple epenthetic processes that could explain this variation, we argue there is a simpler solution: the underlying forms of certain numerals contain a consonant which is absent in their morphologically bare surface forms.

This latter hypothesis posits a deletion process targeting consonants. There is still, however, more than one possible combination of underlying forms. Reconsider the example above: [ŋabd͡ʒu] ‘fifty’ can either be morphologically analyzed as /ŋab-d͡ʒu/ or /ŋa-bd͡ʒu/. We first consider the ramifications of /b/ belonging to the underlying form of ‘five’ and then of it belonging to the underlying form of ‘ten.’

If the underlying form of 'ten' were /d͡ʒu/ and the underlying form of 'five' were /ŋab/, one would reasonably conclude the underlying forms of 'four' and 'nine' are /ʃib/ and /gub/, respectively, because this would conform to the pattern observed in multiples of ten. Under this analysis, one could account for the surface forms of the multiples of ten by positing a word-final consonant deletion process. This accurately predicts the absence of a word-final [b] in the morphologically bare surface forms and in the surface forms for 'fourteen', 'fifteen', and 'nineteen.' It would also correctly predict the word-internal [b] that appears in the surface forms for 'forty', 'fifty', and 'ninety'.

However, the surface forms [d͡ʒig] 'one' and [d͡ʒugd͡ʒig] 'eleven' would not be predicted because both have a word-final consonant, [g]. One could modify this hypothesis to say only /b/ deletes word-finally, narrowing the process' scope. There is another problem with this analysis; it fails to explain why the surface forms [d͡ʒubʃi] 'fourteen' and [d͡ʒurgu] 'nineteen' also include consonants not present in the underlying forms of the supposed roots that make up these compound numerals.

On the other hand, if the underlying form of 'ten' were /bd͡ʒu/ and the underlying form of 'five' were /ŋa/, the morphologically bare form [d͡ʒu] 'ten' would be accounted for by a deletion process which targets the first consonant of a word-initial consonant cluster. If 'ten' is underlying /bd͡ʒu/ and the underlined consonants in Table 11.1 are present underlying, then the underlying forms of the numbers between 11 and 19 are plausibly /bd͡ʒu-gd͡ʒig/ 'eleven', /bd͡ʒu-bʃi/ 'fourteen', /bd͡ʒu-ŋa/ 'fifteen', and /bd͡ʒu-rgu/ 'nineteen.' From this, one can deduce that the underlying forms for the numbers less than ten are /gd͡ʒig/ 'one,' /bʃi/ 'four,' /ŋa/ 'five,' and /rgu/ 'nine.' Furthermore, the surface variation among numerals and their roots in compound numerals can be explained by the same word-initial cluster simplification process already introduced. Hence /bʃi/ surfaces as [ʃi] 'four.' This analysis accurately predicts the absence of word-initial consonant clusters throughout the data, as well as the entire pattern of consonant∼zero alternations seen in compound numerals. Because this explanation is both accurate and economical (in the sense that it only requires one phonological process to predict all observed variation in surface forms), this analysis is preferred over the ones previously explored.

The theory of prosodic licensing (Ito, 1986; Itô, 1989) provides a reason why initial consonant clusters may be simplified. Under this theory, extra-syllabic segments fail to surface. In other words, segments which fail to be

incorporated into syllables are deleted, a process known as *stray erasure* since the "stray" consonants are erased (McCarthy, 1979; Steriade, 1982; Harris, 1983).

In what follows, we provide a computational formalization of these generalizations. In particular, the licensing formula provides a straightforward way to implement the theory of prosodic licensing.

11.2 Representations

Underlying representations are analyzed in terms of a word model which contains the binary successor relation ($\triangleleft$) for order with a standard set of phonological features $\mathcal{F}$ represented as unary relations. In this analysis, we refer specifically to the features `cons` for 'consonantal' and `voc` for 'vocalic.' We do not specify other features in $\mathcal{F}$ and instead assume they are present and sufficiently distinguish the transcribed speech sounds in Tibetan.

Per standard phonological theory, syllabic structure is only present in surface forms, and not in underlying forms. Therefore, the surface representations are analyzed with a word model that includes —like the model for underlying forms—the binary successor relation ($\triangleleft$) for order and a standard set of phonological features $\mathcal{F}$, but also includes—unlike the model for underlying forms—unary relations describing syllabic roles such as `onset`, `coda`, and `nucleus`.

The signatures of the underlying and surface models are shown in (11.1) and (11.2), respectively.

$$\mathfrak{U} = \{\triangleleft\} \cup \mathcal{F} \tag{11.1}$$

$$\mathfrak{S} = \{\triangleleft, \texttt{onset}, \texttt{coda}, \texttt{nucleus}\} \cup \mathcal{F} \tag{11.2}$$

Next we specify the logical transduction that changes $\mathfrak{U}$-structures (underlying representations) to $\mathfrak{S}$-structures (surface representations).

11.3 Transformations

This section presents the ingredients necessary to define a logical transduction for consonant cluster reduction in Tibetan. The formulas for determining the output structures are defined in terms of the logical language FO($\mathfrak{U}$).

Specifically, this analysis enforces deletion of unsyllabified consonants through the licensing formula, providing a faithful implementation to the theory of prosodic licensing.

The domain of the transformation is any $\mathfrak{U}$-structure for any underlying form.

$$\varphi_{dom} \stackrel{\text{def}}{=} \texttt{true} \tag{11.3}$$

For example, Figure 11.1 schematizes a valid input given by /gd͡ʒig/ 'one.'

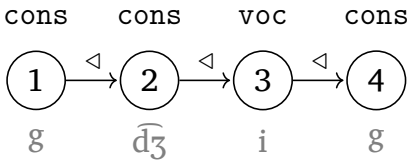


Figure 11.1: A visualization of the $\mathfrak{U}$-structure representing /gd͡ʒig/ 'one.' Only features `cons` and `voc` are shown.

Since the size of the domain of every output is no larger than the size of the input, the copy set contains a single element.

$$C \stackrel{\text{def}}{=} \{1\} \tag{11.4}$$

Consequently this means the size of the domain of the output is maximally the size of the input domain.

Next, Equation 11.5 defines the the binary successor relation in the output form.

$$\varphi_{\triangleleft}(x, y) \stackrel{\text{def}}{=} x \triangleleft y \tag{11.5}$$

Together the copy set and $\varphi_{\triangleleft}$ provide the structure shown in a 'workspace' in Figure 11.2.

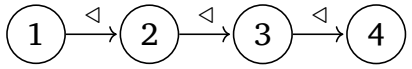


Figure 11.2: The binary relations that would be present in the output structure if every domain element is licensed.

Next, for each unary relation in the $\mathfrak{S}$-signature, we need a formula which identifies which elements have that property. Since no element

changes features, this is relatively straightforward for the relations that are also present in the $\mathfrak{U}$-signature.

$$(\forall\ \mathtt{f} \in \mathcal{F})\ \ \phi_{\mathtt{f}}(x) \stackrel{\text{def}}{=} \mathtt{f}(x) \tag{11.6}$$

Equation 11.6 is actually several equations that are all similar in character, which is why we bundle them together. Essentially it says that if an element x in the input structure belongs to a unary relation $\mathtt{f}$ then its corresponding element x also belongs to that unary relation in the output structure. In other words, elements in the surface form (if they exist) carry the same features they carried in the underlying form.

Of more interest are the unary relations which determine the syllabic roles segments will play. These are defined next.

$$\phi_{\mathtt{nucleus}}(x) \stackrel{\text{def}}{=} \mathtt{voc}(x) \tag{11.7}$$

$$\phi_{\mathtt{onset}}(x) \stackrel{\text{def}}{=} \mathtt{cons}(x) \wedge (\exists y)[\mathtt{voc}(y) \wedge x \lhd y] \tag{11.8}$$

$$\phi_{\mathtt{coda}}(x) \stackrel{\text{def}}{=} \neg\phi_{\mathtt{onset}}(x) \wedge \mathtt{cons}(x) \wedge (\exists y)[\mathtt{voc}(y) \wedge y \lhd x] \tag{11.9}$$

Essentially, these equations guarantee that vowels will be nuclei, consonants immediately preceding vowels will be onsets, and non-prevocalic consonants immediately succeeding vowels will be codas. Table 11.2 illustrates how these formulas are evaluated with respect to the representation of /gd͡ʒig/ 'one.'

	g	d͡ʒ	i	g
		x		
Formulas	1	2	3	4
cons(x)	⊤	⊤	⊥	⊤
voc(x)	⊥	⊥	⊤	⊥
$\phi_{\mathtt{nucleus}}(x)$	⊥	⊥	⊤	⊥
$\phi_{\mathtt{onset}}(x)$	⊥	⊤	⊥	⊥
$\phi_{\mathtt{coda}}(x)$	⊥	⊥	⊥	⊤

Table 11.2: Truth values the syllabic roles given the $\mathfrak{U}$-structure for /gd͡ʒig/.

The remaining formula to be defined, the licensing formula $\phi_{\mathtt{license}}(x)$, specifies which of the possible elements in the workspace are actually

realized in the surface structure. Here this is simply those elements that have syllabic roles.

$$\phi_{\texttt{license}}(x) \stackrel{\text{def}}{=} \quad \phi_{\texttt{nucleus}}(x) \vee \phi_{\texttt{onset}}(x) \vee \phi_{\texttt{coda}}(x) \tag{11.10}$$

Elements in the domain which are licensed are precisely those which have some syllabic role. Clearly, every element x except for $x = 1$ makes $\phi_{\texttt{license}}(x)$ true. Hence, in the running example, element 1 is not part of the final model.

Together, formulas 11.3 to 11.10 define a transduction that transforms $\mathfrak{U}$-structures (representations of underlying forms) to $\mathfrak{S}$-structures (representations of surface forms). There is a domain formula φ_{dom}, a copyset C, one formula of two free variables $\varphi_{\triangleleft}(x, y)$, formulae of one free variable for each unary relation in the $\mathfrak{S}$-signature, and finally one formula of one free variable $\phi_{\texttt{license}}(x)$ which determines which of the possible elements are actually realized.

Figure 11.3 illustrates the relational structure produced by the specified transformation when given the representation of the input /gd͡ʒig/ 'one' shown in Figure 11.1.

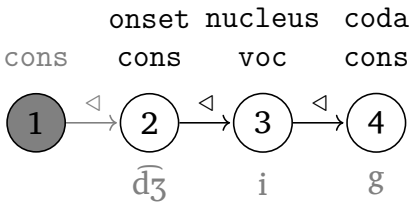


Figure 11.3: The relational structure produced by the input /d͡ʒig/ 'one' shown in Figure 11.1 under the transductions defined in Equations 11.3 through 11.10. Unlicensed elements and their associated relations are grayed out.

11.4 Conclusion

The prosodic theory of licensing is of course not the only way to analyze the alternations observed in Tibetan numerals. Another analysis could have forgone the syllabic roles and simply licensed all positions except

word-initial consonants immediately succeeded by other consonants. Such an analysis would also successfully account for the present data. These two analyses however are not extensionally equivalent: the prosodic licensing analysis predicts that if triple consonant clusters were to occur word-internally in underlying forms, the middle consonant would delete (since it would be the only one unsyllabified). However, this other proposal makes no such prediction.

More generally, this case study shows how aspects of phonological theories can be formalized and incorporated into logical transductions. First, on the representation side, we specifically chose to make syllabic roles parts of the output structure, and not parts of the input structure. Second, we used these syllabic roles to license the segments that surface faithfully.

Chapter 12

Autosegmental Representations in Zigula and Shambaa

ADAM JARDINE

This chapter provides computational analyses of two common patterns in tone, unbounded tone shift and unbounded tone spreading, using autosegmental representations (Goldsmith, 1976). These patterns, exemplified here by data from two Bantu languages, illustrate two common characteristics of tone (see Yip, 2002). Unbounded tone shift in Zigula (Kenstowicz and Kisseberth, 1990) exemplifies the 'mobility' of tonal units, or their ability to move long distances. Unbounded tone spreading, exemplified by data from Shambaa (Odden, 1982), illustrates the ability of a single tonal unit to be associated to multiple vowels. This chapter shows how MSO-definable transductions over autosegmental representations can insightfully capture these phenomena. In all examples in this chapter, the data have been converted to IPA from their original sources.

12.1 Zigula

We first turn to the distribution of tones in Zigula verbs (also known as Zigua or Chizigula; Kenstowicz and Kisseberth, 1990), and argue that the correct generalization is that a single underlying tone shifts to the penultimate vowel. Zigula verb roots come in two flavors: toned and toneless, as can be seen in the infinitive forms in Table 12.1. The following data are due to

Kenstowicz and Kisseberth (1990). Surface high tones are marked with an acute accent on the vowel ([á]); low-toned vowels are unmarked.

ku-gulus-a	'to chase'	ku-lombéz-a	'to ask'
ku-damaɲ-a	'to do'	ku-bindilíz-a	'to finish'
ku-songoloz-a	'to avoid	ku-hangalasáɲ-a	'to carry many things at once'

Table 12.1: Verb roots in Zigula

The verbs in the left column in Table 12.1 are pronounced entirely with a low tone, whereas the verbs in the right column all have a high tone on the penultimate vowel. As the affixes are the same, we must conclude that the roots on the right have an underlying high tone. However, there is a restriction on the position of the tone: roots where a high tone appears elsewhere in the infinitive form, such as the hypothetical *[ku-lómbez-a], are not attested.

Furthermore, in toned roots a single high tone appears on the penultimate vowel when the verb is extended to the right with toneless suffixes. Table 12.2 shows two forms from Table 12.1 extended with verbal suffixes [ez]/[iz] 'for' and [an] 'each other'.[1] The verb [ku-damaɲ-a] 'to do', which

ku-damaɲ-a	'to do'	ku-lombéz-a	'to ask'
ku-damaɲ-iz-a	'to do for'	ku-lombez-éz-a	'to ask for'
ku-damaɲ-iz-an-a	'to do for each other'	ku-lombez-ez-án-a	'to ask for each other'

Table 12.2: Suffixes in Zigula

is pronounced with all low tones in the plain infinitive, also shows no high tones in the suffixed forms [ku-damaɲ-iz-a] 'to do for' and [ku-damaɲ-iz-an-a] 'to do for each other'. In contrast, the verb [ku-lombéz-a] 'to ask', which has a high tone on the penultimate vowel in the plain infinitive, also has a single high tone on the penultimate vowel when the infinitive is suffixed: [ku-lombez-éz-a] 'to ask for' and [ku-lombez-ez-án-a] 'to ask for each other'. As these suffixes do not induce a tone for the toneless

[1] The [ez]/[iz] allomorphy is due to vowel harmony, and will not be analyzed here.

root [damaŋ] ‘do’, we can reasonably assume that they do not carry a tone underlyingly. Thus, the single high tone in the forms in the right column must originate from the root [lombez]/[lombéz] ‘ask’. However, this tone always appears on the penultimate vowel, regardless of whether it must shift from its underlying root to a suffix vowel.

That this generalization extends to high tones from other morphemes can be seen in toned prefixes. Table 12.3 gives data showing how the pronunciation of toneless roots changes depending on their prefix.

ku-gulus-a	‘to chase’	ku-songoloz-a	‘to avoid’
na-gulus-a	‘I am chasing’	na-songoloz-a	‘I am avoiding’
a-gulús-a	‘He/she is chasing’	a-songolóz-a	‘He/she is avoiding’

Table 12.3: Prefixes in Zigula

As seen in the second and third rows, finiteness and person are indicated by replacing the infinitive [ku] prefix with other prefixes, here [na] for the first person and [a] for the third person. As established in Table 12.1 and repeated here in Table 12.3, the roots [gulus] ‘chase’ and [songoloz] ‘to avoid’ are not pronounced with any tone in the infinitive. This is also true with the first person suffix: [na-gulus-a] ‘I am chasing’ and [na-songoloz-a] ‘I am avoiding’.

However, with the third person suffix, a high tone appears on the penultimate vowel: [a-gulús-a] ‘He/she is chasing’ and [a-songolóz-a] ‘He/she is avoiding’. We could analyze this as a complex morphological process, in which affixation of /a-/ ‘FINITE-3SG’ also induces a high tone in the penultimate vowel of the word. However, a more parsimonious explanation is that /a-/ ‘FINITE-3SG’, like toned verb roots, carries with it a high tone, which is then subject to the same penultimate-tone shift generalization as root tones. Of course, because affixes can carry tones, it is possible to have more than one underlying high tone in a word, and Kenstowicz and Kisseberth (1990) give a complete picture of the complex interactions which occur among multiple underlying tones. For expositional purposes, the discussion here will be restricted to forms with at most one underlying high tone.

The Zigula data under consideration are thus most generally explained by the following two propositions. One, morphemes may be toneless or

they may carry a high tone. Two, an underlying high tone shifts to the penultimate vowel.

Generalization 12.1. Penultimate shift. An underlying high tone shifts to the penultimate vowel in the word.

We can most directly express these generalizations with *autosegmental representations* (Goldsmith, 1976), in which different kinds of phonological units are arranged on different *tiers*, or distinct strings. Units on different tiers can be *associated* with one another. In the particular case of Zigula, we can posit that high tones exist on a tonal tier independent of the segments in the word, and that in the output of the phonology they are then associated to the penultimate vowel, as in the diagrams in Table 12.4. For example,

		H
Underlying Form	ku - gulus - a	ku - lomb e z - a
		H
Surface Form	ku - gulus - a	ku - lomb e z - a
	‘to chase’	‘to ask’
	H	H
Underlying Form	a - gul u s - a	ku - lombez - ez - a n - a
	H	H
Surface Form	a - gul u s - a	ku - lombez - ez - a n - a
	‘he/she is chasing’	‘to ask for each other’

Table 12.4: Autosegmental representations in Zigula

the underlying form of [ku-lombez-ez-án-a] ‘to ask for each other’ is thus /ku-lombez^{H}-ez-an-a/, to use the superscript H to indicate in-line that the root /lombez^{H}/ ‘ask’ contains an underlyingly unassociated H tone. As this

H resides on a tier by itself, it is able to move outside of its originating morpheme. Why, then, does it shift to the penultimate vowel? Kenstowicz and Kisseberth (1990) offer a metrical story: the final vowel is extrametrical, leaving the penultimate vowel in a metrically prominent position. This prominent position thus attracts the tone. I shall not dwell on this aspect of the analysis, except to later note how extrametricality can be referred to using MSO.

12.2 Shambaa

Autosegmental representations can similarly be invoked to insightfully account for the patterning of verbs in Shambaa (Odden, 1982). Verbs in Shambaa, like verbs in Zigula, can be either 'toned' or 'toneless'. However, in toned verbs in Shambaa, unlike those in Zigula, all vowels from the beginning of the root to the penult are pronounced with a high tone. Table 12.5 contrasts toneless verbs, illustrated with examples in the left column, with toned verbs on the right. That the correct generalization is

ku-ʃunt^{h}-a	'to wash'	ku-táɣ-a	'to buy'
ku-ɣoʃo-a	'to do'	ku-táhík-a	'to vomit'
ku-hand-a	'to plant'	ku-fúmbátíʃ-a	'to tie securely'

Table 12.5: Verb roots in Shambaa

that all vowels up to the penult are pronounced high, and not just all the vowels on the root, can be seen in suffixed forms. When affixed to toneless roots, the suffixes 'for each other' are pronounced [ij-an], with a low tone. When affixed to toned roots, they are pronounced [íj-án], with a high tone. The best explanation in difference in pronunciation of the tone in suffixes

ku-hand-a	'to plant'	ku-fúmbátíʃ-a	'to tie securely'
ku-hand-ij-an-a	'to plant for each other'	ku-fúmbátíʃ-íj-án-a	'to tie securely for each other'

Table 12.6: Suffixes in Shambaa

in [ku-hand-ij-an-a] 'to plant for each other' and [ku-fúmbátíʃ-án-a] is thus

that it due to the contrast between [ku-hand-a] ‘to plant’ and [ku-fúmbátíʃ-a] ‘to tie securely’—that is, the difference between toned and toneless verb roots.

Furthermore, like in Zigula, verb roots are not the only class of morpheme that can carry a tone. Table 12.7 shows how the pronunciation of toneless verb roots changes when affixed with the object marker prefixes [tʃí] ‘it’ and [ví] ‘them’.

Table 12.7: Prefixes in Shambaa

ku-ʃunt^{h}-a	‘to wash’
ku-tʃí-ʃúnt^{h}-a	‘to wash it’
ku-ɣoʃo-a	‘to do’
ku-ví-ɣóʃó-a	‘to do them’
ku-ɣoʃo-a-ɣoʃo-a	‘to do repeatedly’
ku-tʃí-ɣóʃó-á-ɣóʃó-a	‘to do it repeatedly’

When prefixed only with the infinitive prefix [ku], [ku-ʃunt^{h}-a] ‘to wash’ and [ku-ɣoʃo-a] ‘to do’ are pronounced with all low-toned vowels, as previously established. However, when prefixed with one of these object markers, the forms exhibit the familiar pattern of all high-toned vowels up to the penult, such as in [ku-ví-ɣóʃó-a] ‘to do them’. This is illustrated most dramatically in the contrast between the reduplicated form [ku-ɣoʃo-a-ɣoʃo-a] ‘to do repeatedly’ and the same form with an object prefix, [ku-tʃí-ɣóʃó-á-ɣóʃó-a] ‘to do it repeatedly’.

The generalizations in the Shambaa data are thus as follows. Like in Zigula, morphemes may be toneless or they may carry a high tone. In contrast with Zigula, however, this tone manifests on every vowel from its originating morpheme to the penultimate vowel in the word. Let us call this *unbounded spreading*.

Generalization 12.2. Unbounded spreading. An underlying high tone spreads to the penultimate vowel in the word.

Again, autosegmental representations present us with a direct way to express this generalization. Morphemes which carry a tone can be analyzed with a H tone underlyingly associated to their initial vowel. Unbounded spreading (12.2) then associates this H tone with all vowels up to the penult.

		H |
Underlying Form	ku - ɣoʃo - a	ku - fumbatiʃ - a
		H / \
Surface Form	ku - ɣoʃo - a	ku - fumbatiʃ - a
	‘to do’	‘to tie securely’
	H |	H |
Underlying Form	ku - vi - ɣoʃo - a	ku - fumbatiʃ- ij - an - a
	H / \	H / \
Surface Form	ku - vi - ɣoʃo - a	ku - fumbatiʃ- ij - an - a
	‘to do them’	‘to tie for each other’

Table 12.8: Autosegmental representations in Shambaa

This multiple association of a single tonal autosegment to multiple vowels directly captures the generalization that, for example, all of the high toned vowels in [ku-fúmbátíʃ-íj-án-a] ‘to tie for each other’ are the result of the single H tone underlyingly associated to the root /fúmbatiʃ/ ‘tie’ (where the accented /ú/ marks the underlying position of the H tone, as is usual).

12.3 Summary

We have now seen two patterns from tonal phonology which are directly captured through a change in autosegmental representations: penultimate tone shift in Zigula, in which an H tone is unassociated in the underlying form but associated to the penultimate vowel in the surface, and unbounded tone spread in Shambaa, in which a H tone is associated to a single vowel in

the underlying representation but associated to multiple vowels in the surface representation. The following shows how this change can be analyzed using MSO transductions over relational models.

12.4 Representations

A relational model for autosegmental representations is much like a string model, except that in addition to a relation indicating linear order there is a binary relation $\circ$ for association. Equation (12.1) gives such a model, where a standard set of segmental features $\mathcal{F}$ is augmented with a set T of tones.

$$\mathfrak{R} = \{\triangleleft, \circ\} \cup T \cup \mathcal{F} \tag{12.1}$$

For simplicity, the following examples consider a singleton set of tones $T = \{\text{H}\}$ and only the segmental features $\texttt{voc}, \texttt{cons} \in F$ for identifying vowels and consonants, respectively. Other features which distinguish the inventories of these languages are presumed and not otherwise specifically referenced.

Relational structures with the $\mathfrak{R}$-signature are general, and I wish to make clear which $\mathfrak{R}$-structures are valid autosegmental representations. In what follows, I provide a number of conditions which a $\mathfrak{R}$-structure must satisfy to count as an autosegmental representation.

First, tones cannot carry features and vice versa. This means that for all $\texttt{t} \in T$ and for all $\texttt{f} \in \mathcal{F}$, if x carries tone $\texttt{t}$ then x does not carry feature $\texttt{f}$, and vice versa. Formula 12.2 expresses this for a particular tone $\texttt{t}$ and feature $\texttt{f}$. The separation of all tonal properties from all featural properties is expressed in the logical language MSO($\mathfrak{R}$) in formula 12.3.

$$\texttt{sep}_{\texttt{f},\texttt{t}} \stackrel{\text{def}}{=} (\forall x)[(\texttt{f}(x) \rightarrow \neg\texttt{t}(x)) \wedge (\texttt{t}(x) \rightarrow \neg\texttt{f}(x))] \tag{12.2}$$

$$\texttt{sep} \stackrel{\text{def}}{=} \bigwedge_{\texttt{f}\in F,\texttt{t}\in T} \texttt{sep}_{\texttt{f},\texttt{t}} \tag{12.3}$$

Second, tones and segments appear on separate tiers. I define predicates which isolate tones from the segments. Formula 12.4 groups together all elements carrying some tone.

$$\texttt{tone}\,(x) \stackrel{\text{def}}{=} \bigvee_{\texttt{t}\in T} \texttt{t}(x) \tag{12.4}$$

Similarly, Formula 12.5 groups together all elements carrying segmental features.

$$\texttt{segment}\,(x) \stackrel{\text{def}}{=} \bigvee_{\texttt{f} \in F} \texttt{f}(x) \tag{12.5}$$

The predicate $\texttt{same\_tier}\,(x, y)$ is true if and only if x and y are both tones or both segments.

$$\begin{aligned}\texttt{same\_tier}\,(x, y) \stackrel{\text{def}}{=} \quad & (\,\texttt{tone}\,(x) \wedge \texttt{tone}\,(y)) \\ & \vee\,(\,\texttt{segment}\,(x) \wedge \texttt{segment}\,(y))\end{aligned} \tag{12.6}$$

Note that it would be straightforward to extend $\texttt{same\_tier}\,(x, y)$ to any finite number of such tier groupings. The following sentence then constrains tier structure such that only like units can appear on a tier together:

$$\texttt{good\_tiers} \stackrel{\text{def}}{=} (\forall x, y)[x \leq y \rightarrow \texttt{same\_tier}\,(x, y)] \tag{12.7}$$

The sentence $\texttt{good\_tiers}$ uses the predicate $\leq$, so it is important to be clear about it. It is defined in formula 12.8 , where $<$ is the transitive closure of successor defined in formula 2.14 on page 47.[2]

$$x \leq y \stackrel{\text{def}}{=} x = y \vee x < y \tag{12.8}$$

As such, $\texttt{good\_tiers}$ ensures that the tiers are homogeneous, and excludes structures in which, for example, a consonant precedes a tone.

Third, there must be a well-formed association between a tone and a *tone-bearing unit*, or TBU. In Zigula and Shambaa, the phonological generalizations referred to the tones of vowels, and not consonants. We thus specify that vowels are the TBUs.

$$\texttt{TBU}\,(x) \stackrel{\text{def}}{=} \texttt{voc}(x) \tag{12.9}$$

[2]This chapter uses successor as the primitive binary relation for order, and thus requires MSO logic to relate arbitrary elements under the general precedence relation. If general precedence was the primitive binary relation for order, FO logic could have been used.

It should be noted that, while this definition of TBU is satisfactory for the current discussion, other units have been proposed as TBUs, such as syllables and moras, and some researchers claim what counts as a TBU to be language specific (for in-depth discussion see Yip 2002). Note, however, that identifying a different TBU in the logical framework simply requires replacing $\texttt{voc}(x)$ in the definition for $\texttt{TBU}\,(x)$ with a different predicate.

Regardless of the definition of TBU, the following defines well-formed associations (WFAs) as those in which a tone is matched with a TBU:[3]

$$\texttt{WFA}\,(x,y) \stackrel{\text{def}}{=} \quad x \circ y \wedge \texttt{tone}\,(x) \wedge \texttt{TBU}\,(y) \tag{12.10}$$

Formula 12.11 then requires that all associations are of this type.

$$\texttt{good\_associations} \stackrel{\text{def}}{=} \quad (\forall x, y)[x \circ y \rightarrow \texttt{WFA}\,(x,y)] \tag{12.11}$$

For completeness, I also define one further constraint on well-formed autosegmental representations, although it is not relevant to the current discussion. The oft-posited No-Crossing Constraint (NCC; Goldsmith, 1976; Hammond, 1988; Coleman and Local, 1991) states that pairs of associated elements must precede each other; in other words, that association lines cannot cross. Formally,

$$\texttt{NCC} \stackrel{\text{def}}{=} \quad (\forall x_1, x_2, y_1, y_2)[(x_1 \circ x_2 \wedge y_1 \circ y_2 \wedge x_1 \leq y_1) \rightarrow x_2 \leq y_2] \tag{12.12}$$

This constraint is usually defined as a universal constraint on autosegmental representations (beginning with their initial definition in Goldsmith 1976). However, as the example tonal patterns given above only involve a single tone, the NCC will not come into play in the examples below. For a thorough discussion of the nature of the NCC, the reader is referred to Coleman and Local (1991).

[3]This defines association as inherently directional; that is, a tone is associated to a TBU, but not vice-versa. Association is often thought of as unordered (see, ex., Kornai, 1995). However, it will simplify the formulas in our transduction to treat them as directional, as any binary predicate referring to association between x and a y will only need to consider the case in which x is a tone and y is a TBU (and not vice versa). This choice has no effect on the function of the association relation.

At last, it is possible to say when an $\mathfrak{R}$-structures is an autosegmental representation (ASR).

$$\texttt{ASR} \stackrel{\text{def}}{=} \texttt{good\_tiers} \wedge \texttt{good\_associations} \wedge \texttt{NCC} \wedge \texttt{sep} \tag{12.13}$$

It is useful to be able to take an arbitrary $\mathfrak{R}$-structure and be able to decide if it is a valid autosegmental representation.

An example autosegmental representation that satisfies `ASR` is given in Figure 12.1. This model corresponds to the autosegmental representation of the underlying form of Shambaa /ku-ví-ɣoʃo-a/ 'to do them' (c.f. Table 12.8). (Note that morpheme boundaries are ignored.)

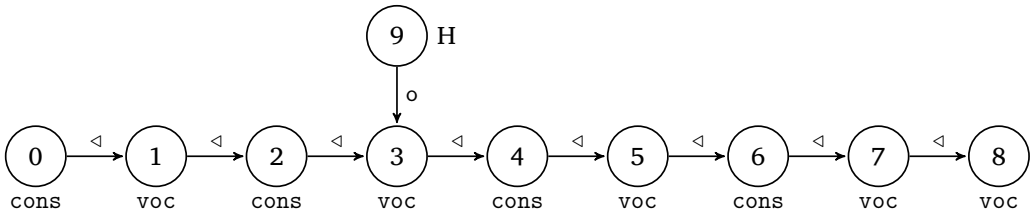


Figure 12.1: A graph representing the autosegmental model of Shambaa /ku-ví-ɣoʃo-a/ 'to do them'. Note both the edges representing successor $\triangleleft$ and association $\circ$.

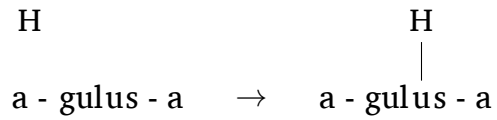
The graph in Figure 12.1 has the usual directed edges representing the successor relation $\triangleleft$ among segments. Additionally, there is an association $\circ$ between the domain element 9, which bears the H tone, and domain element 3, which corresponds to the second vowel.

It is interesting to ask where conditions expressed by `ASR` are in a theory of phonology. Are they universal constraints on surface well-formedness? Are they constraints on underlying well-formedness? In the latter cases, we could choose to refer to `ASR` in the domain formula of any logical transduction, to ensure that processes only apply to valid autosegmental representations. Another possibility is to assert that the conditions expressed by `ASR` are axiomatic and "outside" the theory per se. This last possibility may feel unsatisfying because one wonders whether there are any limits as to what kinds of conditions could be axiomatic. I cannot answer this deeper question at present, but it is worth pointing

out that the finiteness of $\mathfrak{R}$-structures, which is an implicit condition on their well-formedness, cannot be stated with MSO logic. In other words, attempts to limit conditions to the kinds of logical languages studied here will be insufficient to ensure that $\mathfrak{R}$-structures only contain finitely many elements. At present, I leave aside the important question of where conditions like `ASR` "live" in a theory of phonology, but acknowledge their utility for deciding whether a given $\mathfrak{R}$-structure is a valid autosegmental representation or not.

12.5 Transformations

We can then define transductions that operate on these representations. We begin with Zigula. Recall that in Zigula, a high tone shifts to the penultimate vowel in the word, as is illustrated for /a$^{\text{H}}$-gulus-a/$\rightarrow$[a-gulús-a] 'he/she is chasing' in Figure 12.2.



'he/she is chasing'

Figure 12.2: An example of high tone shift in Zigula

Following convention, I let the transduction apply to any $\mathfrak{R}$-structure.

$$\varphi_{dom} \stackrel{\text{def}}{=} \texttt{true} \tag{12.14}$$

As there is no epenthesis, the copy set is a singleton set: $C \stackrel{\text{def}}{=} \{1\}$. Furthermore, no domain element changes any feature or tone values, so all the unary relations can be defined as shown below.

$$\phi_{\texttt{f}}(x) \stackrel{\text{def}}{=} \texttt{f}(x) \quad (\forall \texttt{f} \in \mathcal{F}) \tag{12.15}$$

$$\phi_{\texttt{t}}(x) \stackrel{\text{def}}{=} \texttt{t}(x) \quad (\forall \texttt{t} \in T) \tag{12.16}$$

This is because, with autosegmental representations, the featural and tonal representations of the domain elements are not changing, but only the associations between them.

Finally, since the Zigula generalization does not involve deletion, the transformation preserves the successor relation.

$$\varphi_{\lhd}(x) \stackrel{\text{def}}{=} \quad x \lhd y \tag{12.17}$$

This leaves $\varphi_{\circ}(x,y)$, the predicate which determines when the copies of x and y are associated in the output structure. Informally, in Zigula, this is when x is a H tone and y is the penultimate vowel in the word. I must first, then, define predicates identifying the penultimate vowel. The predicate in (12.18) defines relative order $\lhd_V$ of vowels, ignoring consonants.

$$\begin{aligned} x \lhd_{\texttt{V}} y \stackrel{\text{def}}{=} \quad & \texttt{voc}(x) \wedge \texttt{voc}(y) \wedge x \leq y \\ & \wedge (\forall z)[(x \leq z \leq y) \rightarrow \neg \texttt{voc}(z)] \end{aligned} \tag{12.18}$$

In other words, $x \lhd_{\texttt{V}} y$ iff x precedes y and no other vowels intervene.

The final vowel can then be defined as the vowel for which no other vowel follows in the order $\lhd_{\texttt{V}}$.

$$\texttt{finalV}\,(x) \stackrel{\text{def}}{=} \quad \texttt{voc}(x) \wedge \neg(\exists y)[x \lhd_{\texttt{V}} y] \tag{12.19}$$

The penultimate vowel is then the vowel that precedes the final vowel with respect to $\lhd_{\texttt{V}}$.

$$\phi_{\texttt{penultV}}(x) \stackrel{\text{def}}{=} \quad (\exists y)[x \lhd_{\texttt{V}} y \wedge \texttt{finalV}\,(y)] \tag{12.20}$$

The final definition for the penultimate shift transduction is then simple: a H tone associates to the penultimate vowel.

$$\varphi_{\circ}(x,y) \stackrel{\text{def}}{=} \quad \texttt{H}(x) \wedge \texttt{penultV}\,(y) \tag{12.21}$$

To illustrate how this obtains the correct output autosegmental representation for penultimate shift in Zigula, Figure 12.3 gives the graphs for the autosegmental transformation in Figure 12.2 for the form /a$^{\text{H}}$-gulus-a/$\rightarrow$[a-gulús-a] ‘he/she is chasing’.

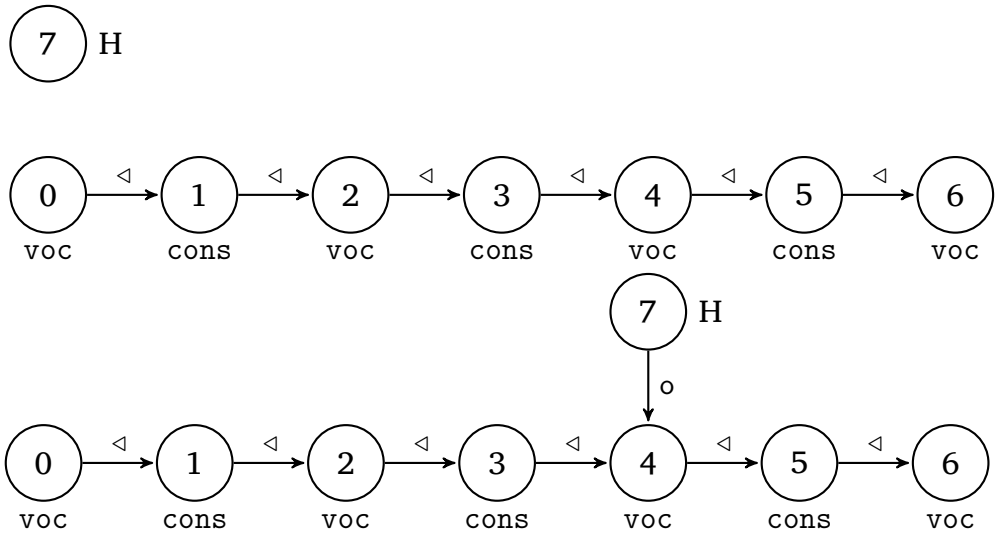


Figure 12.3: Visualization of the transduction of the autosegmental representations of Zigula /a$^{\text{H}}$-gulus-a/ $\rightarrow$ [a-gulús-a] 'he/she is chasing'.

All of the work in the transduction is done by $\varphi_{\circ}(x, y)$: it identifies elements 7 and 4 as the ones whose copies are to be associated in the output structure because 7 is a H tone and 4 is the penultimate vowel. In this way, the MSO transduction is entirely determined identifying by the well-formed associations in the output structure.

This is also the case for unbounded spreading in Shambaa. Recall that in Shambaa, an underlying H tone spreads up to the penultimate vowel, as Figure 12.4 recalls for /ku-ví-ɣoʃo-a/$\rightarrow$[ku-ví-ɣóʃó-a] 'to do them'.

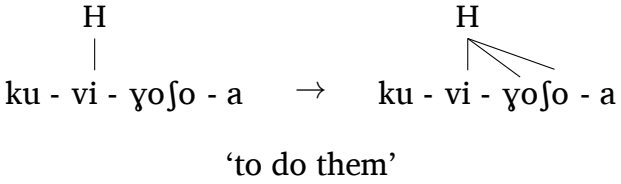


Figure 12.4: An example of unbounded spreading in Shambaa

The analysis for this as a MSO transduction is identical for that in Zigula, save for the predicate $\varphi_{\circ}(x, y)$. Instead, the predicate $\varphi_{\circ}(x, y)$ as defined below in (12.22) creates surface associations between the H tone and all vowels—save the extrametrical final vowel—to the right of the

vowel associated to this H in the input. Note that it uses several predicates defined previously for φ_{dom} and the transduction in Zigula.

$$\varphi_{\circ}(x,y) \stackrel{\text{def}}{=} \texttt{WFA}\,(x,y) \wedge (\exists x_1)[x \circ x_1 \wedge x_1 \leq y] \qquad (12.22)$$
$$\wedge\, (\exists y_2)[\,\texttt{penultV}\,(y_2) \wedge y \leq y_2]$$

The reference to $\texttt{WFA}\,(x,y)$ ensures that any surface association is well-formed (i.e. between a tone and TBU). The next part of the conjunct, $(\exists x_1)[x \circ x_1 \wedge x_1 \leq y]$, specifies that y must be equal to, or to the right of, the position x_1 to which x was originally associated. Finally, $(\exists y_2)[\,\texttt{penultV}\,(y_2) \wedge y \leq y_2]$ specifies that y is either equal to, or is to the left of, the penultimate vowel.

That this obtains the correct transduction is illustrated in Figure 12.5 corresponding to the mapping between the autosegmental representations for /ku-ví-ɣoʃo-a/→[ku-ví-ɣóʃó-a] 'to do them'.

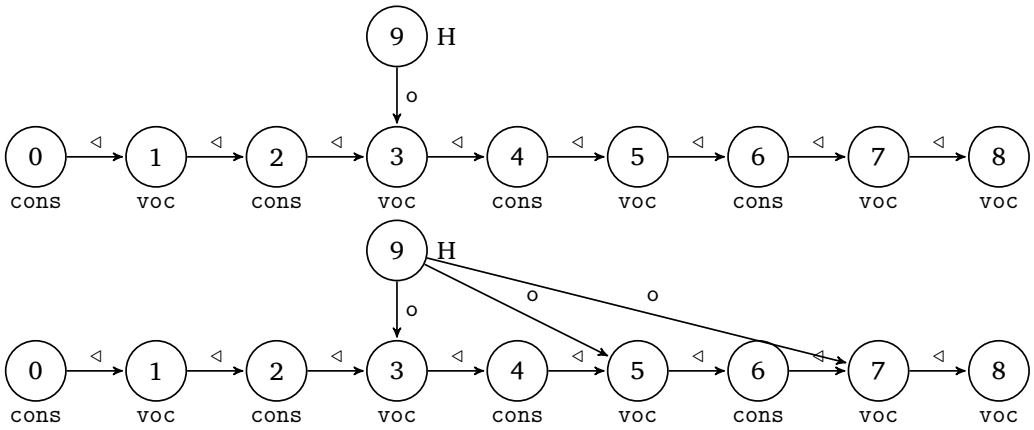


Figure 12.5: Visualization of the transduction of the autosegmental representation of Shambaa 'he/she is chasing'/ku-ví-ɣoʃo-a/→[ku-ví-ɣóʃó-a] 'to do them'.

The associations are built as follows. That domain element 9, as the only tone, satisfies the role of x in $\texttt{WFA}\,(x,y)$ is clear. The potential TBU nodes are 1, 3, 5, 7, and 8 (i.e., the voc positions). Out of these, only 3, 5, and 7 satisfy the role of y in the other conjunctions in $\varphi_{\circ}$. Position 3 satisfies the

other conditions on y as it is the domain element to which 9 is associated, and it is to the left of the penultimate vowel. So an association between 9 and 3 is drawn. Node 5 also satisfies these conditions because it is to the right of 3 but to the left of the penult 7; node 7 satisfies them because it is to the right of 3 and it is the penultimate vowel. So associations between 9 and 5 and between 9 and 7 are drawn. Associations are *not* drawn to nodes 1 and 8, because 1 is to the left of 3 (the originally associated vowel) and 8 is to the right of 7 (the penultimate vowel).

12.6 Discussion

Thus, in the analyses for both Zigula and Shambaa, the transduction is defined in terms of well-formed surface associations. In Zigula, this involved specifying that the only valid surface association is between a tone and the penultimate vowel. In Shambaa, this involved specifying a range of vowels in between the underlyingly associated vowel and the penult. In some ways, this is similar to Yip (2002)'s Optimality-Theoretic analyses for similar tonal processes, which also motivate the mapping between underlying and surface autosegmental structures through the well-formedness of surface associations. However, in OT, well-formedness is implemented through competition among individual constraints. For example, both spreading and shift to a penultimate vowel can be analyzed in OT through the interaction of a ALIGN-R constraint motivating the association of a H tone to as close to the word as possible with a NON-FINALITY constraint barring association to the final vowel. The difference between spreading and shift is whether or not a highly-ranked *ASSOC constraint bans the addition of new association lines. In contrast, in the MSO transductions defined in the present chapter, these conditions on the creation of surface associations are stated directly in the definition of $\varphi_\circ$.

There is one aspect of previous analyses we have not yet discussed with respect to MSO. Recall that Kenstowicz and Kisseberth (1990) explain the attraction of the tone to the penult via the extrametricality of the final vowel, whereas the MSO definitions above directly refer to the penultimate vowel. However, it is just as possible to create predicates which reference whether or not vowels are extrametrical. First, the following formula defines the 'metrical' vowels; i.e., those excluding the extrametrical final

vowel.

$$\texttt{metrical}\,(x) \stackrel{\text{def}}{=} \quad \texttt{voc}(x) \wedge \neg\, \texttt{finalV}\,(x) \tag{12.23}$$

Note that the above formula simply lists the conditions for being metrical—to implement other language-specific conditions for extrametricality, one only needs to add formula of the form $\neg\varphi(x)$ (where $\varphi(x)$ indicates the property that qualifies a unit as extrametrical).

We can then rewrite the final definition given in (12.21) for the Zigula penultimate shift transduction as the following: a H tone associates to the last metrical vowel.

$$\begin{aligned}\varphi_\circ(x,y) \stackrel{\text{def}}{=} & \texttt{H}(x) \wedge \texttt{metrical}\,(y) \\ & \wedge (\forall z)[\,\texttt{metrical}\,(z) \rightarrow \neg y \lhd_V z]\end{aligned} \tag{12.24}$$

The reader can confirm with Figure 12.3 that this has the same effect as the previous definition. A single association is drawn between domain elements 7 and 4 because 7 carries the H tone and 4 is the last vowel to satisfy $\texttt{metrical}\,(x)$.

We can similarly recruit $\texttt{metrical}\,(x)$ to define the surface association relation in Shambaa unbounded spreading. Recall that in Shambaa, the goal of the predicate $\phi_\circ(x,y)$ was to define the range of vowels to which the H tone should associate. We can recast the definition in (12.22) in metrical terms in the following way: the H tone associates to all metrical vowels to the right of the originally associated vowel:

$$\varphi_\circ(x,y) \stackrel{\text{def}}{=} \quad \texttt{WFA}\,(x,y) \wedge (\exists x_1)[x \circ x_1 \wedge x_1 \leq y] \wedge \texttt{metrical}\,(y)] \tag{12.25}$$

The definition for $\varphi_\circ$ in (12.25) differs only from that in (12.22) in the final conjunct. The original definition in (12.22) referred directly to the penult and all vowels to the left of it. In (12.25), however, we simply need only state that y is metrical. The reader can confirm that this defines the same transduction by referring back to Figure 12.5: nodes 3, 5, and 7 are the only metrical vowels to the right (or equal to) the originally associated node 3. Node 8, by the definition of $\phi_{\texttt{metrical}}(x)$, is extrametrical, and so is not associated.

12.7 Summary

To conclude, this chapter has reviewed two tonal phenomena, penultimate tone shift in Zigula and unbounded tone spreading in Shambaa, in which the correct generalizations referred not to the changing of features and segments, but to associations between units on independent tiers in autosegmental representations. Specifically, these autosegmental representations placed tonal units on a separate tier from segmental units.

These autosegmental representations were formalized as relational structures with an additional association relation. Constraints on well-formed autosegmental representations were defined using the logical language MSO($\triangleleft, T, \mathcal{F}$). The definitions provided scale to any other language, or to languages with multiple tiers.

Finally, we saw how MSO transductions could specify a change from underlying forms to surface forms by directly stating the conditions on association from tonal units to segmental units in surface forms. This chapter could not possibly cover the vast range of tonal phenomena that has been observed (see, e.g., Yip, 2002), but the analyses given here have shown how some fundamental properties of tone can be captured with MSO transductions over autosegmental representations.

Chapter 13

Compound Reduction in Signed Phonology

JONATHAN RAWSKI

Sign languages arise spontaneously in deaf communities, are acquired during childhood through normal exposure without instruction, and exhibit all of the facets and complexity found in spoken languages. Sandler and Lillo-Martin (2006) and Brentari (2019) provide groundbreaking overviews. Sign languages offer, as Sandler (1993) puts it, "a unique natural laboratory for testing theories of linguistic universals and of cognitive organization." They give insight into the concrete contents of grammatical form, and conditions on which aspects of grammar are amodal and which are tied to the modality.

13.1 Model-Theoretic Representations of Signs

The model-theoretic perspective gives us a flexible position from which to compare the representational content of spoken and signed phonological words. One obvious strategy is to say that that the representation for signs is essentially equivalent to that of spoken words, i.e. they are strings. Virtually all models of sign phonology allow sequentiality, as a sequence of static and dynamic segments (Liddell, 1984; Sandler, 1986; Liddell and Johnson, 1989; Perlmutter, 1993; Newkirk, 1998), or a sequence of abstract timing units where only the non-dynamic endpoints ultimately associate

(van der Hulst, 1993; Brentari, 1998).

While some sequential structure is acknowledged in almost all models of signs, sign representations have increasingly been argued to be inherently autosegmental in nature. Sandler (1986, 1989) demonstrated that hand configurations can be independent morphemes (classifiers), and exhibit autosegmental stability phonologically. She proposed the Hand Tier model, which represents sequential Location (L) and Movement (M) segments on a skeletal tier, allowing explicit reference to sequential information, and represents Handshape Configuration (H) autosegmentally, where one handshape characterizes the whole sign, in contrast to the one-per-segment in the Move-Hold model. Autosegments for place of articulation (P) features are a more recent innovation, seen in Brentari's (1998) Prosodic Model and in van der Hulst's Dependency Phonology Model (van der Hulst, 1993, 1994). An image from Sandler and Lillo-Martin (2006) of a monosyllabic sign in American Sign Language (ASL) and an autosegmental representation of it with an expanded feature-geometry for Place is in Figure 13.1.

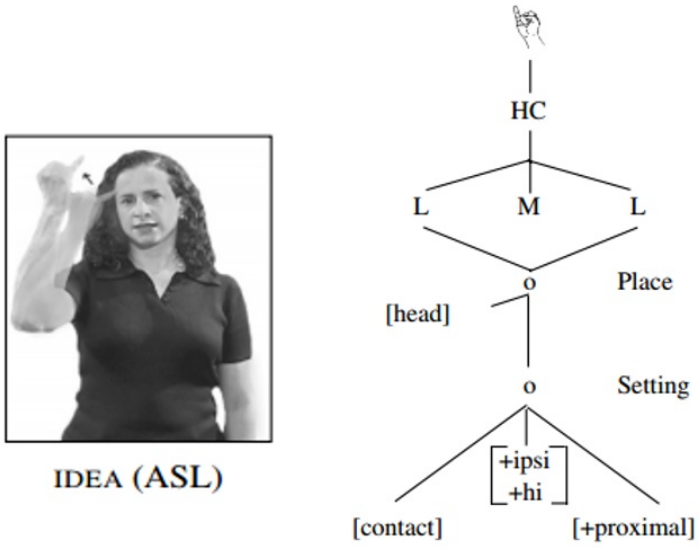


Figure 13.1: The ASL sign 'IDEA' (left) and its Hand Tier Model (right) (Images copyright Wendy Sandler & Diane Lillo-Martin).

The computational model presented here is not intended to be exhaustive nor definitive. It is simply sufficient to illustrate the applicability and

flexibility of model-theoretic representations for linguistic representation, regardless of modality. The model-theoretic relational structure for representing signs I present distinguishes three tiers: a skeletal tier for Location and Movement, a tier for Handshapes, and a tier for Places. I let L, M, H, and P denote nonempty finite sets of Locations, Movements, Handshapes, and Places, respectively. In what follows, I abstract away from particular locations, movements, handshapes, and places and treat them as individual units. They will be denoted with lowercase letters and subscripts, e.g. $\texttt{l}_1$, $\texttt{m}_1$, $\texttt{h}_1$, $\texttt{p}_1$. For concreteness, one can consider that Locations include unary relations such as `proximal` and `contact`; Movements include unary relations such as `path`; Handshapes include unary relations such as `open_B`; and Places include unary relations such as `head` and `non_dominant_hand`. An example is given in the bottom of Figure 13.3 on page 209. In the theory of sign language phonology, Handshapes and Places often have a richer structure. One could include such detailed feature geometries by, again, adding more relational structure.

Let $S = L \cup M$ denoting properties carried by elements on the skeletal tier. The skeletal tier is related to the handshape and location tiers via association relations (cf. Chapter 12). The relation `A` associates elements on the skeletal tier to the handshape tier, and the location relation `loc` relates elements on the skeletal tier to specific elements on the Place tier. While these are both association relations, they are included separately for clarity. The general precedence relation is also included. Altogether these yield the following relational signature.

$$\mathfrak{R} \stackrel{\text{def}}{=} \quad S \cup H \cup P \cup \{\texttt{A}, \texttt{loc}, <\} \tag{13.1}$$

As an example, a $\mathfrak{R}$-structure representing a monosyllabic sign with one place feature and one handshape feature is given in Figure 13.2.

As was the case with autosegmental representations (Chapter 12), not any $\mathfrak{R}$-structure is a valid sign. To be a valid sign, $\mathfrak{R}$-structures need to satisfy additional properties. These are generally analogous to the properties that Jardine expressed in Chapter 12 for autosegmental representations for tone: separability of the units into good tiers, well-formed associations, and no violation of the no-crossing constraint.

$$\texttt{sign\_ASR} \stackrel{\text{def}}{=} \texttt{separability} \wedge \texttt{good\_tiers} \wedge \texttt{good\_associations} \wedge \texttt{NCC} \tag{13.2}$$

$$
\begin{aligned}
\langle \mathcal{D} &= \{1,2,3,4,5\} \\
\mathtt{l}_1 &= \{1,3\} \\
\mathtt{m}_1 &= \{2\} \\
\mathtt{h}_1 &= \{4\} \\
\mathtt{p}_1 &= \{5\} \\
&\ldots \\
<= &\{\langle 1,2\rangle, \langle 1,3\rangle, \langle 2,3\rangle\} \\
\mathtt{A} &= \{\langle 1,4\rangle, \langle 2,4\rangle, \langle 3,4\rangle\} \\
\mathtt{loc} &= \{\langle 1,5\rangle, \langle 3,5\rangle\}\rangle
\end{aligned}
$$

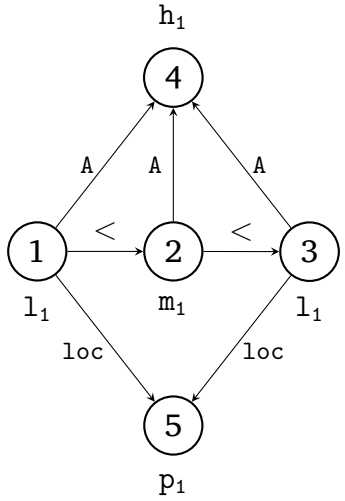


Figure 13.2: Autosegmental Hand Tier word model of a monosyllabic sign and a visualization. For all $n \neq 1$, unary relations $\mathtt{l}_n \in L$, $\mathtt{m}_n \in M$, $\mathtt{h}_n \in H$, and $\mathtt{p}_n \in P$ equal the empty set and are omitted for readability. Additionally, only some elements of the precedence relation are displayed in the visualization for readability.

For completeness, the logical formulas for these four properties are spelled out in an appendix to this chapter.

13.2 Compound Reduction

One salient property that emerges across sign languages concerns the boundedness of the sign. Many have argued for a syllable-like unit in sign languages, with movement corresponding to the syllable nucleus (Brentari, 1990; Wilbur, 1982, 2011; Sandler, 1989; Perlmutter, 1993). While internal movement resulting from a change in finger position or palm orientation may coincide with a path movement from one location to another, the simultaneous movements still constitute one syllable. Two movements in succession are counted as two syllables. This means that most monomorphemic words, and multimorphemic words are monosyllabic. This tendency, combined with the overwhelmingly nonconcatenative nature of sign morphology, has resulted in what some call a "monosyllable conspiracy" (Sandler and Lillo-Martin, 2006), with some using a CVC template as a heuristic comparison with the monosyllabic LML sequences (Perlmutter, 1993).

Also across sign languages, many lexicalized sign compounds undergo a type of phonological reduction to preserve the monosyllabic character of canonical signs (Frishberg, 1975). Compound reduction is an amalgam of several processes. Often, sequential segments of both members of the compound delete (Liddell, 1984; Liddell and Johnson, 1989), the hand configuration of the first member also deletes, and the hand configuration autosegment of the second member spreads to characterize the whole surface compound (Sandler 1986; 1989). Other compounds reduce in different ways. Some maintain all segments and both hand configurations. Others reduce segmental structure only, maintaining two hand configurations (though see Lepic (2015)).

As an example, consider the ASL compounds 'FAINT' ('MIND' + 'DROP') and 'BELIEVE' ('THINK' + 'MARRY') as well as the Israeli Sign Language compound 'SURPRISED' ('THINK' + 'STOP'). These compounds are characterized by the following processes:

1. regressive total handshape spreading from the second sign to the first,

2. deletion of the initial location segments of both signs and the first movement, and

3. coalescence of the signs such that place information is uniquely specified for both L segments.

These are shown in Figure 13.3, along with Sandler (1989)'s Hand Tier model analysis of the reduction of ASL 'BELIEVE'. These generalizations are the ones that will be formalized in the next section using first order logic over $\mathfrak{R}$-structures.

13.3 Compound Reduction as a Logical Transformation

This section presents a computational treatment of the compound reduction process illustrated in Figure 13.3 using the aforementioned $\mathfrak{R}$-structures. Recall that a logical transduction requires the following: a domain formula $\phi_{\texttt{dom}}$, a copy set C, one or more licensing formulas depending on the size of the copy set, formulas with two free variables for each binary relation in the model signature for output structures, and formulas of one free for each unary relation in the model signature for output structures. Each of these formulas is written in a logical language based on the model signature of the input structure. Here I use First Order logic. For compound reduction, both the relational structures for both inputs and outputs will be the same, and use the signature $\mathfrak{R}$ introduced above.

I assume the domain of the transformation is any $\mathfrak{R}$-structure.

$$\phi_{\texttt{dom}} \stackrel{\text{def}}{=} \texttt{true} \tag{13.3}$$

Since the size of the domain of every output is no larger than the size of the input we set the copy set to contain a single element.

$$C \stackrel{\text{def}}{=} \{1\} \tag{13.4}$$

This means the size of the domain of the output is maximally the size of the input domain. The reduction itself will primarily be handled by the licensing function defined later below.

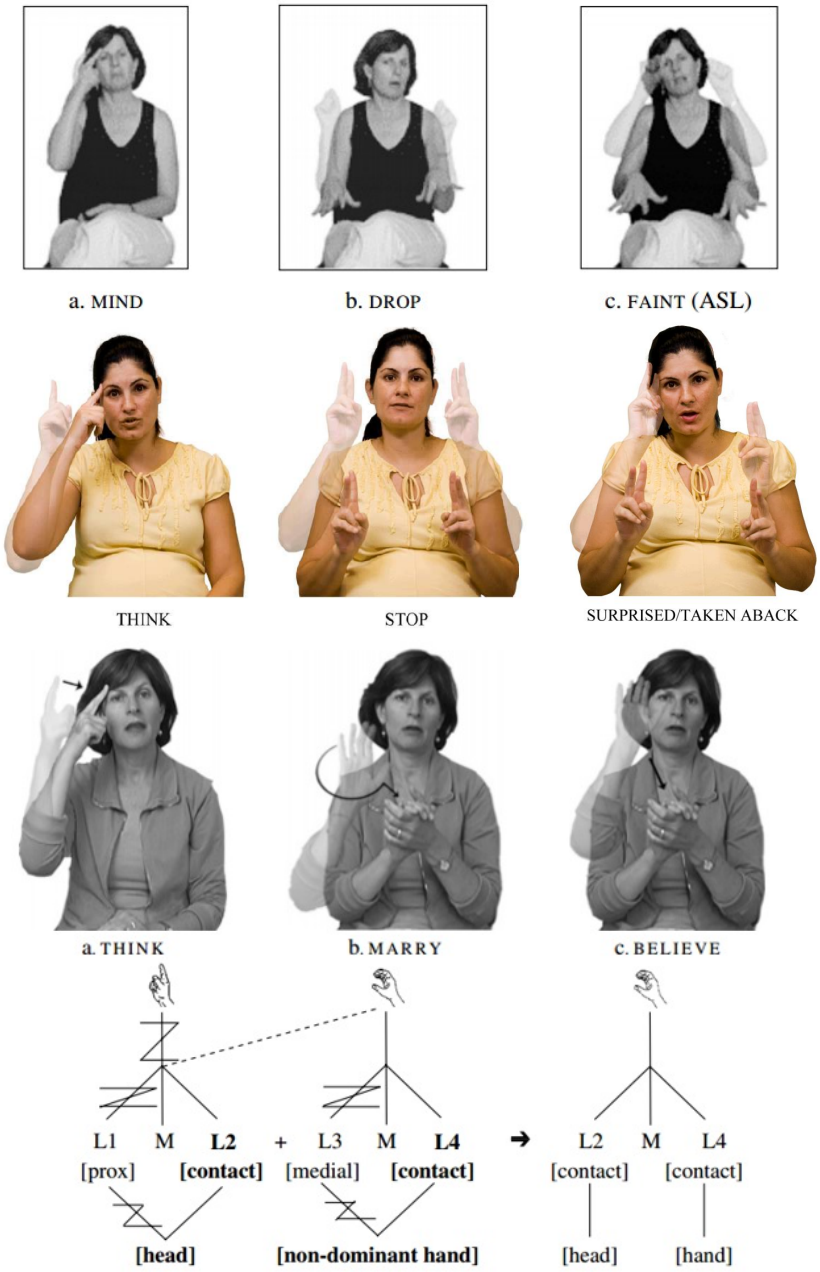


Figure 13.3: Compound Reduction. Top: ASL MIND + DROP = FAINT; Middle: ISL THINK + STOP = SURPRISED; Bottom: ASL THINK + MARRY = BELIEVE, with Hand Tier model of reduction (Images copyright Wendy Sandler & Diane Lillo-Martin).

Since elements of relations which include unlicensed components are excluded from the output structure (see discussion in Chapter 10), all of the unary relations in the output structure can be fixed to be the same as they are in the input structure (13.5-13.7). This amounts to something akin to an identity relation, enforcing faithfulness.

$$\phi_{\mathtt{s}}(x) \stackrel{\text{def}}{=} \mathtt{s}(x) \quad (\forall \mathtt{s} \in S) \tag{13.5}$$

$$\phi_{\mathtt{h}}(x) \stackrel{\text{def}}{=} \mathtt{h}(x) \quad (\forall \mathtt{h} \in H) \tag{13.6}$$

$$\phi_{\mathtt{p}}(x) \stackrel{\text{def}}{=} \mathtt{p}(x) \quad (\forall \mathtt{p} \in P) \tag{13.7}$$

Similarly, the binary relations for precedence and the location association relation also can be preserved, since the only change to these relations is the exclusion of those pairs which contain unlicensed elements.

$$x < y \stackrel{\text{def}}{=} x < y \tag{13.8}$$

$$\phi_{\mathtt{loc}}(x, y) \stackrel{\text{def}}{=} \mathtt{loc}(x, y) \tag{13.9}$$

The next part of the mapping concerns handshape spreading, which is handled by the association relation `A`. It will be useful to have predicates identifying which tier an element of the domain belongs to, as well as ones identifying particular positions in a sign. Formulas 13.10, 13.11, and 13.12 pick out elements on the skeletal, Handshape, and Place tiers, respectively.

$$\mathtt{S}(x) \stackrel{\text{def}}{=} \bigvee_{\mathtt{s} \in S} \mathtt{s}(x) \tag{13.10}$$

$$\mathtt{H}(x) \stackrel{\text{def}}{=} \bigvee_{\mathtt{h} \in H} \mathtt{h}(x) \tag{13.11}$$

$$\mathtt{P}(x) \stackrel{\text{def}}{=} \bigvee_{\mathtt{p} \in P} \mathtt{p}(x) \tag{13.12}$$

Predicates which pick out certain privileged positions in a sign are provided below. These use the successor relation ($\lhd$), which recall from Chapter 2

(see formula 2.21 on page 53), can be defined with FO($<$).

$$\texttt{first}\,(x) \stackrel{\text{def}}{=} \neg(\exists y)[y \lhd x] \tag{13.13}$$

$$\texttt{last}\,(x) \stackrel{\text{def}}{=} \neg(\exists y)[x \lhd y] \tag{13.14}$$

$$\texttt{third}\,(x) \stackrel{\text{def}}{=} (\exists y, z)[\,\texttt{first}\,(z) \wedge z \lhd y \lhd x] \tag{13.15}$$

$$\texttt{penult}\,(x) \stackrel{\text{def}}{=} (\exists y)[\,\texttt{last}\,(y) \wedge x \lhd y)] \tag{13.16}$$

Now formula 13.17 states that every element on the skeletal tier is associated to the final handshape.

$$\phi_{\texttt{A}}(x, y) \stackrel{\text{def}}{=} \texttt{S}\,(x) \wedge \texttt{H}\,(y) \wedge \texttt{last}\,(y) \tag{13.17}$$

The deletion of timing and handshape segments is handled by the licensing function $\phi_{\texttt{lic}}(x)$, which specifies which potential domain elements are actually present in the output structure. I specify licensing functions for each tier, and then $\phi_{\texttt{lic}}$ itself. Formula 13.18 says that elements on the skeletal tier are licensed if they are third, penultimate, or final element. These elements help capture the coalescence of the reduced compound. Formula 13.19 says that only the final handshape of the compound is licensed. Formula 13.20 says elements on the place tier are licensed. Formula 13.21 says that in general, an element is licensed only if it satisfies one of these conditions.

$$\texttt{lic}_{\texttt{S}}\,(x) \stackrel{\text{def}}{=} \texttt{S}\,(x) \wedge (\,\texttt{third}\,(x) \vee \texttt{penult}\,(x) \vee \texttt{last}\,(x)) \tag{13.18}$$

$$\texttt{lic}_{\texttt{H}}\,(x) \stackrel{\text{def}}{=} \texttt{H}\,(x) \wedge \texttt{last}\,(x) \tag{13.19}$$

$$\texttt{lic}_{\texttt{P}}\,(x) \stackrel{\text{def}}{=} \texttt{P}\,(x) \tag{13.20}$$

$$\phi_{\texttt{lic}}(x) \stackrel{\text{def}}{=} \texttt{lic}_{\texttt{S}}\,(x) \vee \texttt{lic}_{\texttt{H}}\,(x) \vee \texttt{lic}_{\texttt{P}}\,(x) \tag{13.21}$$

To illustrate how this logical specification works, I show how it applies to the mapping to the model of the ASL compound BELIEVE (THINK + MARRY) under the Hand tier model signature $\mathfrak{R}$, which is shown in Figure 13.3. In this figure, the unary relations $\texttt{h}_1$ and $\texttt{h}_2$ are arbitrary Handshapes in H. Similarly, $\texttt{p}_1$ and $\texttt{p}_2$ are arbitrary Places in P. Table 13.1 shows how the predicates relevant to the license formula are evaluated.

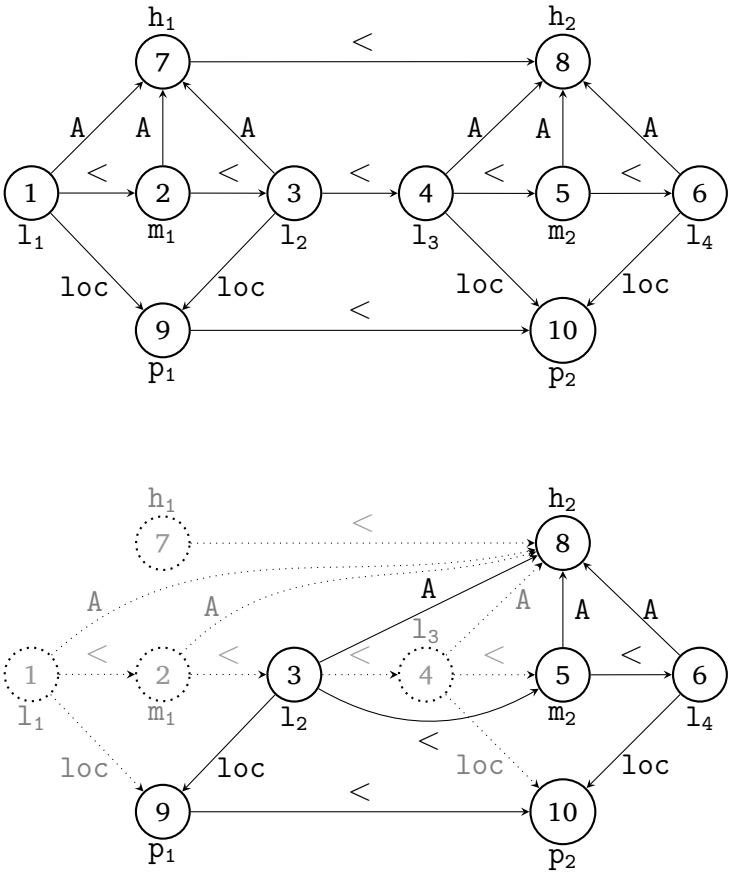


Figure 13.4: Visual of Input (top) and Output (bottom) $\mathfrak{R}$-structures for compound reduction of ASL BELIEVE (THINK + MARRY). Unlicensed elements and their associated relational elements are dashed and in gray. Only some elements of the precedence relation are shown for readability.

	x									
Formulas	1	2	3	4	5	6	7	8	9	10
$\texttt{S}\,(x)$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$
$\texttt{H}\,(x)$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\bot$	$\bot$
$\texttt{P}\,(x)$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$
$\texttt{third}\,(x)$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\texttt{penult}\,(x)$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\top$	$\bot$
$\texttt{last}\,(x)$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\top$
$\texttt{lic}_{\texttt{S}}\,(x)$	$\bot$	$\bot$	$\top$	$\bot$	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$
$\texttt{lic}_{\texttt{H}}\,(x)$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$
$\texttt{lic}_{\texttt{P}}\,(x)$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$
$\phi_{\texttt{lic}}(x)$	$\bot$	$\bot$	$\top$	$\bot$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$

Table 13.1: Evaluating the licensing function for compound reduction.

13.4 Discussion

The logical transduction gets its power by factoring the process into its necessary parts, which are provided by the signature of the relational model. For example, in this transduction, the fact that handshape spreading and segment deletion are independent is captured via the association relation in the output structure $\phi_{\texttt{A}}$ on the licensing formula $\phi_{\texttt{lic}}$, which picks out different segments on the skeletal and Handshape tier are not deleted. This factorization allows one to capture the fact that in compound reduction cross-linguistically, these parts of the process can vary.

For example, in the ASL compound GOOD-NIGHT (GOOD + NIGHT) (Figure 13.5), the second handshape still spreads, but it is the first L of the first member of the compound and the second L of the second member of the compound which survive. Handling reductions of this flavor involves merely changing the licensing function, while the rest of the transduction remains unchanged. Another example: hand configuration assimilation may take place regardless of whether or not the compound loses

Figure 13.5: ASL GOOD (left), NIGHT (middle), and the compound GOOD-NIGHT (right). Illustration from Liddell and Johnson (1986)

segments. Total hand configuration assimilation occurs on the reduced monosyllabic ASL compound HUSBAND (MAN + MARRY) and on the unreduced disyllabic compound OVERSLEEP (SLEEP + SUNRISE) (Liddell and Johnson, 1986).

Finally, while there are many more types of compound reduction, involving partial handshape assimilation, and/or deletion of different elements, I have argued that these and other sign language processes inhabit the same complexity class as phonological processes in spoken language (Rawski, 2017); in particular, the Input Strictly Local class (Chandlee and Heinz, 2018). Chandlee and Lindell in Chapter 22 provide a logical characterization of this class and Strother-Garcia (2018a) and Chandlee and Jardine (2019a) use this logical characterization to investigate the computational complexity of nonlinear representations in spoken language phonology such as syllable structure and autosegmental representations, respectively. Conducting a similar analysis for sign language phonology is an important area of future research.

13.5 Conclusion

This chapter shows how logical and model-theoretic methods can fruitfully be applied to the study of sign language phonology. Many signed languages include processes which reduce compounds to single "syllables." One such process in ASL was analyzed using first order logic over representations recognizing three distinct tiers which related properties designating the location, movement, handshape, and place of manual signs. Future work can further investigate and articulate the rich representational structures

that have been posited (see (Brentari, 2019) for a recent review), as well as more closely examine the kinds of logical languages necessary and sufficient for describing all aspects of sign language phonology.

Appendix

This appendix defines the four predicates `separability`, `good_tiers`, `good_associations`, `NCC` identifying those $\mathfrak{R}$-structures which represent signs.

$$\begin{aligned}\texttt{sep}_{\texttt{s,h,p}} \stackrel{\text{def}}{=} \quad & (\forall x)[(\texttt{s}(x) \rightarrow (\neg\texttt{h}(x) \wedge \neg\texttt{p}(x))) \\ & \wedge (\texttt{h}(x) \rightarrow (\neg\texttt{s}(x) \wedge \neg\texttt{p}(x))) \\ & \wedge (\texttt{p}(x) \rightarrow (\neg\texttt{h}(x) \wedge \neg\texttt{s}(x)))]\end{aligned} \tag{13.22}$$

$$\texttt{separability} \stackrel{\text{def}}{=} \bigwedge_{\texttt{s}\in S,\ \texttt{h}\in H,\ \texttt{p}\in P} \texttt{sep}_{s,h,p} \tag{13.23}$$

$$\begin{aligned}\texttt{same\_tier}\,(x,y) \stackrel{\text{def}}{=} \quad & (\,\texttt{S}\,(x) \wedge \,\texttt{S}\,(y)) \\ & \vee (\,\texttt{H}\,(x) \wedge \,\texttt{H}\,(y)) \\ & \vee (\,\texttt{P}\,(x) \wedge \,\texttt{P}\,(y))\end{aligned} \tag{13.24}$$

$$\texttt{good\_tiers} \stackrel{\text{def}}{=} (\forall x, y)[x < y \rightarrow \texttt{same\_tier}\,(x,y)] \tag{13.25}$$

$$\begin{aligned}\texttt{WFA}\,(x,y) \stackrel{\text{def}}{=} \quad & (\texttt{A}(x,y) \wedge \,\texttt{S}\,(x) \wedge \,\texttt{H}\,(y)) \\ & \vee (\texttt{loc}(x,y) \wedge \,\texttt{S}\,(x) \wedge \,\texttt{P}\,(y))\end{aligned} \tag{13.26}$$

$$\texttt{good\_associations} \stackrel{\text{def}}{=} (\forall x, y)[(\texttt{A}(x, y) \vee \texttt{loc}(x, y)) \rightarrow \texttt{WFA}\,(x, y)] \tag{13.27}$$

$$x \leq y \stackrel{\text{def}}{=} x = y \vee x < y \tag{13.28}$$

$$\texttt{NCC} \stackrel{\text{def}}{=} (\forall x_1, x_2, y_1, y_2) \bigwedge_{R \in \{\texttt{A},\texttt{loc}\}} [(R(x_1, x_2) \wedge R(y_1, y_2) \wedge x_1 \leq y_1) \rightarrow x_2 \leq y_2] \tag{13.29}$$

Chapter 14

Focus and Verb Movement in Hungarian

Mai Ha Vu

This chapter gives a logical formalization of left periphery movement in Hungarian, specifically of focus and verb movement. In this way, this chapter shows that the methods presented earlier in the context of phonological analysis can also be applied to other linguistics domains as well.

There are several proposals accounting for Hungarian word order that differ from each other in details such as the underlying tree structure and the specific functional projections involved (Brody, 1990; Puskás, 2000; É. Kiss, 2002). In order to give a simple illustration of formalizing syntactic processes with logical transductions, I adopt the early account by Brody (1990) that argues for focus movement to specifier of FocusP, and verb movement to Focus^0.

The current analysis assumes familiarity with traditional concepts of generative syntax, such as functional projections, as well as phrasal and head movement. Specifically, I follow Brody's (1990) analysis in assuming X-bar theoretic syntactic structures (Stowell, 1981). The logical transduction maps from an underlying base-generated tree to a surface tree. The base-generated tree is unorthodox in that it already contains all nodes of the final tree derivation, and so no new node is built as the result of movement. The graph transduction provided in this chapter thus solely illustrates the displacement of lexical items from one tree node to another, and not other parts of the derivation. Nevertheless, this simple illustration

suffices to show that the logical formalization method itself ought to be applicable to a wider array of other syntactic processes, structures and assumptions.

The chapter is organized as follows. Section 14.1 gives the syntactic analysis to Hungarian focus and verb movement. Section 14.2 provides a brief introduction to tree models. Section 14.3 formally describes left periphery movement in Hungarian on the tree models defined in Section 14.2. Section 14.4 concludes.

14.1 Syntactic analysis

The consensus in the Hungarian syntax literature is that Hungarian word order is determined by information structure (Horvath, 1976; É. Kiss, 1981), specifically that the word order is Topic-Focus-Verb in the left periphery as shown in (14.1), where focused elements are indicated with capital letters. To account for this order, topic and focus are assumed to be in their relevant functional projections, TopicP and FocusP. While there are disagreements on whether these functional projections are to be found in the CP or IP (É. Kiss, 2002; Puskás, 2000), the current analysis does not hinge on this decision. I follow the position of É. Kiss (2002) and Brody (1990) that FocusP immediately scopes over the VP.

(1) [$_{\text{Topic}}$ János] [$_{\text{Focus}}$ ”MARIT] szereti.
John.NOM MARY.ACC loves
‘John loves MARY.’

There is also a general agreement that the Topic-Focus-Verb order is derived by movement, rather than base-generated by phrase-structure rules. The evidence is that all information structure is optional in Hungarian. In the absence of topic or focus, sentences are verb initial as shown in (2). For the rest of this chapter, I restrict my discussion to the focus and verb.

(2) Szereti János Marit.
loves John.NOM Mary.ACC
‘John loves Mary.’

According to Brody (1990), the focused constituent moves to Spec-FocusP, and the verb moves to Focus^0 as shown in Figure 14.1. Verb movement is evident from the position of the verbal particle in relation

to the verb in the presence of focus. In a sentence containing focus, the particle is always post-verbal (3b), while in sentences without focus, the particle is pre-verbal (3a). This fact indicates that the verb moves to a higher position than the particle when focus is present.

(3) a. Kati fel olvasta a verseket.
Kati.NOM PRT read the poems.ACC
'Kate read the poems aloud.'

b. A VERSEKET olvasta fel Kati.
the POEMS.ACC read PRT Kati.NOM
'It was the POEMS that Kate read aloud.'

c. * A VERSEKET fel olvasta Kati.
the POEMS.ACC PRT read Kati.NOM
'It was the POEMS that Kate read aloud.'

Taken altogether, I conclude the tree transformation illustrated in Figure 14.1 for the derivation of (4) is the same as the one without the verbal particle (3b). I follow É. Kiss (2002) in assuming that the Hungarian VP has a flat structure. Note that the input to the transduction consists of a tree which has already been assembled, and the transduction presented in the next section only explicates the movement operations.

(4) VERSEKET olvasott Kati.
POEMS.ACC read.PST Kate.NOM
'It was the POEMS that Kate read.'

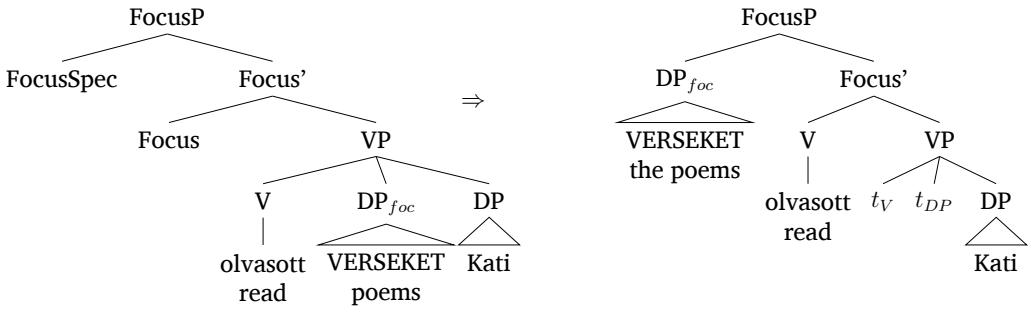


Figure 14.1: Base tree and derived tree illustrating focus and verb movement in Hungarian.

14.2 Representations

To formally define syntactic structures and processes, I use the model-theoretic signature $\mathfrak{T}$ in Equation 14.1. Its main difference from the model-theoretic representations of strings used for most phonological processes is that it contains two binary relations, one for dominance (δ) and one for successor ($\triangleleft$), and a set of syntactic features $\mathcal{F}$, which are unary relations.

$$\mathfrak{T} = \{\delta, \triangleleft\} \cup \mathcal{F} \tag{14.1}$$

Readers are referred to Rogers (2003) for technical background, and to Frank and Vijay-Shanker (2001) for a model-theoretic structures for syntax which uses c-command as a primitive relation in the signature.

The relevant syntactic features are listed in (5). Each domain element in a $\mathfrak{T}$-structure can have multiple features. Structural features indicate the role a node in a tree plays with respect to its X-bar structure, whether they are a head, phrase, bar, or specifier. Category features correspond to familiar syntactic categories, such as verb, determiner, and focus. Lastly, nodes can also bear properties indicating whether they are traces and whether they are focused.

(5) a. **Structural features**: `phrase`, `head`, `bar`, `specifier`

b. **Category features**: `Focus`, `V`, `D`

c. **Other**: `trace`, `focused`

The features listed in (5) are the unary relations in $\mathcal{F}$ that I will be using.

Below I define a number of predicates corresponding to the node labels of the trees in Figure 14.1. Figure 14.2 visualizes the $\mathfrak{T}$-structure corresponding to the input tree in Figure 14.1. Each node is a domain element indicated with an integer and labeled with the shorthand labels listed in

(14.2). Dominance relations are shown as edges between the nodes.

$$\texttt{FP}\ (x) \stackrel{\text{def}}{=} \texttt{Focus}(x) \wedge \texttt{phrase}(x) \tag{14.2}$$

$$\texttt{FS}\ (x) \stackrel{\text{def}}{=} \texttt{Focus}(x) \wedge \texttt{specifier}(x) \tag{14.3}$$

$$\overline{\texttt{F}}\ (x) \stackrel{\text{def}}{=} \texttt{Focus}(x) \wedge \texttt{bar}(x) \tag{14.4}$$

$$\texttt{FH}\ (x) \stackrel{\text{def}}{=} \texttt{Focus}(x) \wedge \texttt{head}(x) \tag{14.5}$$

$$\texttt{VP}\ (x) \stackrel{\text{def}}{=} \texttt{V}(x) \wedge \texttt{phrase}(x) \tag{14.6}$$

$$\texttt{VH}\ (x) \stackrel{\text{def}}{=} \texttt{V}(x) \wedge \texttt{head}(x) \tag{14.7}$$

$$\texttt{DP}\ (x) \stackrel{\text{def}}{=} \texttt{D}(x) \wedge \texttt{phrase}(x) \tag{14.8}$$

$$\texttt{DP}_{\texttt{foc}}\ (x) \stackrel{\text{def}}{=} \texttt{DP}(x) \wedge \texttt{focused}(x) \tag{14.9}$$

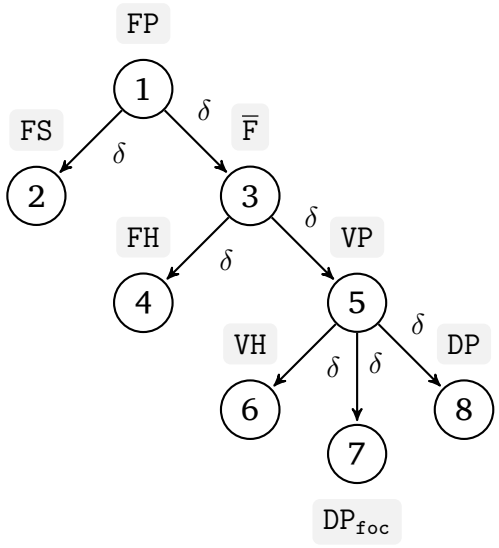


Figure 14.2: Model of input tree in Figure 14.1. The successor relation among siblings is omitted for readability.

Similarly, the model-theoretic representation of the derived tree in Figure 14.1 can be given in terms of the primitive relations in the $\mathfrak{T}$-signature. As above, I define additional predicates for the conjunctions of

features that are relevant for this structure.

$$\texttt{DP}_{\texttt{foc}}(x) \stackrel{\text{def}}{=} \texttt{DP}(x) \wedge \texttt{specifier}(x) \wedge \texttt{Focus}(x) \wedge \texttt{focused}(x) \quad (14.10)$$

$$\texttt{V}_{\texttt{foc}}(x) \stackrel{\text{def}}{=} \texttt{VH}(x) \wedge \texttt{Focus}(x) \quad (14.11)$$

$$\texttt{t}_{\texttt{V}}(x) \stackrel{\text{def}}{=} \texttt{VH}(x) \wedge \texttt{trace}(x) \quad (14.12)$$

$$\texttt{t}_{\texttt{DP}}(x) \stackrel{\text{def}}{=} \texttt{DP}(x) \wedge \texttt{focused}(x) \wedge \texttt{trace}(x) \quad (14.13)$$

Figure 14.3 shows the $\mathfrak{T}$-structure corresponding to the derived tree in Figure 14.1.

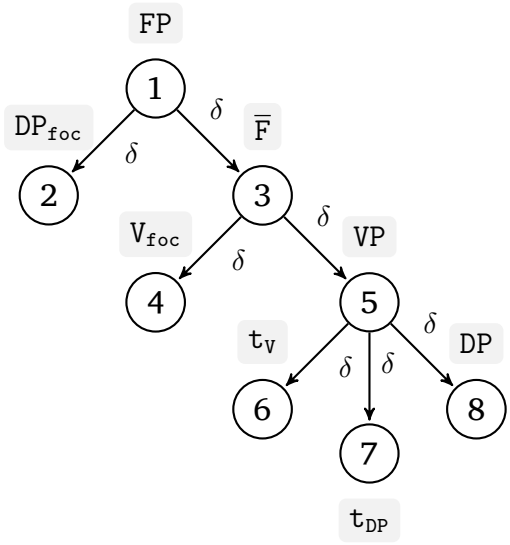


Figure 14.3: Model of the output tree in Figure 14.1. The successor relation among siblings is omitted for readability.

I also wish to explicate the basic Hungarian phrase structure tree formally by stating that trees follow a basic X-bar structure and that FocusP selects for VP. To accomplish this, it will be helpful to make use of additional concepts such as sisterhood, the root of a tree, and matching categories. I first define the 'sister' relationship in trees: two nodes are sisters if there is a node that immediately dominates both of them.

$$\texttt{sister}(x, y) \stackrel{\text{def}}{=} (\exists z)[\delta(z, x) \wedge \delta(z, y)] \quad (14.14)$$

Next, roots are nodes that do not have a parent.

$$\texttt{root}\,(x) \stackrel{\text{def}}{=} \neg(\exists y)[\delta(y,x)] \tag{14.15}$$

Finally, two nodes match in category if they have the same category feature.

$$\texttt{same\_cat}\,(x,y) \stackrel{\text{def}}{=} (\texttt{Focus}(x) \wedge \texttt{Focus}(y)) \vee (\texttt{V}(x) \wedge \texttt{V}(y)) \vee (\texttt{D}(x) \wedge \texttt{D}(y)) \tag{14.16}$$

I now define X-bar structure in Hungarian: every phrase node dominates a specifier and bar node, and the bar node dominates a head node. The category feature of all those nodes match. Because of the flat VP structure, the X-bar structure does not apply to verb phrases.

$$\begin{aligned}\texttt{Xbar} \stackrel{\text{def}}{=}\ & (\forall x)(\exists y,z,w)[(\neg\ \texttt{VP}\,(x) \wedge \texttt{phrase}(x)) \rightarrow \\ & (\delta(x,y) \wedge\ \texttt{same\_cat}\,(x,y) \wedge \texttt{specifier}(y) \\ & \wedge\, \delta(x,z) \wedge\ \texttt{same\_cat}\,(x,z) \wedge \texttt{bar}(z) \\ & \wedge\, \delta(z,w) \wedge\ \texttt{same\_cat}\,(x,w) \wedge \texttt{head}(w))]\end{aligned} \tag{14.17}$$

It is also the case that the head of Focus selects for VP.

$$\texttt{focus\_select} \stackrel{\text{def}}{=} (\forall x)[\ \texttt{FH}\,(x) \rightarrow \exists(y)[\ \texttt{sister}\,(x,y) \wedge\ \texttt{VP}\,(y)]] \tag{14.18}$$

Taken all this together, I define Hungarian phrase structure trees as follows: the tree is rooted by a FP node, it follows X-bar structure, and focus selects for VP.

$$\begin{aligned}\texttt{HuPST} \stackrel{\text{def}}{=}\ & (\exists x)[\ \texttt{root}\,(x) \wedge\ \texttt{FP}\,(x)] \\ & \wedge\ \texttt{Xbar} \wedge\ \texttt{focus\_select}\end{aligned} \tag{14.19}$$

With these representations in place, it remains to formalize the transformation that converts the input the tree visualized in Figure 14.2 to the output tree visualized in Figure 14.3.

14.3 Logical transduction

Here I describe the tree-to-tree mapping in (14.1) in terms of a MSO logical transduction. Because the structure of the tree does not change, only one copy set of the nodes is needed.

$$C = \{1\} \tag{14.20}$$

The domain and licensing formulas are set to `true`.

The dominance and successor relations stay the same.

$$\phi_{\delta}(x,y) \stackrel{\text{def}}{=} \quad \delta(x,y) \tag{14.21}$$

$$\phi_{\triangleleft}(x,y) \stackrel{\text{def}}{=} \quad \triangleleft(x,y) \tag{14.22}$$

The unary relations `phrase`, `specifier`, `bar`, `head`, and `Focus` also do not change in the transduction.

$$\phi_{\texttt{phrase}}(x) \stackrel{\text{def}}{=} \quad \texttt{phrase}(x) \tag{14.23}$$

$$\phi_{\texttt{specifier}}(x) \stackrel{\text{def}}{=} \quad \texttt{specifier}(x) \tag{14.24}$$

$$\phi_{\texttt{bar}}(x) \stackrel{\text{def}}{=} \quad \texttt{bar}(x) \tag{14.25}$$

$$\phi_{\texttt{head}}(x) \stackrel{\text{def}}{=} \quad \texttt{head}(x) \tag{14.26}$$

$$\phi_{\texttt{Focus}}(x) \stackrel{\text{def}}{=} \quad \texttt{Focus}(x) \tag{14.27}$$

On the other hand, the unary relations `V`, `D`, `focused`, and `trace` can change between an input and output structure, depending in part on whether there is a focused constituent within the VP. I define the $\texttt{focus?}\,(X)$ predicate to check for focused constituents within the VP. I will assume here that all focused constituents are DP.

$$\texttt{focus?}\,(X) \stackrel{\text{def}}{=} \quad (\exists x, y \in X)[\delta(x,y) \wedge \texttt{VP}(x) \wedge \texttt{focused}(y)] \tag{14.28}$$

A node in the output tree is V either if it was already a V in the input tree, or if it was a Focus head and there is a focused element in the sentence.

$$\phi_{\texttt{V}}(x) \stackrel{\text{def}}{=} \quad \texttt{V}(x) \vee ((\exists X)[\,\texttt{focus?}\,(X)] \wedge \,\texttt{FH}\,(x))$$

A node in the output tree is D if it was a D in the input tree or if it was a Focus specifier and there is a focused element in the sentence.

$$\phi_{\texttt{D}}(x) \stackrel{\text{def}}{=} \texttt{D}(x) \vee ((\exists X)[\,\texttt{focus?}\,(X)] \wedge \,\texttt{FS}\,(x)) \tag{14.29}$$

Nodes in the output tree are focused if they satisfy an analogous situation.

$$\phi_{\texttt{focused}}(x) \stackrel{\text{def}}{=} \texttt{focused}(x) \vee ((\exists X)[\,\texttt{focus?}\,(X)] \wedge \,\texttt{FS}\,(x)) \tag{14.30}$$

Finally, nodes in the output tree are traces, if the original lexical item moved out from there. This occurs when there was a focused element in the sentence, focused DPs and verbs get labeled with trace.

$$\phi_{\texttt{trace}}(x) \stackrel{\text{def}}{=} \texttt{focus?}\,(X) \wedge (\,\texttt{DP}_{\texttt{foc}}\,(x) \vee \,\texttt{VH}\,(x)) \tag{14.31}$$

Table 14.1 shows aspects of the computation for the unary relations `V`, `D`, `focused`, and `trace` for domain elements 2, 4, 6, and 7 in the output tree in Figure 14.3. Observe that in the input structure in Figure 14.2, $\texttt{focus?}\,(\{5,2\})$ evaluates to true.

14.4 Conclusion

In this chapter, I have described focus and verb movement in Hungarian using model-theoretic representations and a logical transductions, based on the analysis by Brody (1990). In order to adequately describe the transduction, monadic second order (MSO) logic was used. The transduction relied on some unorthodox assumptions, such as a fully built input tree.

This assumption was crucial for the current analysis. If this were not the case, the copy set would have to be larger than one to allow for the output structures to be larger than the input structure. However, any particular choice for the copy set effectively limits the number of times a lexical item can potentially move. If a lexical item can move an unboundedly many times then there could be no copy set large enough. This issue also arises in Chapters 20 and 21 in the context of the phonological cycle as well as the linearization of reduplicative morphemes under certain representational assumptions.

An alternative without such nonstandard assumptions would have been to use the Minimalist Grammars formalism (Stabler, 1997). In that case,

	x			
Formulas	2	4	6	7
$\texttt{D}(x)$	$\bot$	$\bot$	$\bot$	$\top$
$\texttt{V}(x)$	$\bot$	$\bot$	$\top$	$\bot$
$\texttt{Focus}(x)$	$\top$	$\top$	$\bot$	$\bot$
$\texttt{head}(x)$	$\bot$	$\top$	$\top$	$\bot$
$\texttt{phrase}(x)$	$\bot$	$\bot$	$\bot$	$\top$
$\texttt{specifer}(x)$	$\top$	$\bot$	$\bot$	$\bot$
$\texttt{focused}(x)$	$\bot$	$\bot$	$\bot$	$\top$
$\texttt{DP}_{\texttt{foc}}(x)$	$\bot$	$\bot$	$\bot$	$\top$
$\texttt{VH}(x)$	$\bot$	$\bot$	$\top$	$\bot$
$\texttt{FH}(x)$	$\bot$	$\top$	$\bot$	$\bot$
$\texttt{FS}(x)$	$\top$	$\bot$	$\bot$	$\bot$
$\phi_{\texttt{D}}(x)$	$\top$	$\bot$	$\bot$	$\top$
$\phi_{\texttt{V}}(x)$	$\bot$	$\top$	$\top$	$\bot$
$\phi_{\texttt{focused}}(x)$	$\top$	$\bot$	$\bot$	$\top$
$\phi_{\texttt{trace}}(x)$	$\bot$	$\bot$	$\top$	$\top$

Table 14.1: Computing $\phi_{\texttt{D}}(x)$, $\phi_{\texttt{V}}(x)$, $\phi_{\texttt{focused}}(x)$, and$\phi_{\texttt{trace}}(x)$ given the input structure in Figure 14.2.

only syntactic derivational features would be given with each lexical item, and every merge and move would be motivated by these features. All syntactic processes in the Minimalist Grammars (MGs) framework can be described in terms of MSO logical constraints on the derivation itself (Graf, 2013). Still, an additional mapping is needed from a derivation tree to a derived tree to yield the surface sentence string. It could be interesting future work to examine the complexity of a logical tree-to-tree transductions between a derivation tree and a derived tree in the MGs framework.

Part III

Theoretical Contributions

Chapter 15

Syllable Structure and the Sonority Sequencing Principle

KRISTINA STROTHER-GARCIA

15.1 Introduction

This chapter[1] examines well-formedness constraints as they pertain to syllable structure in light of Chapter 5, which argued that constraints like *NT and *N..L can actually be expressed with logical languages weaker than first-order logic. Those logical languages included propositional logic, and a restricted form of propositional logic referred to as conjunction of negative literals (CNL). This chapter argues that this restricted propositional logic is sufficient to describe constraints which determine the language-specific well-formedness of syllabic structures constituting words across languages. This is important because it means deciding whether such a structure is well-formed amounts to a series of local decisions and therefore recourse to global optimization is unnecessary.

Recall from Chapter 5, that a propositional logic can be defined in terms of the **factors** of relational structures. Each factor picks out a connected structure. For example, in the context of syllable structure, consider the two factors C_{ons} and C_{cod} shown in Figure 15.1. The factors in Figure 15.1 present the necessary and sufficient components of structures that contain complex onsets (left) and complex codas (right). In the figure, the

[1] This chapter is essentially an updated version of Strother-Garcia (2019, Chapter 5).

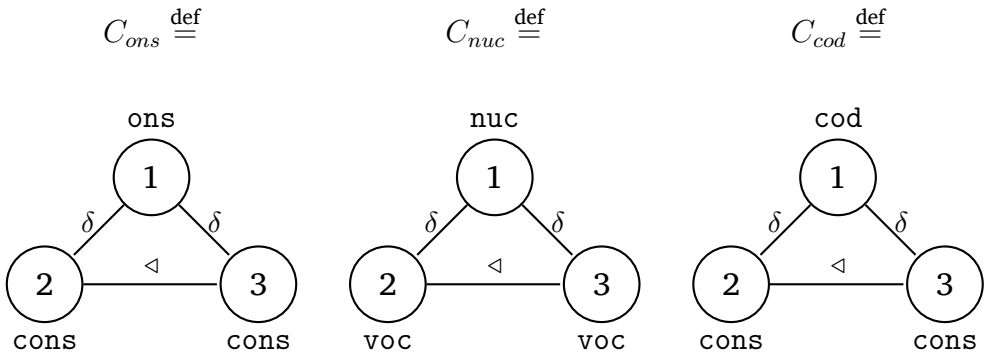


Figure 15.1: Factors for complex onsets (left), complex nuclei (center), and complex codas (right).

symbol $\triangleleft$ denotes the successor relation, and δ indicates the dominance relation. (Syllabic relational structures are formalized in the next section.) Factors like the ones in Figure 15.1 constitute the atoms (the 'literals') of a propositional logic.

The interpretation of a factor S in this logical language is all the relational structures *containing* S. Consequently, if one wishes to characterize a language *without* complex syllable margins, the logical sentence below suffices.

$$\texttt{*ComplexMargin} \stackrel{\text{def}}{=} \neg C_{ons} \wedge \neg C_{cod} \tag{15.1}$$

Recall from Chapter 5, that negation is interpreted as the complement. Thus $\neg C_{ons}$ is interpreted at the set of all structures which *do not contain* a complex onset. Similarly, $\neg C_{cod}$ is interpreted at the set of all structures which do not contain a complex coda. Their conjunction thus picks out all structures which do not contain a complex syllable margin.

A sentence of propositional logic is a conjunction of negative literals provided that it is a conjunction of terms $\neg F_1 \wedge \ldots \wedge \neg F_n$ where each F_i is a factor. The `*ComplexMargin` is a conjunction of negative literals.

This chapter examines language-universal and language-specific aspects of syllable-structure, and observes that they are expressible in CNL logic over syllabic representations. This kind of analysis has been applied to tonal mapping (Jardine, 2017) and to other aspects of phonotactics (Rogers and Lambert, 2019b).

In the remainder of this chapter, I first present a model-theoretic representations of the syllable and discuss universal principles of syllable structure discussed and analyzed by Clements and Keyser (1983) and (Davis, 1988). I express these principles with logic. Next I review the four major language types based on CV typology as treated in OT by Prince and Smolensky (2004). I present factors within a CNL logic sufficient to characterize this four-way typology. Moving beyond the simplest CV-type structures, I also describe the formation of complex syllable margins, relying on a logical formulation of the Sonority Sequencing Principle (SSP). Some implications of this result are discussed in the conclusion to this chapter.

15.2 Universal Principles of Syllable Structure

Clements and Keyser (1983) argue for three levels of syllable structure: a syllabic tier, a CV tier and a segmental tier, which are organized hierarchically. While Clements and Keyser recognize the syllabic constituent nucleus, they argue that the the constituent categories onset and coda are unnecessary in part because they can be derived from other information from their model. In contrast, (Davis, 1988) argues for the primacy of syllabic constituents onset, nucleus and coda. Such categories likewise played a prominent role for constraint-based theories of phonology such as Optimality Theory (Prince and Smolensky, 2004). Goldsmith (2011) provides an overview of theories and representations of the syllable.

This chapter considers a representational theory of the syllable recognizing the following the three levels: a syllabic tier, a tier for syllabic roles (onset, nucleus, coda), and a segmental tier. For example, the English word *plenty* [plɛnti] would have the following syllabic representation.

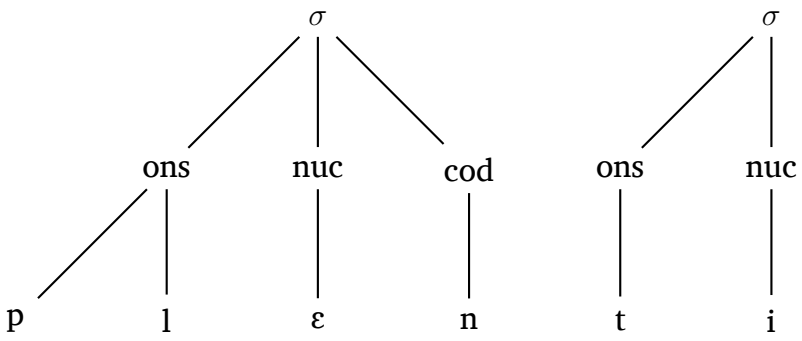


Figure 15.2: A syllabic representation of the English word *plenty*.

One goal is define the universal constraints on syllabic representations using relational structures and sentences in logical language. A relational model for syllabic representations is much like an autosegmental model (see Chapter 12). Equation (15.2) provides a model signature, where $\triangleleft$ is the successor relation, δ is a binary dominance relation, $\mathcal{F}$ is a set of unary relations for standard segmental features, S is a of unary relations for the syllabic roles $\texttt{ons}, \texttt{nuc}, \texttt{cod}$,, B is the set of two unary relations indicating the word boundary symbols $\rtimes, \ltimes$, and finally σ is the unary relation for elements at the topmost level of the syllabic structure.

$$\mathfrak{R} = \{\triangleleft, \delta\} \cup \mathcal{F} \cup S \cup B \cup \{\sigma\} \tag{15.2}$$

As an example, Figure 15.3 presents the $\mathfrak{R}$-structure for the English word *plenty*. (This figure does not show the word boundary symbols, but they are present at each level of structure.)

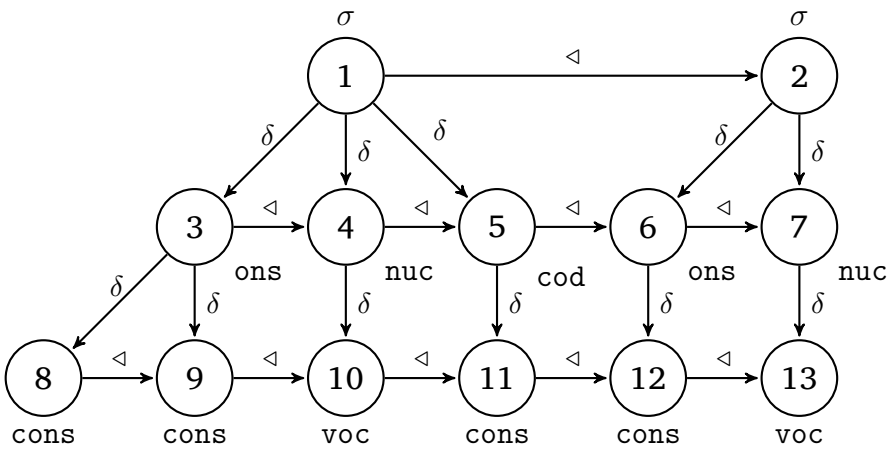


Figure 15.3: The $\mathfrak{R}$-structure for the English word *plenty*, with only the consonantal and vocalic features shown for the segments. Word boundaries are not shown, but are present at each level of structure.

Like autosegmental representations (Chapter 12) it is not the case that any arbitrary assignment of domain elements to the relations in $\mathfrak{R}$ constitutes a well-formed syllable structure. It is important that there are three distinct tiers, whose elements are related by the $\triangleleft$ relation. It is also important that the unary relations not in B only pertain to exactly one of these levels. Here, the relation σ pertains to the highest level, the relations in S pertain to the middle tier, and the relations in $\mathcal{F}$ pertain to the lowest level, the segmental tier. The dominance relation occurs only between adjacent levels. Finally, every syllable must dominate some nucleus, and every position on the syllable role tier must dominate some segment and be dominated by some syllable. Logical sentences establishing these conditions on well-formedness are provided in the appendix.

There are additional universals of syllable structure. For example, the strings that compose the middle tier with the syllabic roles form a strictly 2-local language. In particular, the following 2-factors are forbidden: $\rtimes c, \rtimes\ltimes, cc, cn, oc, oo, o\ltimes$, where c indicates a coda position, n a nucleus position, and o an onset position. The constraint on cn sequences follows from the principle of syllabification that onset formation is prioritized over coda formation. If a lower-sonorous position x precedes a nucleus position, then x must be an onset.

Additional syllable structure limits the number of roles elements on the

syllabic tier can dominate. I will let the reader work out the details for themselves, but syllables with two onsets, two codas, two nuclei, with a nucleus followed by an onset, and so on, are forbidden.

It follows that syllables have one of the four basic structures shown in Figure 15.4. There are two observations worth mentioning. First, the

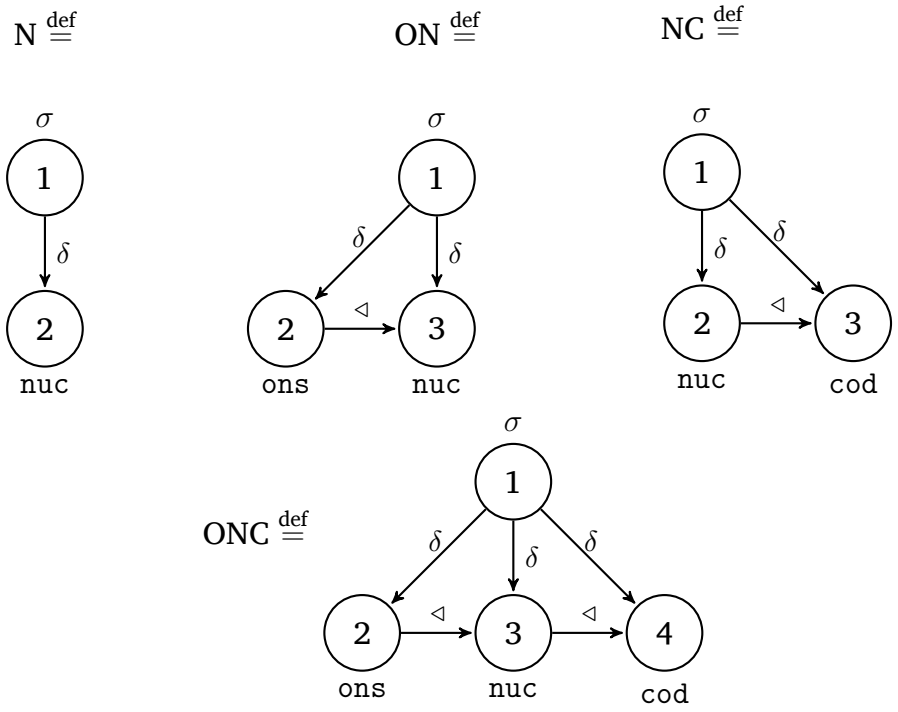


Figure 15.4: The four permissible syllable structures.

structure N is contained within the structures ON, NC, and ONC. This follows from the requirement that every word contain at least one syllable and each syllable contains exactly one nucleus (see appendix). Second, it is not the case that any of these syllable structures can be "concatenated" with each other to form well-formed words. For example, while NC is a valid structure for a word, concatenating NC with itself to yield a structure like NC $\cdot$ NC would not be a valid word because it violates the ban on cn sequences in the syllable role tier.

15.3 Basic CV Typology

Basic CV typology refers to the a four-way typology of syllable types occurring in languages which can be derived from two principles: (1) that onsets be required and (2) that codas be forbidden (Jakobson, 1962; McCarthy, 1979; Clements and Keyser, 1983; Blevins, 1995). This typology assumes that complex syllable margins and complex nuclei are forbidden, and thus only syllables with a single C or V in each syllabic position are considered. In the context of the propositional logical language based on factors of $\mathfrak{R}$-structures, this means sentences conjoined with the sentence $\neg C_{ons} \wedge \neg C_{nuc} \wedge \neg C_{cod}$ (Figure 15.1). Syllables with complex syllable margins are discussed in §15.4. Syllables with complex syllable nuclei are left for future research.

15.3.1 The Typology

If a given language requires onsets and forbids codas, the only acceptable syllable type will be CV.[2] If codas are forbidden and onsets are not required, syllables may take the form V or CV. Requiring onsets and allowing codas yields CV and CVC syllables. Finally, when onsets are not required and codas are not forbidden, V, CV, VC, and CVC syllables will all be grammatical. It is possible to place these syllable types in one-to-one correspondence with the structures shown in Figure 15.4 since complex onsets, nuclei, and codas are not under consideration. It follows that there 2^4 logically possible subsets of these four syllable types. Nonetheless, languages with simple syllable margins have only been identified with only four of these logically possible subsets, as summarized in Table 15.1.

[2]Technically, C denotes a domain element x satisfying $\texttt{cons}(x)$. Here we make no distinction between $\texttt{cons}(x)$ and $\neg\texttt{voc}(x)$ and so glides are grouped with consonants. Atypical syllable structures (e.g., complex nuclei, syllabic consonants) are beyond the scope of this chapter, and would be interesting to study in future research.

	Onsets Required	**Onsets Not Required**
Codas Forbidden	CV	V, CV
Codas Not Forbidden	CV, CVC	V, CV, VC, CVC

Table 15.1: Basic typology of syllable structures.

With the basic CV typology made clear, I now demonstrate how it can be accounted for using a conjunction of negative literals, where the literals are the factors of $\mathfrak{R}$-structures of the kind discussed in §15.2.

15.3.2 Analysis of the Typology

It is straightforward to identify $\mathfrak{R}$-structures for syllables with onsets or codas. These are just factors of size 1, banning any element satisfying `ons` and `cod`, respectively. Following the notation introduced in §15.2, I will refer to these factors as o and c respectively. As positive factors, o and c pick out all and only those $\mathfrak{R}$-structures with at least one onset or coda, respectively. The negative literals $\neg o$ and $\neg c$ pick out all and only those $\mathfrak{R}$-structures with no onsets or codas, respectively. This means that it is straightforward to implement a ban on codas with the negative literal $\neg c$.

Interestingly, the positive literal o also fails to capture the essence of the generalization "All syllables have onsets." To see why, consider the expression in first order logic which corresponds to the positive literal `Onset` shown in Equation 15.3.

$$\exists x[\texttt{ons}(x)] \tag{15.3}$$

Since words can have more than one syllable, this sentence does not capture the generalization that "All syllables have onsets." It just requires that there exist a single syllable with an onset. Any onset effectively licenses the whole word (with respect to this factor), even if other syllables do not have onsets.

In first order logic, one way to express the generalization that "All syllables have onsets" is shown in Equation 15.4.

$$\forall x[\texttt{syl}(x) \rightarrow (\exists y[\texttt{ons}(y) \wedge \delta(x,y)]] \tag{15.4}$$

I am interested in whether or not this generalization can be described in terms of a conjunction of negative literals, where the literals are factors of $\mathfrak{R}$-structures. In what follows I show that it can be.

Onsets precede nuclei in syllables. Equivalently, this means that nuclei in syllables cannot be preceded by the word boundary, by another nucleus, or by a coda. In other words, banning the 2-factors $\rtimes n, nn, cn$ on the syllable role tier has the effect that onsets must always precede nuclei. Since each syllable has exactly one nucleus this means that every syllable will have an onset.

§15.2 posited that forbidding the 2-factor cn was a universal. Therefore the language-specific generalization "all syllables have onsets" comes down to the conjunction of negative literals $\neg \rtimes n \wedge \neg nn$.

It follows that the four way typology is described with the following conjunctions of negative literals shown below. Letting $\texttt{NoCoda} = \neg c$ and $\texttt{Onset} = \neg \rtimes n \wedge \neg nn$, Table 15.2 shows the conjunctions of negative literals which give raise to the basic CV typology.

sentence	allowable syllables
$\texttt{NoCoda} \wedge \texttt{Onset}$	CV
$\texttt{NoCoda}$	V, CV
$\texttt{Onset}$	CV, CVC
$\texttt{true}$	V, CV, VC, CVC

Table 15.2: Basic typology of syllable structures.

Of course the complex nature of the predicate $\texttt{Onset}$ raises the question whether its component parts $\neg \rtimes n$ and $\neg nn$ ever assert themselves independently. If they did, then we would expect to find languages which forbid onsets word-initially but admit them word-internally (satisfying $\neg \rtimes n$ but not $\neg nn$). Similarly, we would expect to find languages which forbid onsets word-internally but admit them word-initially (satisfying $\neg nn$ but not $\neg \rtimes n$).

This turns out to be exactly the case. In the introduction to her article, Flack (2009) writes "Word-initial syllables in Axininca Campa may either have onsets or be onsetless, but onsets are required in all non-initial syllables

(Payne 1981, McCarthy & Prince 1993)" and that "Initial syllables in Madi must have onsets, while medial syllables may have onsets or be onsetless (Tucker 1967)."

Flack (2009) also points out codas show analogous cross-linguistic effects. She writes "Similarly in Kamaiurá, codas are permitted only in word-final syllables (Everett & Seki 1985, McCarthy & Prince 1986)" and that "in Chamicuro, codas are banned in word-final syllables but may occur non-finally (Parker 2001: 365–366)." In the context of the analysis being presented here, this means that Kamaiurá and Chamicuro, codas are allowed somewhere and so the factor c is not forbidden. Forbidding the word-final codas is expressed by the 2-factor $c\ltimes$ and forbidding word-internal codas is effectively expressed by the 2-factor co. (Recall that the 2-factor cn is universally banned.) It follows that in Kamaiurá and Chamicuro, one of these 2-factors, but not the other is banned.

One might reasonably ask whether one expects the logically possible constraint $\neg o$ ("No Onset") to also appear independently. This is an excellent question, but it is not unique to the analysis presented here. It is an important issue for OT as well, as I explain in §15.3.3 below.

To conclude this section, the basic CV typology follows simply from the the conjunction of negative factors. While there is no single factor requiring onsets, the two factors which together require onsets for every syllable appear to operate independently across languages.

15.3.3 Comparison to OT

It is instructive to compare the analysis above to the one offered by Optimality Theory. In OT, (Prince and Smolensky, 2004) arrive at the basic CV typology language types by examining the interactions of two markedness constraints, ONSET and NOCODA, and two faithfulness constraints, PARSE and FILL. The markedness constraints straightforwardly implement the two principles driving the four-way typology described above. The faithfulness constraints enforce a one-to-one relation between underlying segments and syllabic positions. All four constraints are given below.

1. OT constraints from Prince and Smolensky (2004, pp. 93-95).

   (a) **ONSET**: A syllable must have an onset.
   (b) **NOCODA**: A syllable must **not** have a coda.

(c) **PARSE**: Underlying segments must be parsed into syllable structure.
(d) **FILL**: Syllable positions must be filled with underlying segments.

In accordance with the OT tenant that all languages share a universal set of violable constraints (CON), Prince and Smolensky (2004) generate the four basic syllable typologies by reordering (1a) through (1d). Even though there are 24 logically possible constraint rankings, only *four* unique typologies emerge, the ones shown in Table 15.1.

There are two points worth mentioning. The first is that this result obtains only if the logically possible constraints NOONSET and CODA are *excluded* from CON. If these constraints are included in CON then the factorial typology expands to include languages which forbid onsets and require codas.

The second point is that ranking ONSET and NOCODA relative to the faithfulness constraints determines whether they are active or not in the language. If they are ranked above the faithfulness constraints, they are active and onsets will be required and codas forbidden. If they are ranked below some faithfulness constraint then they can be considered inactive, and onsets will not be required and codas will not be forbidden. In terms of sentences of propositional logic, whether a constraint is active or not means whether or not it is included as a term in the conjunction of negative literals.

The third point is that EVAL computes a global evaluation over candidates to determine the optimal form. However, the logical analysis makes clear not such global computation is necessary for determining well-formedness. The presence or absence of each factor is a local computation, and those local computations are fed directly into a propositional sentence for evaluation.

In sum, the logical analysis of the basic CV typology distills the the issues to their core. That onsets are required and codas forbidden is captured exactly by the terms $\texttt{Onset}$ and $\neg\,\texttt{Coda}$, which are merely conjunctions of negative literals. The fact that the typology is what it is derives from the availability of these constraints in a theory of language and the non-availability of logically possible constraints like $\texttt{Coda}$ and $\neg\,\texttt{Onset}$. Accounting for this asymmetry is as much an issue for OT as it is for the present analysis. Whatever explanation is forthcoming will likely not necessarily follow the formal structure of these theory, but from factors external

to it. Finally, the four-way typological distinctions have been shown to be derived without global optimization; inviolable constraints work at least as well as violable ones in this regard. I conclude that in this simple corner of phonology, when it comes to accounting for the basic CV typology, the above analysis fares no worse than OT.

There are many other arguments for OT which are not addressed here. For example, the OT analysis not only determines which surface sequences are well-formed, it provides a mapping from an underlying sequence of Cs and Vs to well-formed, parsed syllable structure. In contrast, the analysis presented here only focuses on well-formedness. However, I have argued elsewhere that logical transductions of the kinds presented in Chapter 3 provide similar insights into the process of syllabification itself (Strother-Garcia, 2018a, 2019). The primary conclusion of that work is that, even for the complex cases of syllabification found in for Imdlawn Tashlhiyt Berber and Moroccan Arabic, the necessary computations are local, and global optimization is unnecessary.

15.4 Complex Onsets and Codas

I now expand the analysis in the previous section to account for complex syllable margins. My approach is to formalize the Sonority Sequencing Principle (SSP) as a conjunction of forbidden factors.

Sievers (1881) wrote one of the first versions of the SSP (also called the Sonority Sequencing Generalization or SSG), observing that sonority generally rises between a given segment and the sonority peak of its syllable. This principle has been reiterated in one form or another by many influential linguists (e.g. Saussure, 1916; Hooper, 1976; Selkirk, 1984; Clements, 1990). While there is a great deal of disagreement regarding the psychological and physiological realizations of sonority, some principles are generally agreed upon. Namely, vowels are more sonorous than sonorant consonants (glides, liquids, and nasals), which are in turn more sonorous than obstruents (fricatives, affricates, and stops). Finer distinctions between members of these three natural classes are more contentious, but the exact details of the sonority hierarchy do not matter for the purposes of this chapter. It is also worth noting that many languages allow SSP violations, so this is not a universal principle in its most strict interpretation.

In what follows, I define binary sonority relations that encode the

relative sonority of two segments. I then introduce two factors, which when included as negative literals, enforce a certain version of the SSP.

15.4.1 Sonority Relations

I assume there is some known sonority hierarchy (either for a given language or more universally) such that, for every pair of phonemes, it is known whether or not one is less sonorous than the other. If position x is less sonorous than position y, I write $<_s(x, y)$ or, equivalently, $x <_s y$. It is a useful exercise for the reader to derive this sonority relation from the relations in the $\mathfrak{R}$-signature. In accordance with the traditional notion of lesser sonority, this binary relation $<_s$ is *irreflexive*, *asymmetric*, and *transitive*.

Given these properties of $<_s$, it is simple to define a relation $=_s$ to represent equal sonority.

$$=_s(x, y) \stackrel{\text{def}}{=} \neg <_s(x, y) \wedge \neg <_s(y, x) \tag{15.5}$$

Essentially, two positions x and y are equally sonorous iff x is not less sonorous than y and y is not less sonorous than x.

Similarly, I define the relation $\leq_s$ to represent equal or lesser sonority.

$$\leq_s(x, y) \stackrel{\text{def}}{=} <_s(x, y) \vee =_s(y, x) \tag{15.6}$$

This equation says the position x is equally or less sonorous than position y iff x is less sonorous than y or x and y are equally sonorous.

I also define the relation $>_s$ to represent greater sonority.

$$>_s(x, y) \stackrel{\text{def}}{=} \neg \leq_s(x, y) \tag{15.7}$$

Position x is more sonorous than position y iff x is not equally or less sonorous than y.

In this analysis, the relations $<_s, =_s$, and $>_s$ are atomic predicates to be included in the signature of $\mathfrak{R}$-structures. I discuss this and other issues around whether these relations are atomic or derived after presenting the bulk of the analysis.

Figure 15.5 presents an example of a $\mathfrak{R}$-structure for the word *blink* [blɪnk]. In this figure, the sonority relations are shown only for on the segmental tier for positions $x, x + 1$. For the sonority sequencing principle, these are the only sonority relations that matter. Position 4 is less sonorous than position 5, and position 7 is more sonorous than position 8.

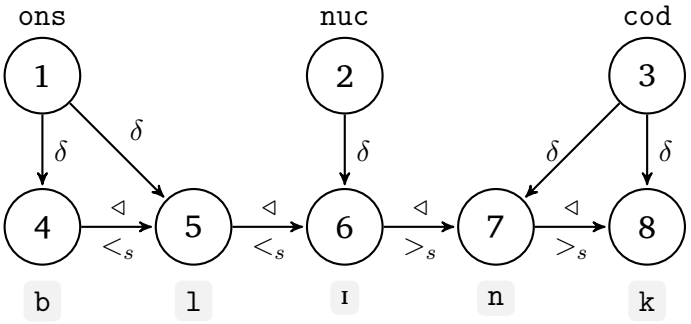


Figure 15.5: A $\mathfrak{R}$-structure for the word [blɪnk] with sonority relations illustrated from left to right.

15.4.2 Constraints on Sonority Sequencing

The SSP can be understood in different ways with respect to successive positions of equal sonority. Some languages allow equally sonorous consonants in a complex onset/coda, meaning that a strict rise in sonority from the syllable boundaries to the nucleus is not required. Here, I analyze the the SSP in this way.

To enforce this version of the SSP, it suffices to ban two factors: first, an onset segment must not be more sonorous than its successor; second, a coda segment must not be more sonorous than its predecessor. These factors are shown in Figure 15.6. It follows that the non-strict version of

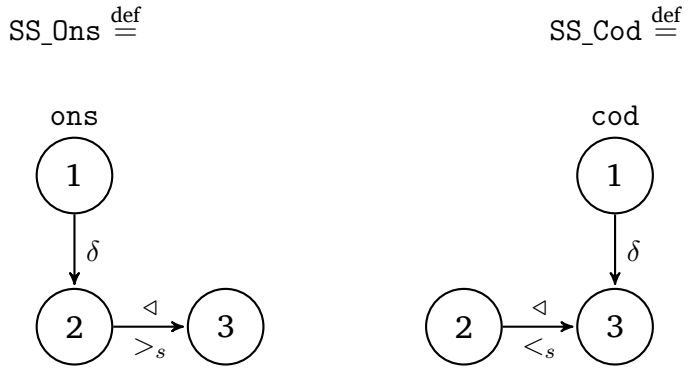


Figure 15.6: Sonority Sequencing factors for Onsets (left) and Codas (right).

the SSP follows from the conjunction shown in Equation 15.8.

$$\texttt{SSP} \stackrel{\text{def}}{=} \neg\texttt{SS\_Ons} \wedge \neg\texttt{SS\_Cod} \tag{15.8}$$

These constraints are mirror images of each other.

To illustrate how these factors enforce the SSP, consider the forbidden factor SS_Ons. It applies in two situations. For one, it forbids an onset from being more sonorous than a nucleus immediately following it. Figure 15.7 illustrates such a structure, an onset `w` with a nucleic `s`. The banned structure SS_Ons is bolded.

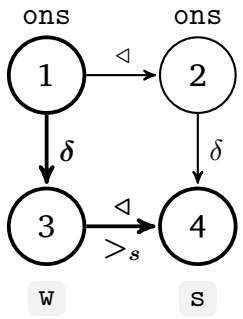


Figure 15.7: An onset followed by a nucleus of lesser sonority.

Importantly, SS_Ons also bans a left-to-right fall in sonority among segments within a complex onset. Consider the nonce word *blick* [blık], which native English-speakers generally agree is a viable (though unattested) word in English (Chomsky and Halle, 1965). Reversing the order of the onset segments yields *lbick* [lbık], which is clearly not acceptable as an English word. One explanation for this is that *blick* satisfies the SSP while *lbick* does not. Figure 15.8 illustrates the $\mathfrak{R}$-structure of [lbık]. The bolded structure precisely matches the forbidden factor SS_Ons.

In contrast, Figures 15.9 illustrates the $\mathfrak{R}$-structure for the word *blick* [blık]. This structure satisfies SS_Ons because it does not contain the banned factor. Crucially, the first segment `b` in [blık] is less sonorous than its successor `l`.

Forbidding the factor SS_Cod has the same kind of effects as forbidding SS_Ons. I leave it to the reader to confirm this by considering $\mathfrak{R}$-structures which satisfy the expression ¬SS_Cod, and which do not satisfy it.

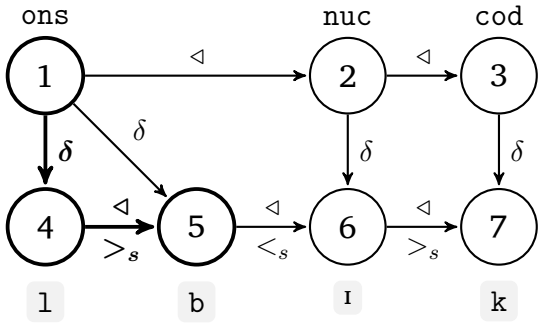


Figure 15.8: The $\mathfrak{R}$-structure of [lbɪk] with the forbidden factor SS_Ons bolded. Word boundaries and the syllable domain element omitted.

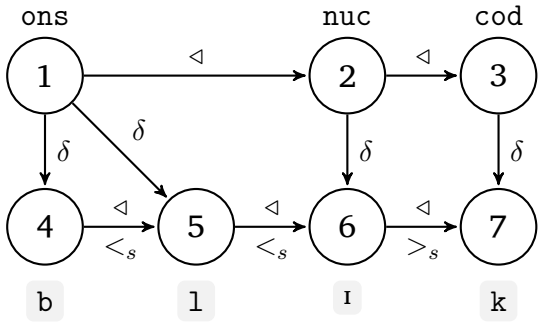


Figure 15.9: The $\mathfrak{R}$-structure of [lbɪk] with the forbidden factor SS_Ons bolded. Word boundaries and the syllable domain element omitted.

This section analyzed a version of the SSP that forbids falling sonority from the syllable boundaries to the nucleus. This generalization is expressed in Equation 15.8, which effectively forbids two factors: one where onset elements are succeeded by a less sonorous position, and one where coda elements are preceded by a less sonorous position.

A key assumption in obtaining this result was that the sonority relations are part of the signature. It makes sense then to ask whether this position is reasonable. As discussed in Chapter 2, this amounts to a decision as to whether sonority relations are included in the primitive psychological units that make up mental representations of words. If they are, then it is appropriate to include in the $\mathfrak{R}$-signature. In this regard, it is important to note that while the sonority relation was defined for any two positions

x and y, the only sonority relations that matter for this analysis are those when x and y are adjacent. Given that sonority relations from adjacent positions essentially encodes a difference (e.g. a rate of change) in the sonority between successive positions, I do not think it is too far-fetched to hypothesize that they possess a degree of psychological reality.

15.5 Conclusion

Drawing on propositional logic and model-theoretic representations of syllable structure, I developed a new way of deriving Jakobson's (1962) typology. I showed that the well-formedness of various language specific constraints on both simple and complex syllable margins can be expressed simply by forbidding finitely many syllabic structures. The language-specific principles that determine the exact size and structure of complex syllabic margins in each language (beyond the scope of the SSP) remain to be described. Complex nuclei, ambisyllabic, and extra-syllabic segments are also important areas for future research. While there are many syllabic structures not covered here, I have provided the groundwork for a computational model of syllable structure.

This chapter has not treated the process of syllabification itself, and I refer readers to (Strother-Garcia, 2018a) and (Strother-Garcia, 2019, Chapter 7) for explicit accounts of the syllabification processes in Imdlawn Tashlhiyt Berber and Moroccan Arabic, respectively. In those analyses, logical transductions of the kind presented in Chapter 3 were provided. Furthermore, those transductions were shown to be Quantifier-Free, which means that the process of syllabification process itself is local in a particular way (see Chapter 22).

Altogether these results indicate that computing the well-formedness of syllable structures and the syllabification process itself do not require globally optimizing over ranked inviolable constraints, and that, at least in the context of the syllable, computations are local.

Appendix

This appendix provides the logical formulas and sentences that explain which $\mathfrak{R}$-structures form multilinear syllabic structures.

To create three distinct tiers, it is important that the relevant properties are only present on the appropriate tier.

$$\texttt{sep}_{\texttt{f,s}} \stackrel{\text{def}}{=} (\forall x)[(\texttt{f}(x) \rightarrow \neg\texttt{t}(x)) \wedge (\texttt{s}(x) \rightarrow \neg\texttt{s}(x))] \tag{15.9}$$

$$\texttt{sep} \stackrel{\text{def}}{=} \bigwedge_{\texttt{f}\in F,\texttt{s}\in S} \texttt{sep}_{\texttt{f,t}} \bigwedge_{\texttt{f}\in F} \texttt{sep}_{\texttt{f},\sigma} \bigwedge_{\texttt{f}\in S} \texttt{sep}_{\texttt{f},\sigma} \tag{15.10}$$

Likewise it is important that elements of each tier refer only to certain properties. Formula 15.11 groups together all the segmental elements.

$$\texttt{segment}\,(x) \stackrel{\text{def}}{=} \bigvee_{\texttt{f}\in F} \texttt{f}(x) \tag{15.11}$$

Similarly, Formula 15.12 groups together all elements carrying syllabic roles.

$$\texttt{role}\,(x) \stackrel{\text{def}}{=} \bigvee_{\texttt{s}\in S} \texttt{s}(x) \tag{15.12}$$

The predicate $\texttt{same\_tier}\,(x, y)$ is true if and only if x and y are both segments, syllabic roles, or σ.

$$\begin{aligned}\texttt{same\_tier}\,(x,y) \stackrel{\text{def}}{=} \quad & (\,\texttt{role}\,(x) \wedge \texttt{role}\,(y)) \\ & \vee (\,\texttt{segment}\,(x) \wedge \texttt{segment}\,(y)) \\ & \vee (\sigma(x) \wedge \sigma(y))\end{aligned} \tag{15.13}$$

The sentence below constrains syllabic representations to structures in which tier only contain like units.

$$\texttt{good\_tiers} \stackrel{\text{def}}{=} (\forall x, y)[\,\texttt{same\_tier}\,(x,y) \leftrightarrow (x \leq y \vee y \leq x)] \tag{15.14}$$

It is also important that the dominance relations between tiers are the expected ones. In particular, the only dominance relations allowed are ones between σ elements and role elements as well as between role elements and segmental elements. The predicate $\texttt{WFD}\,(x, y)$ determines whether (x, y) a dominance relation is well-formed.

$$\texttt{WFD}\,(x,y) \stackrel{\text{def}}{=} x\delta y \rightarrow ((\sigma(x) \wedge \texttt{role}\,(y)) \vee (\,\texttt{role}\,(x) \wedge \texttt{segment}\,(y))) \tag{15.15}$$

Formula 15.16 then requires that all associations are of this type.

$$\texttt{good\_dom} \stackrel{\text{def}}{=} (\forall x, y)[x\delta y \rightarrow \texttt{WFD}\,(x, y)] \tag{15.16}$$

Formula 15.17 requires every domain element satisfying σ to dominate some nucleus and every domain elements satisfying a syllable role to dominate some segment.

$$\begin{aligned}\texttt{syl\_dom} \stackrel{\text{def}}{=} \; & (\forall x[(\sigma(x) \implies \exists y[\texttt{nuc}(y) \wedge x\delta y]) \\ & \wedge \texttt{role}\,(x) \implies \exists y[\texttt{segment}(y) \wedge x\delta y])]\end{aligned} \tag{15.17}$$

Putting this altogether, $\mathfrak{R}$-structures which can be said to be legitimate syllabic representations must satisfy Equation 15.18.

$$\texttt{SylR} \stackrel{\text{def}}{=} \texttt{sep} \wedge \texttt{good\_tiers} \wedge \texttt{good\_dom} \wedge \texttt{syl\_dom} \tag{15.18}$$

We observe that these three predicates have the same function for autosegmental representations (Chapter 12). They establish a multilinear structure whose elements on distinct tiers are related to each other via some kind of association relation (here dominance). Other universal aspects of syllable structure are discussed in §15.2.

Chapter 16

Primitive Constraints for Nonlexical Stress Patterns

Dakotah Lambert

This chapter explores the kinds of patterns that characterize word-stress. These patterns are built from constraints at the syllable level, restrictions on what kinds of stressed and unstressed syllables can cooccur and in which configurations. Unlike the constraints of Optimality Theory where analysis begins with a universal set of violable constraints (Prince and Smolensky, 2004), the constraints discussed here are language-specific unviolated descriptions of the surface structure. Thus, universality is neither assumed by, nor encoded in, the grammar. Nonetheless, this formalism allows for easy verification of putative universals.

This chapter makes the following contributions. First, it discusses some principles of formal description in terms of propositional logic. For example, in §16.2, the principle that accurate descriptions need to be complete often necessitates explicit reference to universal constraints that may otherwise go unmentioned. Another important point raised in that section is that there can be more than one accurate formal description of a given pattern, even when restricted to a particular kind of propositional logic. This naturally leads to the question of which such description is to be preferred, which is closely related to the problem of abductive reasoning and inference (Rawski, 2021). In §16.3 automatic methods are discussed as a way to ameliorate some of these issues. In terms of linguistic typology, the primary finding is that non-lexical stress patterns can be expressed with

a propositional logic using successor-based and precedence-based factors.

§16.1 begins with a discussion of what word-stress is and presents a basic feature system to encode it. Then §16.2 describes a principled way to derive a precise description of the surface forms in question. Many patterns admit several extensionally equivalent grammars, but because all stress patterns are regular, they have a normal form as a forward-deterministic, trim, minimal finite-state acceptor (Heinz, 2009). An automatic method of extracting some of the simplest kinds of constraints from these acceptors (Rogers and Lambert, 2019a) appears in §16.3.

16.1 The Phonotactics of Stress

Word-stress refers to phonological marking of one or more prominent syllables within the scope of a single word (van der Hulst, 2014). The most prominent among these are said to bear primary stress, while the least are called unstressed. If there is a perceptually clear third level between these two, it is a secondary stress. The particular markers of stress may be any of loudness, pitch, or duration, but we abstract away from these phonetics properties.

An important aspect of stress patterning is syllable weight. This is a language-specific categorization of syllables into different classes, generally based on characteristics such as vowel quality or quantity, or the presence or absence of a coda. The languages studied here consist of those in the StressTyp2 database[1] compiled by Goedemans *et al.* (2015) that have associated finite-state acceptors and prose descriptions; of these languages, none distinguish more than five weights. In this chapter, no example will use more than two: light and heavy.

These features combine to form a (finite) inventory of syllable types. Syllables are either light or heavy, and they either have zero stress, secondary stress, or primary stress. This can be formalized with the following model-theoretic structure, which also include word boundaries.

$$\mathfrak{R} = \{\texttt{light}, \texttt{heavy}, \texttt{primary}, \texttt{secondary}, \texttt{none}, \rtimes, \ltimes, \lhd, <\} \tag{16.1}$$

I include both the successor and precedence relations in order to consider factors using either order relation.

[1] Available at `http://st2.ullet.net/`.

Table 16.1 presents notation that will be used to indicate factors of $\mathfrak{R}$-structures of size 1. For example, the symbol L denotes a single domain element satisfying the atomic formulas `light` and `none`, symbol $^{\shortmid}$H denotes a single domain element satisfying the atomic formulas `primary` and `heavy`, symbol $^{\times}$L denotes a single domain element satisfying the atomic formula `light`, symbol $_{\shortmid}\sigma$ denotes a single domain element satisfying the atomic formula `secondary`, and symbol $^{\times}\sigma$ denotes any single non-boundary domain element. I will also use the symbols $\rtimes$ and $\ltimes$ to denote domain elements representing word boundaries.

syllable	stress			
weight	none	secondary	primary	unspecified
light	L	$_{\shortmid}$L	$^{\shortmid}$L	$^{\times}$L
heavy	H	$_{\shortmid}$H	$^{\shortmid}$H	$^{\times}$H
unspecified	σ	$_{\shortmid}\sigma$	$^{\shortmid}\sigma$	$^{\times}\sigma$

Table 16.1: Notation for factors of $\mathfrak{R}$-structures of size 1.

To refer to factors larger than size 1, I use the following notation for the order relations. Two factors x, y of size 1 related by the successor relation will be expressed with concatenation as xy, and two elements related by the precedence relation will be expressed with "$..$" as $x \,..\, y$. Figure 16.1 illustrates two factors of size 2 and the notation used for them. In Figure 16.1, the factor $^{\times}$L$^{\times}$L picks out structures with two adjacent light syllables, irrespective of their stress levels. The factor $_{\shortmid}\sigma \,..\, _{\shortmid}\sigma$ picks out structures with two primary stressed syllables, irrespective of their syllable weight.

Figure 16.1: Example factors of size 2.

With this notation in place, it is straightforward to use a propositional logic over factors of $\mathfrak{R}$-structures as described in Chapter 5. In fact one of the primary conclusions of this chapter is that virtually all nonlexical stress

patterns in the world's languages can be expressed with a propositional logic over factors of $\mathfrak{R}$-structures. Indeed, with few exceptions most of them belong to a restricted propositional logic such as the conjunction of negative literals.

This text adopts a Socratic approach to describing stress patterns. As such, individual stress patterns will be described multiple ways throughout, and descriptions will be modified as important considerations are introduced. The final complete version of each stress pattern will be indicated with a star: ★.

16.2 Multifaceted Descriptive Methodology

An initial description of a pattern need not be the simplest possible, only accurate. A simpler equivalent description can be constructed later. For a concrete example, let us consider the stress pattern of Khmer (Cambodian) as described in StressTyp2:

1. (a) In words of all sizes, primary stress falls on the final syllable.
   (b) In words of all sizes, secondary stress falls on all heavy syllables.
   (c) Light syllables occur only immediately following heavy syllables.
   (d) Light monosyllables do not occur.

The constraints described by (1a) and (1d) are trivial to write with propositional logic: $\text{ˈ}\sigma\ltimes$ and $\neg\rtimes^{\times}\text{L}\ltimes$, respectively. For (1b) one must ask what this truly means. If the final syllable of a word is heavy, does it bear primary or secondary stress? As evidenced by words such as [ɗɗɑŋˈhaəm] ("breath") a final heavy syllable bears primary stress. In other words, constraint (1b) states a restriction on the alphabet, where heavy syllables are never unstressed and light syllables never have secondary stress: $\neg\text{H} \wedge \neg_{\text{ˌ}}\text{L}$. Finally, (1c) splits into two constraints: $\neg\rtimes^{\times}\text{L} \wedge \neg^{\times}\text{L}^{\times}\text{L}$. Then it seems that a reasonable description of this stress pattern is given by the following locally testable expression shown in Equation 16.2.

$$\text{ˈ}\sigma\ltimes \wedge \neg\rtimes^{\times}\text{L}\ltimes \wedge \neg\text{H} \wedge \neg_{\text{ˌ}}\text{L} \wedge \neg\rtimes^{\times}\text{L} \wedge \neg^{\times}\text{L}^{\times}\text{L}. \tag{16.2}$$

This expression can be converted to a finite-state automaton. I used the Language Toolkit software and its associated program plebby (Lambert,

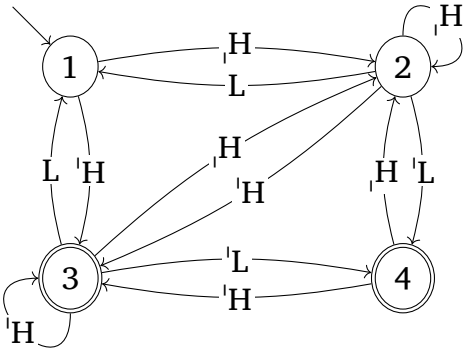


Figure 16.2: An initial attempt to describe the stress pattern of Khmer.

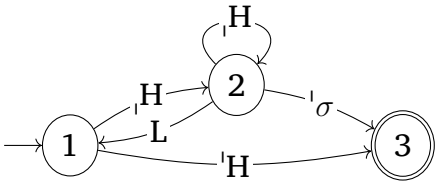


Figure 16.3: The stress pattern of Khmer as described in StressTyp2.

2024).[2] The result is shown in Figure 16.2. I compared this automaton against the one provided in the StressTyp2 database to verify that they are isomorphic. But they are not! The true description is shown in Figure 16.3. What happened?

16.2.1 Universal Constraints

Some constraints are so widespread that one may not even think to mention them. The automaton of Figure 16.2 generates every word that the correct pattern does, but it also includes several illicit words such as those of the form ˌHˈHˈH. One commonly proposed universal constraint is that each word contains one and only one syllable with primary stress (Hyman, 2009). Hyman further suggests that this be split into two constraints: obligatoriness, requiring each word to contain some syllable with primary stress, and culminativity, prohibiting multiple such syllables. Obligatoriness is covered by (1a), but culminativity must be added to the description: $\neg{}^{\text{ˈ}}\sigma \; . \; . \; {}^{\text{ˈ}}\sigma$.

[2] Available at `https://hackage.haskell.org/package/language-toolkit`.

Equation 16.3 shows the updated description.

$$^{\shortmid}\sigma \ltimes \wedge \neg \rtimes^{\times} \mathrm{L} \ltimes \wedge \neg \mathrm{H} \wedge \neg_{\shortmid} \mathrm{L} \wedge \neg \rtimes^{\times} \mathrm{L} \wedge \neg^{\times} \mathrm{L}^{\times} \mathrm{L} \wedge \neg^{\shortmid}\sigma \,..\, ^{\shortmid}\sigma. \tag{16.3}$$

When the logical expression in Equation 16.3 is converted to an automaton, we obtain the expected result, shown in Figure 16.3.

An important lesson in this exercise is that a complete formal description of a pattern must account for not only those constraints that differentiate it from others, but also the atmosphere of putative universals in which it sits.

16.2.2 Identifying Differences with Differences

The previous discussion focused on descriptions that are incomplete due to the existence of unstated, implied constraints, but this is not the only possible source of incompleteness. There are two ways in which a description of a formal language can fail to capture the truth: it might forbid words that are in fact allowed, or it might allow words that are in fact forbidden. Both may even be true at the same time.

The following discussion assumes that both the set to be described, L, and its proposed description, G, are regular, so that equality is decidable and the Boolean operations are computable. Any description G with respect to L can be written as $G \equiv (L \cup A) - B$, where $A \subseteq \complement L$ is the set of words improperly allowed, and $B \subseteq L$ is the set of improperly forbidden words. Figure 16.4 illustrates. Ideally A and B are both empty.

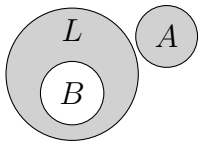


Figure 16.4: The extension of a description G (shaded) for a language L.

It follows then that $L - G = B$. In other words, $L - G$ is exactly the set of words that were forbidden that should have been allowed. If this set is not empty, constraints should be removed from G until it is. Removal of as few constraints as possible is preferred. In the example of Khmer given above, no extra constraints were supplied, but one could imagine any of a myriad of constraints accidentally being applied. Once B is empty, the extension of G is a superset of L.

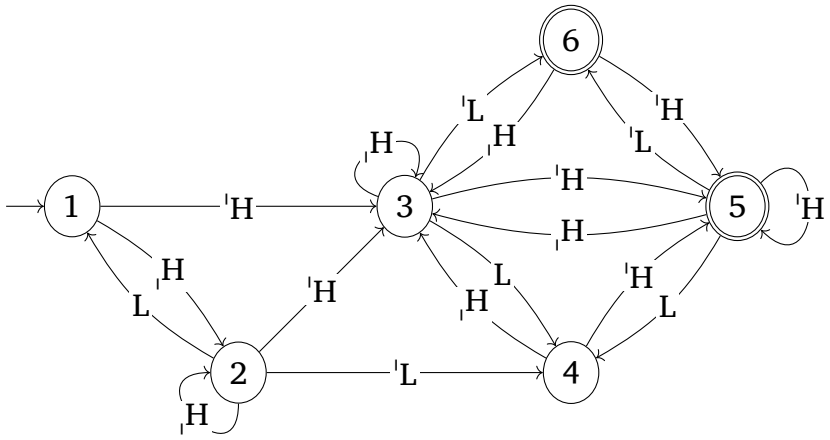


Figure 16.5: Strings that witness that the original description of the stress pattern of Khmer in Equation 16.2 was an overestimate.

On the other hand, $G - L = A$. In other words, $G - L$ is exactly the set of words that were allowed while they should have been forbidden. The necessary constraints to complete the description are then those that forbid these words. Interestingly, a constraint may forbid more than just elements of A, so long as it only forbids elements of $\complement L$. Consider the case where L is the stress pattern of Khmer and G is the initial guess for its grammar in Equation 16.2, the version that did not account for culminativity whose associated automaton is in Figure 16.2. The difference $G - L$ is shown in Figure 16.5, and it appears fairly complicated. Rather than attempting to describe exactly these forms, the culminativity constraint brings simplicity by overlapping with other constraints and ruling out more.

Sometimes the result of differentiating some superset of G from some superset of L is sufficient. The *downward closure* (called the subsequence closure by Rogers and Lambert 2019a) is one kind of superset that has proven useful in separability problems (Zetzsche, 2018). The downward closure of a stringset X, written $\downarrow X$, is the set of all strings formed by deleting zero or more symbols from strings in X. The set formed from $\downarrow G - \downarrow L$ (shown in Figure 16.6) is much easier to describe: it is the set of strings that contain a syllable with primary stress followed by any other syllable. The corresponding constraint is $\neg {}^{\shortmid}\sigma^{\times}\sigma$. Conjoining this constraint to Equation 16.2 yields the equation below.

$$ {}^{\shortmid}\sigma \ltimes \wedge \neg \rtimes^{\times} \mathrm{L} \ltimes \wedge \neg \mathrm{H} \wedge \neg {}_{\shortmid}\mathrm{L} \wedge \neg \rtimes^{\times} \mathrm{L} \wedge \neg^{\times} \mathrm{L}^{\times} \mathrm{L} \wedge \neg {}^{\shortmid}\sigma^{\times}\sigma. \tag{16.4} $$

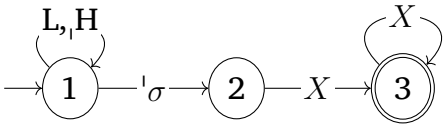


Figure 16.6: A difference of downward closures, $X = \{\text{L}, {}^{\text{ˈ}}\text{L}, {}_{\text{ˌ}}\text{H}, {}^{\text{ˈ}}\text{H}\}$.

The automaton obtained from the logical expression in Equation 16.4 is also equivalent to the expected result shown in Figure 16.3. This form of the expression makes no explicit mention of culminativity, but that constraint is entailed by the ones that are present.

16.2.3 Simplifying Descriptions

At this point two different complete descriptions of the Khmer stress pattern have been shown. Equation 16.3 uses factors with the successor relation and one factor with the precedence relation. Equation 16.4 uses factors with only the successor relation. Is there a way to find a simplest description? Of course this depends on what simple means. This question has also been studied in the context of learning well-formedness patterns, where abductive principles are used to prefer certain descriptions over others (Rawski, 2021). One principle is to prefer more general constraints, and to simplify a description when a more general constraint subsumes a more specific one.

In the two aforementioned equations, both contain redundancy that can be eliminated: if $\rtimes^{\times}\text{L}$ is forbidden, then of course $\rtimes^{\times}\text{L}\ltimes$ must also be forbidden, so the latter constraint can be removed. This is almost a strictly local description. The only part getting in the way of this is the required ${}^{\text{ˈ}}\sigma\ltimes$ constraint. But requiring strings to end with a specific symbol is the same as forbidding ending any other way:

$$ {}^{\text{ˈ}}\sigma\ltimes \equiv \neg\rtimes\ltimes \wedge \neg\sigma\ltimes \wedge \neg{}_{\text{ˌ}}\sigma\ltimes \tag{16.5}$$

By removing the redundancy and making this replacement, a final form of the description emerges, both simple and complete, as shown in Equation 16.6. If desired, the constraints forbidding single symbols can also be removed by choosing an appropriate alphabet that does not contain those symbols.

★ $$\neg \rtimes \ltimes \wedge \neg \sigma \ltimes \wedge \neg_{\shortmid} \sigma \ltimes \wedge \neg \mathrm{H} \wedge \neg_{\shortmid} \mathrm{L} \wedge \neg \rtimes {}^{\times}\mathrm{L} \wedge \neg {}^{\times}\mathrm{L}^{\times}\mathrm{L} \wedge \neg {}^{\shortmid}\sigma^{\times}\sigma \quad (16.6)$$

In general, a constraint can be excluded if it is guaranteed by the set of remaining constraints. A mechanism capable of generating finite-state acceptors from these expressions can systematically test for this condition and return formulas with less redundancy. However, proposing different constraints entirely is not always possible.

Finally, Equation 16.6 is a conjunction of negative literals, and all the literals are factors using the successor relation. Therefore, as discussed in Chapter 5, the stress pattern of Khmer is a strictly local pattern.

The next section details some methods for automatically extracting constraints from a given automaton.

16.3 Partial Factoring of Regular Patterns

Some kinds of constraints can be extracted algorithmically from formal descriptions of a pattern. While the last section developed a description of the stress pattern of Khmer in a propositional logic beginning with a prose description, this section will discuss some techniques for developing a description of stress patterns in a propositional logic beginning with a formal description in terms of a finite-state automaton. What follows is a compressed and reordered discussion of the methods used by Rogers and Lambert (2019a), slightly expanded with new results.

16.3.1 Strictly Piecewise Constraints: Factoring via Downward Closures

Heinz (2014) draws attention to the role strictly piecewise languages can play in describing stress patterns. The strictly piecewise stringsets are exactly those that are closed under deletion of symbols (Rogers *et al.*, 2010). The downward closure of a set is then its smallest strictly piecewise superset.

Zetzsche (2018) points out that downward closures are effectively constructible from several classes, including higher-order push-down automata and recursive schemes. This section however will address only construction

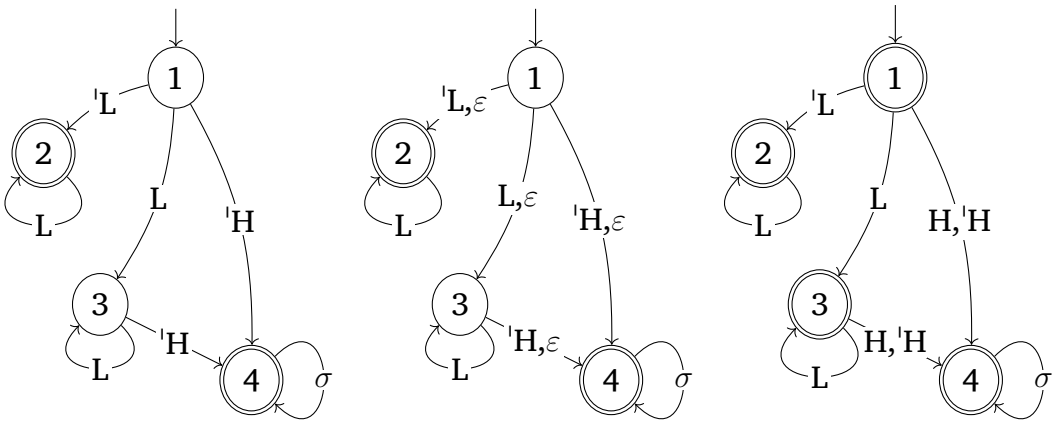


Figure 16.7: Constructing the downward closure of the stress pattern of Amele. At left is the stress pattern itself. In the center is the nondeterministic automaton constructed for the downward closure. And at right is the deterministic version of the same.

from finite-state acceptors representing regular stringsets. In particular, the finite-state automaton representation of a regular language admits an efficient construction of the downward closure, linear time in the number of edges of the automaton: simply add an empty transition in parallel with every existing edge. Loops from a state to itself can be ignored in this process, as reachability via the empty string is already guaranteed in that case. The *powerset construction* of Rabin and Scott (1959) is one way to obtain determinism again if that is necessary.

Figure 16.7 demonstrates this on the stress pattern of Amele as described in StressTyp2. This stress pattern is an example of a "leftmost heavy otherwise leftmost" unbounded stress pattern, where stress falls on the leftmost heavy syllable in words with heavy syllables and on the initial syllable in words without heavy syllables.

Every subsequence that appears in $\downarrow X$ also appears in X and vice-versa, so the two sets have the same set of forbidden subsequences. However, the downward closure is strictly piecewise. Given a representation of $\downarrow X$ as a complete, minimal, deterministic automaton whose nonaccepting sink, if any, is labeled $\perp$, a sufficient set of forbidden factors is the set of strings that label acyclic paths from the initial state to $\perp$ (Rogers and Lambert, 2019a). For the stress pattern of Amele, the resulting constraints are shown in Equation 16.7. (As there is no secondary stress, edges labeled "H,ˈH" are

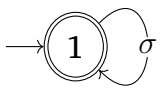


Figure 16.8: What remains once the stress pattern of Amele is subtracted from its own downward closure.

simplified to ˣH.)

$$\neg_{\text{ˈ}}\sigma \wedge \neg \text{L} \,..\, {}^{\text{ˈ}}\text{L} \wedge \neg \text{L} \,..\, {}^{\times}\text{H} \,..\, {}^{\text{ˈ}}\sigma \wedge \neg {}^{\text{ˈ}}\text{L} \,..\, {}^{\text{ˈ}}\text{L} \wedge \neg {}^{\text{ˈ}}\text{L} \,..\, {}^{\times}\text{H} \wedge \neg {}^{\times}\text{H} \,..\, {}^{\text{ˈ}}\sigma \tag{16.7}$$

Some of these can be grouped for simplicity. With no secondary stress and with $\neg \text{L} \,..\, {}^{\text{ˈ}}\text{L}$, $\neg {}^{\text{ˈ}}\text{L} \,..\, {}^{\text{ˈ}}\text{L}$, and $\neg {}^{\times}\text{H} \,..\, {}^{\text{ˈ}}\sigma$, the overall effect is that no syllable precedes a stressed light syllable: $\neg {}^{\times}\sigma \,..\, {}^{\text{ˈ}}\text{L}$. Further, because the final conjunct ($\neg {}^{\times}\text{H} \,..\, {}^{\text{ˈ}}\sigma$) states that primary stress cannot appear after a heavy syllable, the earlier conjunct forbidding $\text{L} \,..\, {}^{\times}\text{H} \,..\, {}^{\text{ˈ}}\sigma$ is redundant and can be removed. After combining the constraints on light syllables with primary stress and filtering away the redundant constraints, the result is a smaller description as shown in Equation 16.8.

$$\neg_{\text{ˈ}}\sigma \wedge \neg {}^{\times}\sigma \,..\, {}^{\text{ˈ}}\text{L} \wedge \neg {}^{\text{ˈ}}\text{L} \,..\, {}^{\times}\text{H} \wedge \neg {}^{\times}\text{H} \,..\, {}^{\text{ˈ}}\sigma \tag{16.8}$$

But this is only a description of the downward closure. Again we can identify the differences with differences: if L is the actual stress pattern of Amele, then $\downarrow L - L$ is the set of strings that are improperly accepted by $\downarrow L$. This difference, which can be computed, is shown in Figure 16.8. It says that the improperly accepted strings are those which contain only unstressed syllables. To avoid accepting these, some stress must be required, and alongside the fact that secondary stress does not appear, the result is obligatoriness: primary stress is required. This is the complement of a strictly piecewise pattern: ${}^{\text{ˈ}}\sigma$. Adding this yields Equation 16.9.

★
$$ {}^{\text{ˈ}}\sigma \wedge \neg_{\text{ˈ}}\sigma \wedge \neg {}^{\times}\sigma \,..\, {}^{\text{ˈ}}\text{L} \wedge \neg {}^{\text{ˈ}}\text{L} \,..\, {}^{\times}\text{H} \wedge \neg {}^{\times}\text{H} \,..\, {}^{\text{ˈ}}\sigma \tag{16.9}$$

This description is equivalent the automaton provided, as can be verified by the Language Toolkit software. It is also equivalent to the prose description from StressTyp2: “In words of all sizes, primary stress falls on the left-most heavy syllable, else on the initial syllable.” In this case, the formal logical description was devised only using the description of the finite-state automaton, without consulting the prose description.

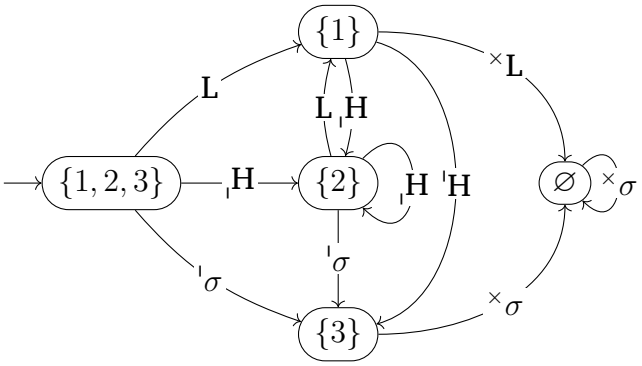


Figure 16.9: The powerset graph for Khmer.

16.3.2 Strictly Local Constraints: The Powerset Graph

Unlike the strictly piecewise constraints, strictly local constraints are not derived solely from finite-state acceptors. Rather they can be derived from the *powerset graph*, which is essentially the same powerset construction as used in determinization, except that the initial state from which it builds is the set of all states. For instance, Figure 16.9 shows the powerset graph of Khmer. The strictly local stringsets are those for which the powerset graph contains no cycles that include states labeled by a set containing more than a single element (Caron, 2000). In Figure 16.9, every cycle only encounters states which are labeled by a single element. Thus, Khmer is strictly local, an observation made previously because its logical description was a conjunction of negative literals where the factors were based on the successor relation. Notice that while neither obligatoriness nor culminativity can explicitly included in such a description (as they are not strictly local constraints), both are guaranteed by the interactions of the constraints that do appear.

A strictly local grammar is composed of four components: free forbidden factors (which cannot occur anywhere), forbidden suffixes, forbidden prefixes, and forbidden words. How each of these can be identified from the powerset graph is taken in turn.

Free Forbidden Factors

A sufficient set of free forbidden factors is the set of strings that label acyclic paths from the initial state (labeled by the set of all states) to the

state labeled by the empty set. In general, the result will include several redundant factors; for example in the Khmer example here, both $\text{L}^{\times}\text{L}$ and $_{|}\text{HL}^{\times}\text{L}$ label such paths, but if the former is forbidden then so too must be the latter. The two nonredundant constraints are $\neg\text{L}^{\times}\text{L}$ and $\neg^{|}\sigma^{\times}\sigma$. The $\neg\text{H}$ and $\neg_{|}\text{L}$ constraints are not included simply because these syllable types were not included in the universe of available symbols. Aside from that detail, these represent exactly the nonanchored forbidden factors from the final description of Khmer.

Forbidden Suffixes

The forbidden suffixes are computed in exactly the same way, except that the paths are not to the state labeled by the empty set, but instead to states labeled by non-empty sets that contain no accepting state. While the state labeled by the empty set also contains no accepting state, it is redundant to consider free forbidden factors as forbidden suffixes, so this state is omitted. In this case, the set of accepting states is $\{3\}$, so the relevant paths are those that end in states $\{1\}$ and $\{2\}$. Exactly as expected, we find two such paths that are not redundant: $\neg\text{L}\ltimes$ and $\neg_{|}\text{H}\ltimes$. There are other paths that produce more forbidden suffixes, such as $\neg\text{L}_{|}\text{H}\ltimes$, but they are redundant due to containing a shorter forbidden suffix, such as $\neg_{|}\text{H}\ltimes$ in this example.

Forbidden Prefixes

The forbidden prefixes of a stringset are merely the forbidden suffixes of its reversal. Construct the reversal of an automaton by swapping the directionality of its edges and exchanging the sets of initial and final states. Then, after canonicalizing the result and constructing the new powerset graph, one can verify that $\neg\rtimes^{\times}\text{L}$ is the only non-redundant forbidden prefix that appears.

Forbidden Words

Unlike the other components, the forbidden words are computed from the automaton itself. Let k be the maximum of the number of states in the automaton, the length of the longest free forbidden factor, and one more than the length of the longest forbidden prefix or suffix. Any word of length at most $k-2$ that is not accepted by the automaton constitutes a forbidden

word. These can be computed by a bounded depth- or breadth-first search on the automaton itself. The only nonredundant constraint discovered in this way is $\neg \rtimes \ltimes$. And at this point, each of the constraints of Khmer has been automatically extracted.

Extraction from non-Strictly Local Regular Languages

For a stringset that is not strictly local, all of these methods will still work but some constraints may be missed. In fact, if a stringset is not strictly local then there are infinitely many forbidden factors. If the powerset graphs are first made reverse-deterministic, i.e. it is reversed, determinized, and then reversed again, then a larger set of constraints will be collected, resulting in a smaller, tighter superset. This reverse-determinism results in each state being labeled not by a set of states but by a set of sets of states; any set containing the set of all states is treated as such. From this point, extraction proceeds as normal. If a strictly local approximation of the original is desired, it can be constructed from the extracted grammar.

As shown with Amele in the discussion of piecewise constraints, the difference between the overestimating approximation and the original may be factorable. If this is the case, then a more complete factorization is made by accounting for this difference.

16.3.3 Tier-based Strictly Local: Extending SL Results

The tier-based strictly local languages are factored in exactly the same way as their non-tier-based counterparts, with one exception. As shown by Lambert and Rogers (2020), the nonsalient symbols are all those that label self-loops on every state. These symbols must be discarded prior to factoring. See also Lambert (2023).

Many stringsets that include tier-based constraints also include others that may interact in such a way that the relevant tier of salient symbols is obscured. For instance, the one-stress constraint (Hyman, 2009) is strictly local over the primary-stress tier, but if a language includes both this constraint and a stress-alternation requirement, then all symbols are relevant to the pattern as a whole.

16.4 Discussion

The pattern building and factorization methods shown in this chapter have all been implemented in an open-source tool by the present author: The Language Toolkit (Lambert, 2024). It has three three components relevant to the methods discussed here: a Haskell library (`LTK`) with which other programs can use these tools, an interactive environment (`plebby`) for constructing and comparing formal descriptions, and a batch processor for factoring patterns (`factorize`).

Rogers and Lambert (2019a) factored the 106 lects in the StressTyp2 database that have finite-state acceptors associated with them with the automatic methods described in this chapter. Of these, 81 (76.4%) were strictly local with factor widths ranging from 2 to 6. Beyond these, 17 more lects (for a total of 92.5%) were able to be automatically factored into a combination of a strictly local part, a strictly piecewise part, and a part whose complement is strictly local. The latter is often merely obligatory stress, which can be captured with tier-based strict locality as well. Two more were properly regular, due to a hidden alternation pattern reminiscent of the phonological foot. And finally the six remaining subregular patterns were determined to require constraints of the form "if x occurs, ending on y is forbidden". This locally testable constraint is also strictly piecewise-local.

This lends credence to the hypothesis that stress, and possibly phonotactics more broadly, is contained within the (co)strict classes of the subregular hierarchy. The only two patterns in the database that would provide counterexamples to this are stress alternation patterns in which secondary stress does not surface. This appears in descriptions of Cyrenaican Bedouin Arabic and Negev Bedouin Arabic. If it could be shown that some form of secondary stress does in fact surface (cf. Becker, 2022), then the pressure would be off.

The only syllable type that appears in every lect is the unstressed light syllable. When considering constraints only in their fully grounded forms (where all sets have been multiplied out into separate constraints) there were:

- 904 distinct free forbidden factors of width at least 2,
- 35 distinct forbidden prefixes,
- 230 distinct forbidden suffixes,

- 44 distinct forbidden words, and

- 126 distinct forbidden subsequences.

No pattern requires a factor width of more than 6, while most require only between 2 and 4. This is a large number of factors, but applying the strategies of Chandlee *et al.* (2019) might reduce this to a more manageable set.

A constraint is satisfied by a pattern if it is redundant given the latter. Using this, it is possible to determine whether putative universals are indeed universal. For instance, the one-stress constraint as separated into obligatoriness and culminativity was tested. While culminativity was satisfied by every lect in the database and may truly be universal, neither Seneca nor Cayuga satisfy obligatoriness. Hyman (2009) confirms this.

This work thus demonstrates the utility of formal descriptions in classifying and comparing languages and in verifying universality of constraints. It provides both motivation and means for working with these descriptions.

16.5 Conclusion

This chapter studied the extent to which non-lexical stress patterns can be described with propositional logics. It showed how such descriptions can arise from careful analysis of a prose description, and by automatic methods from a formal description, such as those provided by finite-state automata. Along the way, I discussed general principles and methods for choosing among possible constraints.

It is striking that with apart from a couple of examples meriting further study, all of the patterns can be expressed with propositional logic. Furthermore, nearly all of those can be expressed with a conjunction of negative literals, modulo the positive literal requiring words to contain a primary stress (obligatoriness). Altogether, this work supports the hypothesis that well-formedness of phonological representations can be decided with computations that do not rise to the level of first-order logic.

Acknowledgments

This chapter is based heavily on work done with James Rogers, which itself was built on the work of many others. The author would especially like to extend thanks to the alumni of Earlham College who during their time there kept the Theory Group alive, as well as to Jeff Heinz and his students and colleagues from both the University of Delaware and Stony Brook University.

Chapter 17

A Formal Analysis of Correspondence Theory

Amanda Payne and Mai Ha Vu

17.1 Introduction

An influential version of Optimality Theory (OT) adopts Correspondence Theory (McCarthy and Prince, 1995), which explicitly invokes a correspondence relation between the elements in an Underlying Representation (UR) and Surface Representation (SR). This relation allows Correspondence Theory to define a number of faithfulness constraints which help account for a variety of segmental and prosodic phenomena. It follows that understanding Correspondence Theory in light of computational complexity is important. For instance, if it is within a manageable level of computational complexity, it would lend credence to OT being a psychologically plausible phonological theory.

This chapter provides a computational analysis of the complexity of Correspondence Theory. This analysis is stated in terms of Monadic Second Order (MSO) logic and First Order (FO) logic. We assume some familiarity with these logics and in particular the fact that MSO logic is strictly more powerful than FO logic. For example, it is known that MSO logic over a vocabulary for strings with the precedence relation defines exactly the *regular* stringsets (Büchi, 1960), which are the same as the stringsets recognizable by a finite-state automaton (FSA). On the other hand, as dis-

cussed in Chapter 2, FO logic over the same vocabulary defines exactly the *star-free* stringsets (McNaughton and Papert, 1971). It is no accident that the star-free formal languages are a proper subset of the regular languages; the logic underlying the former is more powerful than the logic underlying the latter.

We begin the analysis of Correspondence Theory with the function in OT that generates candidates, the GEN function. We study two different interpretations of GEN: 1) as a *function* that maps an arbitrary underlying phonological form to an infinite set of UR-SR pairs, each of which stands in a correspondence relation and 2) as a *representation* of this infinite set of UR-SR pairs, each of which is in a correspondence relation, obtained from a *given* UR.

We first find that the GEN function is not MSO-definable. One interpretation of this result is that Correspondence Theory is more powerful than the rest of phonological theory, since phonological constraints and maps have been argued to be within the regular region (Kaplan and Kay, 1994) and MSO-definability is associated with this region. This is in line with previous studies that have shown that OT at its full capacity is not *finite-state*, and therefore it is not regular (Frank and Satta, 1998). Second, we show that the *formal language* consisting of the UR-SR Correspondence-theoretic candidates for a *given* underlying representation is FO-definable. In other words, we can characterize the output by GEN for a given UR with FO logic. This result is quite distinct from the first one, and highlights the importance of careful interpretation of complexity results.

Finally, we present evidence from case studies that the optimal candidate from a *given* UR can be determined with FO-definable, language-specific, inviolable constraints, without recourse to optimization. Moreover, these case studies tentatively suggest that even weaker logics may be sufficient here, which is inline with the research program that argues that phonological patterns are subregular (Chandlee, 2014; Lai, 2015; Jardine, 2016; Payne, 2017; Luo, 2017; Heinz, 2018; Strother-Garcia, 2018a, 2019; Dolatian, 2020a; Rawski, 2021; Lambert, 2022).

One way to interpret this result is that reliance on language-specific, inviolable constraints obviates the need for optimization without increasing computational complexity. However, we acknowledge that validity of this interpretation only extends insofar as the case studies are representative of phonological phenomena generally.

In sum, our work shows that the representations promoted in Correspon-

dence Theory–that elements in the UR correspond to elements in the SR–are fairly restrictive computationally, but that generating the set of candidates for an arbitrary UR is not. On the other hand, a description of the output of the GEN function for a given UR is something that can be described with first-order logic. Moreover, we show that it is possible to pick the correct SR for a given UR simply using language-specific, inviolable constraints instead of the violable, ranked constraints used in OT.

Our approach is similar in spirit to Potts and Pullum (2002), which uses logic to formalize constraints. The major difference is that our third result suggests the complexity stays low even with language-specific, inviolable constraints, cf. Scobbie *et al.* (1996); Jardine (2017) and Chapters 15 and 16, this volume. In this regard, our analysis suggests that there is merit to the representations invoked in Correspondence Theory beyond the role they play in Optimality Theory.

The remainder of this chapter is organized as follows. §17.3 presents the first two results, that traditional Correspondence Theory's GEN function is not MSO-definable but that the input-output candidates from a given input are MSO-definable, and in fact, FO-definable. In §17.4 we provide an FO-definable schema for phonological correspondences and illustrate it with some case studies meant to showcase its flexibility. §17.5 concludes with some additional discussion.

17.2 Background to Optimality Theory

Phonological theories seek to study the nature of URs, their corresponding SRs, and the process that 'transforms' URs into their corresponding SRs. This chapter focuses on the Correspondence-theoretic version of OT (McCarthy and Prince, 1995). The transformational component of a phonological grammar in OT is formalized with two components, GEN and EVAL. GEN is the function that generates all possible UR-SR candidates for any given UR, where elements of the UR and SR are in correspondence relations to each other, and EVAL is the function that chooses the correct UR-SR candidate via optimization.

We illustrate them with a simple phonological example, final devoicing, which specifies that final stops must be voiceless. This means that for an arbitrary UR, such as /kad/, the correct SR should be [kat]. Applying GEN to /kad/, GEN(/kad/) would generate a set of UR-SR candidates,

such as (/kad/, [kat], $\mathcal{R}$) or (/kad/, [kad], $\mathcal{R}$), where $\mathcal{R}$ represents the correspondence relations between the UR /kad/ and the SRs [kat] or [kad]. We will come back to a more formal definition of GEN in the next section. EVAL then takes the set of candidates generated by GEN, and selects the optimal UR-SR candidate based on the grammar, which is a series of ranked constraints. The general idea is that the optimal candidate is the one that violates the highest ranked constraints the least.

For languages with word-final devoicing, the relevant constraints are IDENT[VOICE], which stipulates that every SR segment has to be identical in voicing to its corresponding UR segment, and *[−sonorant,+voice]]$_{wd}$, which is violated whenever there is a voiced obstruent in word-final position. For EVAL to pick the correct SR in a language with a word-final devoicing rule, *[−sonorant,+voice]]]$_{wd}$ must be ranked higher than IDENT[VOICE] as shown in Table 17.1. A complete analysis of course re-

		*[−sonorant,+voice]]$_{wd}$	IDENT[VOICE]
a.	/kad/,[kad]	*	
b. ☞	/kad/,[kat]		*

Table 17.1: OT tableau for word-final devoicing given the UR /kad/, *[−sonorant,+voice]]$_{wd}$ must outrank IDENT.

quires discussion of additional candidates and constraints (see Chapter 1.3). For example, GEN also generates a candidate (/kad/, [kan], $\mathcal{R}$), which violates IDENT[NASAL], the constraint that stipulates that every SR segment has to be identical in nasality to its corresponding UR segment. In this analysis, IDENT[NASAL] outranks IDENT[VOICE].

In the next section we formalize GEN as a function based on McCarthy and Prince (1995), provide an alternative way to interpret GEN, and then analyze both interpretations in terms of their computational and logical complexity.

17.3 Correspondence Theory

McCarthy and Prince (1995) characterize GEN as a function which maps an underlying form A to a set of ordered triples. Each triple is a candidate.

The first element of the triple is the underlying form as a string. The second element is the surface form as a string. The third element is the correspondence relation which relates elements in the underlying string to elements in the surface string. An example is given in 17.1.

$$\begin{aligned} Gen(/abc/) \rightarrow \{&(/abc/, [abc], \mathcal{R}_1), \\ &(/abc/, [abcd], \mathcal{R}_2), \\ &(/abc/, [abcd], \mathcal{R}_3), \ldots\} \end{aligned} \tag{17.1}$$

Mathematically, each $\mathcal{R}$ is a subset of $N \times M$ where $N = \{1, 2, \ldots n\}$ with n the length of the UR and $M = \{1, 2, \ldots m\}$ with m the length of the SR. In classic Optimality Theory, the set of candidates is infinite in size because there is no upper bound on the length of the output string. Of course, candidates whose output strings have many epenthesized segments are typically harmonically bounded and would not be selected as the optimal candidate.

17.3.1 Representing Candidates

We model candidates with *relational structures* (Chapter 2). This approach has its roots in finite model theory (Enderton, 2001; Hedman, 2004; Courcelle and Engelfriet, 2012). We call these relational structures *correspondence* structures.

To model Correspondence Theory, we only need unary and binary relations in the signature of the model. These correspondence structures extend the relational structures for words, so we begin with those.

As discussed in Chapter 2, there are multiple models for words. Two important choices are the order relation, and whether the unary relations designate individual segments or properties of those segments. In this chapter we use the precedence relation ($<$) for order, and observe that the successor relation is FO-definable from precedence as shown in Equation 17.2.

$$(x \lhd y) = x < y \wedge \neg(\exists z)[x < z \wedge z < y] \tag{17.2}$$

Regarding the unary relations which indicate the make-up of a phonological segment, one possibility is to have a unary relation for each symbol in the string. Thus, if *a* is a symbol in a string, there would be a unary relation `a` and elements in the domain could stand in this relation to indicate they

represent such symbols. Another, more phonologically motivated approach is to let the unary relations stand for phonological features. In this approach, the unary relations indicate phonetic properties such as voicing, place and manner. In this chapter, we generally adopt the feature-based approach. In our examples we use the basic phonetic properties, as was the case in Chapter 2 (2.5) with the same caveat that we are not expressing a belief that this is the 'right' system of features. It is merely convenient for us to explain our main points which do not depend on the specifics of the feature system. While we adopt the unary relations as features, we also avail ourselves of segments as needed (for example in diagrams). The primary reason for this is convenience, but it is also worth reminding readers that predicates for features are FO-definable from segments and that predicates for segments are FO-definable from features as discussed in Chapter 2.

To model correspondence structures, we need only add two more unary relations and one more binary relation. These unary relations indicate whether the node belongs to the underlying string (`u`) or the surface string (`s`). The binary indicates whether two domain elements stand in the correspondence relation (`c`).

Relational structures can be displayed visually as graphs where the domain elements are nodes of the graph. In a departure from the way these diagrams are displayed in other chapters, the unary relations denoting the properties of each domain element are expressed as labels of the nodes themselves. In particular, we label nodes with symbols denoting phonological segments which carry a u or s subscript denoting whether it belongs to the UR or SR, respectively. The indices for the domain elements are displayed adjacent to the nodes. Binary relations between elements in the structure are drawn as edges connecting nodes, and these edges are labeled to indicate which relation it represents.

We illustrate relational structures with three UR-SR candidates in Figures 17.1-17.3 below. The candidate in Figure 17.1 violates MAX (every UR segment has to have a corresponding SR segment). The candidate in Figure 17.2 violates INTEGRITY (each SR segment must correspond to at most one UR segment), UNIFORMITY (each SR segment must correspond to at most one UR segment), and IDENT. The candidate in Figure 17.3 violates DEP (every SR segment has to have a corresponding UR segment), and IDENT.

Note that in the interest of readability, instead of precedence, we use the edges in the graphs to show the successor relation.

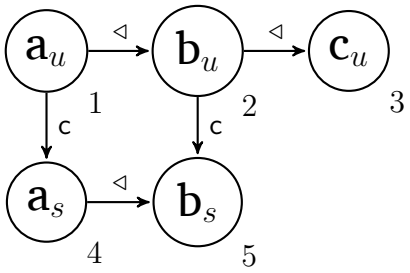


Figure 17.1: A candidate that violates the MAX constraint, as the segment 'c' in the UR has no correspondent in the SR.

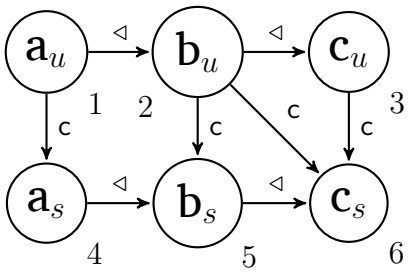


Figure 17.2: A candidate that violates the INTEGRITY and UNIFORMITY constraints, as the 'b' segment in the UR corresponds to two segments in the SR, and the 'c' segment in the SR corresponds to two segments in the UR.

17.3.2 The GEN function

The GEN function is *not* MSO-definable. The core reason is stated in Courcelle and Engelfriet (2012) as Fact 1.37, where U is an arbitrary input string and S is a set of candidates: "For every monadic second-order transduction f there exists an integer k such that, if f transforms a relational structure U into a relational structure S, then $|S| \leq k \cdot |U|$."

In other words, the size of the relational structure must be bounded by some integer $k \times n$, where n is the size of the input and k is fixed in advance. Here we do not explicitly provide a relational structure for a *set* of candidates, but we do not need to. Since there is no upper bound on the length of the surface string in a candidate, there will always be some candidate larger than $k \times n$ and hence any set containing it will also be too large. Consequently, the output of GEN must be finite in size for it to

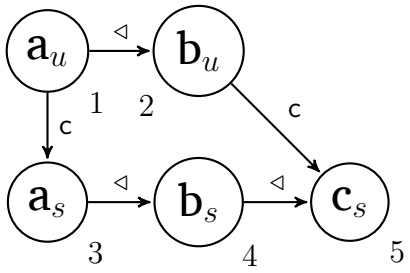


Figure 17.3: A candidate that violates the DEP and IDENT constraints, as the 'b' segment in the SR does not have a corresponding segment in the UR, and the 'b' segment in the UR corresponds to a segment in the UR which is not 'b.'

be an MSO definable function. But since the set of candidates output by GEN in classic OT is generally assumed to be infinite in size, it cannot be described with any MSO-definable *function*.

To illustrate, consider that segments in the UR can have zero, one, or multiple corresponding segments in the SR, and these segments may or may not be identical. Without constraints on GEN, there is no bound on the size of the SR. Some example candidates are shown in Equation 17.3.

$$\begin{aligned} /\mathrm{a}_1\mathrm{b}_2\mathrm{c}_3/ &\rightarrow \mathrm{a}_1\mathrm{b}_2 \\ /\mathrm{a}_1\mathrm{b}_2\mathrm{c}_3/ &\rightarrow \mathrm{a}_1\mathrm{b}_2\mathrm{a}_x\mathrm{c}_3 \\ /\mathrm{a}_1\mathrm{b}_2\mathrm{c}_3/ &\rightarrow \mathrm{d}_1\mathrm{b}_2\mathrm{c}_3 \\ /\mathrm{a}_1\mathrm{b}_2\mathrm{c}_3/ &\rightarrow \mathrm{a}_1\mathrm{b}_2\mathrm{c}_3\mathrm{c}_x\mathrm{c}_x\mathrm{c}_x \end{aligned} \tag{17.3}$$

The subscript x indicates that there is no correspondent in the input. This lack of correspondence could happen for any number of segments, though with increasing numbers of new segments the form may be quite phonologically unnatural. Such candidates, like $\mathrm{a}_1\mathrm{b}_2\mathrm{c}_3\mathrm{c}_x\mathrm{c}_x\mathrm{c}_x$, will typically violate too many faithfulness constraints to ever surface. Nevertheless, because GEN generates all candidates, its output ends up being a single set of an unbounded number of candidates; this makes its output arbitrarily large.

What does this result mean for phonological theory? There are four possibilities.

The first approach is to conceive of GEN not as a function from underlying strings to candidates, but instead as a function from underlying strings to *contenders*. The contenders are the non-harmonically bounded

candidates. Riggle (2004) shows that the contenders typically form a finite set. So, it may be the case that GEN as a function from underlying strings to contenders is MSO-definable.

The second approach is to take this result as a reason to reject classic OT, in favor of an alternative theory like Harmonic Serialism (McCarthy, 2008b) where GEN is a function which maps an underlying string w to a set of candidates, each of which differs at most minimally from w. Restricting GEN in this way might also make it an MSO-definable function. Hao (2019) and Lamont's (2022a) computational analyses of harmonic serialism may shed some light into this question.

The third is that instead of analyzing GEN as a function, we should analyze it as a relation. Although GEN is not a MSO-definable function, it *may* be a MSO-definable *relation*. One way to approach this from a model theoretic perspective may be with infinite structures (Benedikt *et al.*, 2001).

Finally, the fourth is to characterize the infinite set of candidates that is output by a GEN relation. This is the approach we take in the next section, and show that this set is FO-definable.

To summarize, it is interesting to observe that GEN is not a MSO-definable function, and there are several alternatives worth pursuing.

17.3.3 The set of candidates for a given input

In this section we show that if one fixes a particular UR string such as /kætz/, then for this UR, one can provide finitely many sentences in FO logic which are satisfied by all and only the possible UR-SR candidates of that UR. Only five logical statements are needed to describe the infinite set of possible candidates for a given UR. The logical statements, listed in 17.4-17.8, are all described with FO-logic over the relational structures described earlier. Each statement is explained further below. We present the statements first specifically for the UR /kætz/ and then generalize afterwards.

Equation 17.4 says that the precedence relation can only hold between elements on the same tier.

$$(\forall x, y)\Big[x < y \rightarrow [\mathtt{u}(x) \wedge \mathtt{u}(y)] \vee [\mathtt{s}(x) \wedge \mathtt{s}(y)]\Big] \tag{17.4}$$

Equation 17.5 says that correspondence only occurs between elements of different tiers.

$$(\forall x, y)\big[\mathtt{c}(x, y) \rightarrow (\mathtt{u}(x) \wedge \mathtt{s}(y))\big] \tag{17.5}$$

Equation 17.6 says domain elements are either on the UR or SR tier.

$$(\forall x)[\mathtt{u}(x) \vee \mathtt{s}(x) \wedge \neg[\mathtt{u}(x) \wedge \mathtt{s}(x)]] \tag{17.6}$$

Equation 17.7 says that elements on the SR tier form a string.

$$(\forall x, y)[[\mathtt{s}(x) \wedge \mathtt{s}(y)] \rightarrow [x < y \vee y < x]] \tag{17.7}$$

The last equation says that the UR is /kætz/.

$$\begin{aligned}(\exists w, x, y, z)[w < x \wedge w < y \wedge w < z \wedge x < y \wedge x < z \wedge y < z \\ \wedge \mathtt{u}(w) \wedge \mathtt{u}(x) \wedge \mathtt{u}(y) \wedge \mathtt{u}(z) \\ \wedge \mathtt{k}(w) \wedge \mathtt{æ}(x) \wedge \mathtt{t}(y) \wedge \mathtt{z}(z)]\end{aligned} \tag{17.8}$$

Equation 17.4 says that precedence relationships can only be between nodes that belong to the same 'tier', i.e. there cannot be precedence between an underlying and a surface node. Conversely, correspondence edges can only be between nodes that are on different tiers (17.5). All nodes have to either belong to the UR or the SR, so they cannot be both or neither (17.6). The surface tier has to form a string, where the definition of a string is a set of nodes where the precedence relation establishes a linear order (17.7). Lastly, we need a fixed input string and Equation 17.8 provides this. In this example, the input string is the word /kætz/, and it is defined segment by segment.

The graph in Figure 17.4 shows an example of an UR-SR candidate which satisfies these five constraints. There are arbitrarily many other UR-SR candidates that would also do, each of which would have /kætz/ as its UR. Again, for readability, we show the graph with successor relations instead of precedence relations. (Note that this is not the optimal candidate, as we would expect [kæts] to surface, and not [jæz].)

In summary, every relational structure which satisfies the statements 17.4-17.8 is a valid candidate for the underlying string /kætz/.

Equations 17.4-17.7 are the same for any given UR. Only Equation 17.8 depends on the UR. Generally, for any given underlying string $w = a_1 a_2 \ldots a_n$ the fifth statement can be written as follows.

$$(\exists x_1, x_2 \ldots x_n)\Big[\bigwedge_{i<j} x_i < x_j \bigwedge_{1\leq i\leq n} \mathtt{a_i}(x_i) \bigwedge_{1\leq i\leq n} \mathtt{u}(x_i)\Big] \tag{17.9}$$

Equation 17.9 is an abstract template to define a specific string. In 17.9, $\bigwedge(x_i < x_j)_{1\leq i<j\leq n}$ stands for the conjunction of statements $x_i < x_j$,

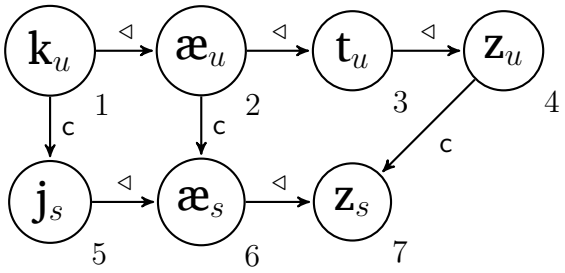


Figure 17.4: A candidate that satisfies all constraints in 17.4 to 17.8.

and says that every node with a lower index precedes every node with a higher index; this guarantees that the nodes form a string shape. Similarly, $\bigwedge(a_i(x_i))_{1\leq i\leq n}$ means a conjunction of n statements; for each i between 1 and n inclusive, the ith statement says x_i stands in the unary relation a_i; that is every ith node is labeled a_i. Finally, $\bigwedge(\mathtt{u}(x_i))_{1\leq i\leq n}$ means each node stands in the unary relation which indicates it is an underlying element.

So far, this analysis has provided the language of all possible UR-SR candidate pairs for a given UR. However, only one of the candidates is selected whose SR is pronounced. In classic OT, the choice of winner is done by the EVAL function, which is universal. In the next section, we consider language-specific EVAL functions, which are defined in terms of language-specific constraints on the well-formedness of correspondence structures. We specifically will deal with examples for simple, common phonological mappings.

17.4 Phonological Maps from FO-constraints over Correspondence Structures

Here we introduce a non-optimizing way for picking the correct candidate. We define language-specific, inviolable FO-constraints that 'filter out' UR-SR candidates. In other words, instead of comparing possible candidates to each other to find the optimal one, the candidates are simply evaluated against a set of inviolable constraints. In the end, only the candidate which satisfies all of these language-specific constraints is selected.

In what follows, we illustrate this approach with examples from phonology that motivated the different faithfulness constraints in Optimality

Theory. In particular, we present language-specific inviolable constraints that would give rise to processes for assimilation, epenthesis, deletion, metathesis, fission, and coalescence.

17.4.1 Assimilation

We show how assimilation can be accounted for with the example of English obstruent devoicing. The generalization is that voiced obstruents become voiceless after voiceless sounds. The desired output for the input /kætz/ is shown in Figure 17.5 – which is the same as the optimal candidate selected by EVAL under a classic OT analysis. In order to state the constraints in

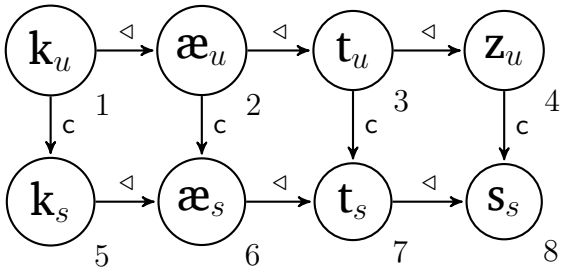


Figure 17.5: The optimal UR-SR candidate for the input /kætz/.

FO-logic, we utilize two predicates: $\texttt{TD}\,(x, y)$ (17.10) identifies a pair of segments where a voiceless sound is followed by a voiced obstruent, and $\texttt{TT}\,(x, y)$ (17.11) identifies a pair of segments where a voiceless sound is followed by a voiceless obstruent within a string.

$$\texttt{TD}\,(x, y) \stackrel{\text{def}}{=} x \lhd y \wedge \texttt{voiceless}(x) \wedge \texttt{voiced}(y) \wedge \texttt{obstruent}(y) \quad (17.10)$$

$$\texttt{TT}\,(x, y) \stackrel{\text{def}}{=} x \lhd y \wedge \texttt{voiceless}(x) \wedge \texttt{voiceless}(y) \wedge \texttt{obstruent}(y) \quad (17.11)$$

The devoicing process can be stated in part with an FO-definable constraint using the implication in Equation 17.12: if the underlying segment is a voiceless sound followed by a voiceless obstruent, then they have to be in correspondence with a voiceless segment and voiceless obstruent, respectively.

$$(\forall x_1, y_1, x_2, y_2)[(\,\texttt{TD}\,(x_1, y_1) \wedge \texttt{u}(x_1) \wedge \texttt{c}(x_1, x_2)) \rightarrow \,\texttt{TT}\,(x_2, y_2)] \quad (17.12)$$

For the devoicing example in Figure 17.5, the variables in Equation 17.12 are satisfied when $x_1 = 3$, $y_1 = 4$, $x_2 = 7$,and $y_2 = 8$.

As in OT, additional constraints are needed to ensure other candidates are disallowed. For example, equations defining constraints similar to UNIFORMITY (17.13), INTEGRITY (17.14), LINEARITY (17.15), MAX (17.16), and DEP (17.17) are shown below.

$$\texttt{Uniformity} \stackrel{\text{def}}{=} (\forall x, y, z)\big[(\texttt{u}(x) \wedge \texttt{c}(x, y) \wedge \texttt{c}(x, z)) \rightarrow y = z\big] \tag{17.13}$$

$$\texttt{Integrity} \stackrel{\text{def}}{=} (\forall x, y, z)\big[(\texttt{s}(y) \wedge \texttt{c}(x, y) \wedge \texttt{c}(z, y)) \rightarrow x = z\big] \tag{17.14}$$

$$\begin{aligned}\texttt{Linearity} \stackrel{\text{def}}{=} &(\forall x_1, x_2)\big[(\texttt{u}(x_1) \wedge \texttt{u}(x_2) \wedge x_1 < x_2 \wedge \texttt{c}(x_1, y_1) \wedge \texttt{c}(x_2, y_2)) \\ &\rightarrow y_1 < y_2\big]\end{aligned} \tag{17.15}$$

$$\texttt{Max} \stackrel{\text{def}}{=} (\forall x)(\exists y)\big[\texttt{u}(x) \rightarrow \texttt{c}(x, y)\big] \tag{17.16}$$

$$\texttt{Dep} \stackrel{\text{def}}{=} (\forall x)(\exists y)\big[s(x) \rightarrow \texttt{c}(y, x)\big] \tag{17.17}$$

We additionally need constraints like IDENT[F] for each feature F except for the features *voiced* and *voiceless*. The general form of the constraint for a feature F which is faithful everywhere is shown in 17.18.

$$\texttt{Ident[F]} \stackrel{\text{def}}{=} (\forall x, y)\big[(\texttt{u}(x) \wedge \texttt{F}(x) \wedge \texttt{c}(x, y)) \rightarrow \texttt{F}(y)\big] \tag{17.18}$$

In the example of English obstruent devoicing, the feature *coronal* is faithful everywhere. The structure in Figure 17.5 above satisfies 17.18. There are two relevant cases: when $x = 3$ and $y = 7$ and when $x = 4$ and $y = 8$. In both instances the y element satisfies `coronal` as 17.18 demands.

As for the features *voiced* and *voiceless*, they must be faithful everywhere except the devoicing environment identified above. In OT, this is accomplished with language-specific constraint rankings, but in the declarative approach here, such environments are specified in language-specific constraints.

$$(\forall x, y)\Big[\big(\texttt{u}(x) \wedge \texttt{voiced}(x) \wedge \texttt{c}(x, y) \wedge \neg(\exists z)[\ \texttt{TD}\ (z, x)]\big) \rightarrow \texttt{voiced}(y)\Big] \tag{17.19}$$

The language-specific, inviolable constraint in Equation 17.19 completes the analysis of the devoicing.

It might be possible to describe the same desired candidate as a conjunction of banned substructures. This would be similar to Jardine's (2017) analysis of tonal mapping, Chapter 15's analysis of syllable structure, and Chapter 16's analysis of non-lexical stress patterns. If so, it would indicate that the logical power needed to state the constraints is actually more restrictive than FO-logic. For instance, the constraint expressed by Equation 17.20 prevents the fully faithful candidate from being selected.

$$\neg(\exists x_1, x_2, y_1, y_2)[\ \texttt{TD}\ (x_1, y_1) \wedge \texttt{u}(x_1) \wedge \texttt{c}(x_1, y_1) \wedge\ \texttt{TD}\ (y_1, y_2)] \qquad (17.20)$$

This constraint can also be expressed in a propositional logic (see Chapter 5) as a factor of a correspondence structure. Figure 17.6 shows a candidate at left and this factor at right. The highlighted part in the candidate shows where the constraint in Equation 17.20 is violated (or equivalently where it matches the banned factor). Several other constraints that forbid different

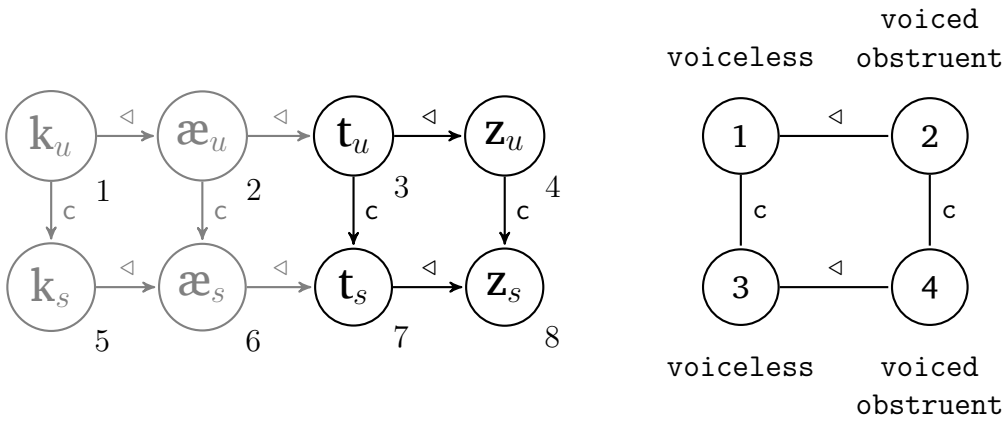


Figure 17.6: Illustration of a banned candidate that violates constraint in 17.20. Equivalently, the candidate contains the forbidden factor shown at right.

types of substructures would of course also be required to ensure that the candidate in Figure 17.5 is the only one that does not contain any forbidden substructures.

It is an open question whether assimilation and every other phonological process can be approached in this way. In this chapter, we leave

this question for future research and continue to express the language-specific constraints on correspondence structures in FO logic following the conditional-style above for the remainder of the case studies.

17.4.2 Epenthesis

One example of epenthesis is Hungarian [j]-insertion to prevent vowel hiatus (Siptar, 2005). We identify the first vowel in the conditioning environment for this process in Equation 17.21: it is the vowel that is succeeded by another vowel.

$$\texttt{VV}\ (x) \stackrel{\text{def}}{=} (\exists y)[x \lhd y \wedge \texttt{vocalic}(x) \wedge \texttt{vocalic}(y)] \tag{17.21}$$

Then the constraint in Equation 17.22 simply states that when there is a VV sequence in the underlying string, there must be a [j] element in the surface string that intervenes between the correspondents of the two underlying vowels. This is also true the other way around: if there is a [j] on the surface without a corresponding underlying element, then it must be the case that underlyingly, there was vowel hiatus.

$$\begin{aligned}\texttt{j\_insertion} \stackrel{\text{def}}{=} (\forall x)\Big[\big(\ \texttt{VV}\ (x) \wedge \texttt{u}(x)\big) \leftrightarrow (\exists y, z)\big[\texttt{c}(x,y) \\ \wedge\, y \lhd z \wedge j(z) \wedge \neg(\exists w)[\texttt{c}(w,z)]\big]\Big]\end{aligned} \tag{17.22}$$

The candidate with the correct SR for the UR /fiu/ is shown in Figure 17.7. Additional constraints are necessary to ensure the rest of the

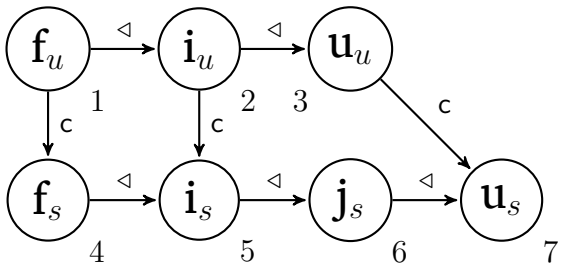


Figure 17.7: Candidate for the UR /fiu/ that satisfies the constraint in 17.22 by j-epenthesis between the vowels.

surface string is faithful. In particular, the constraints `Uniformity`,

Integrity , Linearity , Max , and Ident[F] in Equations 17.13-17.16 and Equation 17.18 also apply. Observe that the constraint j_insertion in Equation 17.22 replaces the constraint Dep (Equation 17.17), which banned epenthesis. Here epenthesis is only permitted in the particular environment specified in Equation 17.22.

17.4.3 Deletion

Tibetan consonant cluster simplification (Halle and Clements, 1983) was studied in Chapter 11. We use it as our example deletion process. The generalization depends on syllable structure, as consonant deletion only applies in onset clusters. We only focus on word initial consonant clusters.

Again, we define word initial consonant clusters very similarly to vowel hiatus in Equation 17.23. They are two successive elements that are both consonants, which are not preceded by any other element.

$$\#\texttt{CC}\,(x) \stackrel{\text{def}}{=} (\exists y)\big[x \lhd y \wedge \texttt{consonant}(x) \wedge \texttt{consonant}(y) \wedge \neg(\exists z)[z \lhd x]\big] \tag{17.23}$$

The next constraint in Equation 17.24 is stated as a biconditional. If there is a consonant cluster in the underlying string then there is no element in the surface string which corresponds to the first underlying consonant. The converse is also true: if there is an element in the underlying form that has no correspondent in the surface, it must be the case that the underlying element is the first consonant in the cluster.

$$\texttt{C\_deletion} \stackrel{\text{def}}{=} (\forall x)\Big[\ \#\texttt{CC}\,(x) \wedge \texttt{u}(x) \leftrightarrow \neg(\exists y)[\texttt{c}(x,y)]\Big] \tag{17.24}$$

Figure 17.8 shows the correct candidate. Note that the underlying /b/ at position 4 does not satisfy the predicate #CC because it is not word-initial. As before, the candidate with the correct surface form has to satisfy other faithfulness constraints given by Uniformity , Integrity , linearity , Dep , and Ident[F] . The constraint Max (Equation 17.16), is replaced by C_deletion (Equation 17.24), which only allows deletion in the conditioning environment.

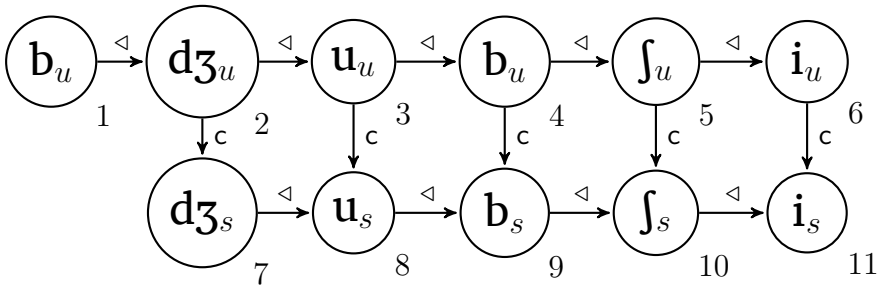


Figure 17.8: Candidate for the UR /bdʒubʃi/ that satisfies the constraint in 17.24 by deleting the first consonant in the onset consonant cluster.

17.4.4 Metathesis

An example of bounded metathesis can be found in Rotuman (Schmidt, 2003). A simplified description of the phenomenon is that in some types of roots, word-final consonant-vowel sequences in the UR surface as vowel-consonant sequences. For example the UR /hosa/ surfaces as [hoas] in the language.

To describe this process, we first identify the first segment in the environment that triggers metathesis. This environment is a word-final consonant-vowel sequence: two successive elements where the first one is a consonant followed by a vowel, and there is no element following them, as in Equation 17.25.

$$\texttt{CV\#}\,(x) \stackrel{\text{def}}{=} (\exists y)\big[x \lhd y \wedge \texttt{consonant}(x) \wedge \texttt{vowel}(y) \wedge \neg(\exists z)[y \lhd z]\big] \quad (17.25)$$

Then we state the constraint as follows. If there is a consonant-vowel sequence word-finally in the UR, then in the SR the element corresponding to the consonant *succeeds* the element corresponding to the vowel.

$$\begin{aligned}\texttt{CV\_metathesis} \stackrel{\text{def}}{=} (\forall x, y)\big[\big[\,\texttt{CV\#}\,(x) \wedge \texttt{u}(x) \wedge x \lhd y\big] \rightarrow \\ (\exists u, v)[\texttt{c}(x, v) \wedge \texttt{c}(y, u) \wedge u \lhd v]\big]\end{aligned} \quad (17.26)$$

Figure 17.9 shows the correct candidate.

As before, the other faithfulness constraints, except for `Linearity`, must also be satisfied in order to filter out the wrong candidates. They are all important, but we note that `Ident[F]`, for all features F (Equation 17.18),

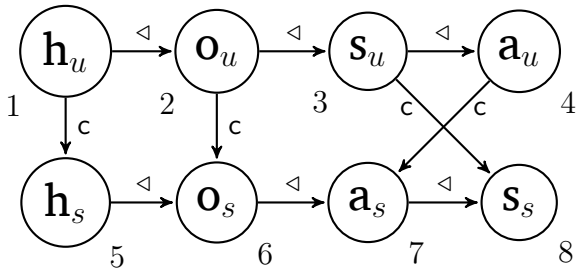


Figure 17.9: The candidate for the UR /hosa/ that satisfies the constraint in 17.26 by undergoing metathesis.

ensures that the elements corresponding to each other in the UR and SR remain the same. In addition, `Uniformity` (Equation 17.13) and `Integrity` (Equation 17.14) make sure that each element in the UR corresponds to at most one other element in the SR, and vice versa.

17.4.5 Fission

Fission is a phonological process when one segment in the UR surfaces as multiple segments in the SR. An example of fission can be found in Tlachichilco Tepehua (Watters, 1988). In this language, underlying /p/ in the coda position surfaces as [wk]. For example, /ʃap.ɬi/ 'X panted (perf.)' surfaces as [ʃawk.ɬi].

To simplify our description of this process, we will assume that the positions in the UR are also identified with syllabic roles such as onset, nucleus, and coda. For example, $\texttt{coda}\,(x)$ would mean that position x is a coda. Readers are referred to Chapter 15 and Strother-Garcia (2019) for more details on model theoretic approaches to syllable structure and syllabification.

First, we define the condition that triggers fission in 17.27. Simply put, a $p]_{coda}(x)$ is true if position x is labeled p and is in a coda position.

$$\texttt{p}]_{\texttt{coda}}\,(x) \stackrel{\text{def}}{=} \texttt{p}\,(x) \wedge \texttt{coda}\,(x) \tag{17.27}$$

We then state the constraint to enforce fission 17.28 as follows. If there is a /p/ in the coda position in the UR, then it corresponds to two segments

in the SR. These two segments are [w] and [k], and [w] precedes [k].

$$\texttt{p\_fission} \stackrel{\text{def}}{=} (\forall x)\big[[\ \texttt{p}]_{\texttt{coda}}\ (x) \wedge \texttt{u}(x)] \leftrightarrow (\exists y, z)[\texttt{c}(x, y) \wedge \texttt{c}(x, z) \wedge y \lhd z \wedge \texttt{w}(y) \wedge \texttt{k}(z)\big] \tag{17.28}$$

Figure 17.10 shows the correct candidate. We indicated whether a segment is an onset, nucleus, or coda in the superscript. As with the previous

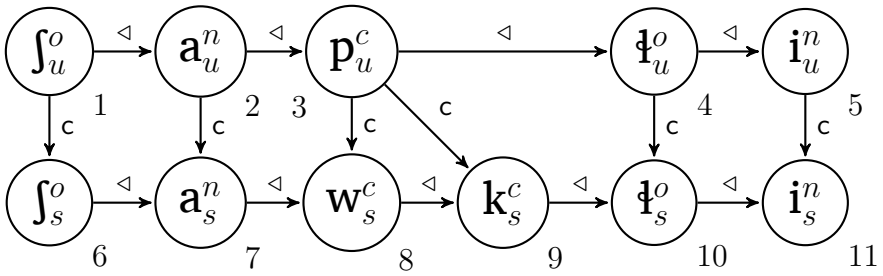


Figure 17.10: The candidate for UR /ʃapɬi/ that satisfies the constraint in 17.28, by fissioning coda /p/ in the UR into [wk] in the SR.

examples, the faithfulness constraints stated in `Ident[F]`, `Integrity`, `Linearity`, `Max`, and `Dep` also must be satisfied to rule out incorrect candidates.

The constraint `Uniformity` will not be satisfied in general. However, it must be satisfied everywhere except when there is a coda /p/. In that case, `p_fission` must be satisfied instead. Abstractly, this situation can be expressed in propositional logic as follows: $P \rightarrow Q \wedge negP \rightarrow R$. In other words, if P is true then Q must be true and if P is not true then R must be true. This idea can be expressed in first-order logic as shown in Equation 17.29.

$$(\forall x)[\big(\ \texttt{p}]_{\texttt{coda}}\ (x) \rightarrow\ \texttt{p\_fission}\ \big) \wedge \big(\neg\ \texttt{p}]_{\texttt{coda}}\ (x) \rightarrow\ \texttt{Uniformity(x)}\ \big) \tag{17.29}$$

The predicate `Uniformity(x)` is the same as `Uniformity` except that the x argument has already been saturated. For concreteness, it is shown below.

$$\texttt{Uniformity(x)} \stackrel{\text{def}}{=} (\forall y, z)\big[(\texttt{u}(x) \wedge \texttt{c}(x, y) \wedge \texttt{c}(x, z)) \rightarrow y = z\big] \tag{17.30}$$

17.4.6 Coalescence

Coalescence is the process where two segments in the UR surface as one segment in the SR. One example of coalescence is found in Indonesian (Lapoliwa, 1981). When the prefix /məŋ-/ is attached to a root that begins with a voiceless stop, the nasal at the end of the prefix coalesces with the voiceless stop. The resulting segment on the surface is a nasal with the place of articulation of the stop (6).

(6) a. /məŋpak/ → [məmak] 'to pack'
b. /məŋtik/ → [mənik] 'to type'
c. /məŋkasih/ → [məŋasih] 'to give'

We first define the conditioning environment for coalescence as a nasal followed by a voiceless stop, shown in Equation 17.31.

$$\texttt{NT}\,(x) \stackrel{\text{def}}{=} (\exists y)[x \lhd y \wedge \texttt{nasal}(x) \wedge \texttt{voiceless}(y) \wedge \texttt{obstruent}(y)] \quad (17.31)$$

The constraints which enforce coalescence are shown in Equations 17.32 through 17.34. If there is a nasal followed by a voiceless obstruent in the UR, then both correspond to one segment in the SR, and this segment is a nasal and matches the place of articulation of the stop that it corresponds to. Conversely, if there is a segment in the SR that has two correspondents in the UR, it must be because the condition stated in 17.31 holds in the UR. The three statements in 17.32-17.34 only differ regarding the place of articulation of the stop in the UR.[1]

$$\texttt{N\_lab\_coalescence} \stackrel{\text{def}}{=} (\forall x, y)\Big[\big(\,\texttt{NT}\,(x) \wedge x \lhd y \wedge \texttt{u}(x) \wedge \texttt{labial}(y)\big) \leftrightarrow (\exists z)[\texttt{c}(x,z) \wedge \texttt{c}(y,z) \wedge \texttt{nasal}(z) \wedge \texttt{labial}(z)]\Big] \quad (17.32)$$

$$\texttt{N\_alv\_coalescence} \stackrel{\text{def}}{=} (\forall x, y)\big[\big(\,\texttt{NT}\,(x) \wedge x \lhd y \wedge \texttt{u}(x) \wedge \texttt{alveolar}(y)\big) \leftrightarrow (\exists z)[\texttt{c}(x,z) \wedge \texttt{c}(y,z) \wedge \texttt{nasal}(z) \wedge \texttt{alveolar}(z)]\big] \quad (17.33)$$

[1] It may be possible to combine and simplify these statements with a model-theoretic approach to features that includes functions, and not only relations, in the signature. In that case, one could write $\texttt{place}(x) = \texttt{place}(y)$ to indicate that the place values of positions x and y must be the same. See Chapter 22 and Chandlee and Jardine (2021) for examples of model-theoretic representations with functions in their signature.

$$\texttt{N\_dor\_coalescence} \stackrel{\text{def}}{=} (\forall x, y)\big[\big(\ \texttt{NT}\ (x) \wedge x \lhd y \wedge \texttt{u}(x) \wedge \texttt{dorsal}(y)\big) \leftrightarrow (\exists z)[\texttt{c}(x,z) \wedge \texttt{c}(y,z) \wedge \texttt{nasal}(z) \wedge \texttt{dorsal}(z)]\big] \tag{17.34}$$

Figure 17.11 illustrates the correct candidate for the UR /məŋpak/.

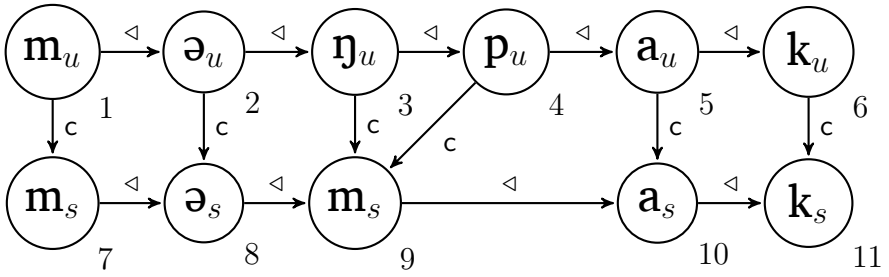


Figure 17.11: The candidate for the UR /məŋpak/ which coalesces /ŋ/ and /p/ to a [m] in the SR. It satisfies the constraints in 17.32-17.34.

The correct candidate must also satisfy the faithfulness constraints stated in `Ident`, `Uniformity`, `Linearity`, `Max`, and `Dep`. The constraint `Integrity` must be satisfied for all positions x except when `NT` $(x) \wedge \texttt{u}(x)$ is true. In that case, the constraints `N_lab_coalescence`, `N_alv_coalescence`, and `N_dor_coalescence` must be satisfied instead. The reader is invited to write out the logical formulas expressing this, following the fission example above.

17.5 Conclusion

This chapter has provided an analysis of the role of GEN, EVAL, and Correspondence Theory in OT and phonological theory more generally. The core of the analysis depends on correspondence structures, which find new expression as relational structures in model-theoretic terms.

Like autosegmental representations (Chapter 12) and syllable structure (Chapter 15), correspondence structures are multilinear structures. That

is, they contain multiple levels, each level is a sequence, and there are relations between each level. While the similarities in the first-order logical expressions defining these different structures should not be overlooked, it is not the case that they are all the same. For example, tonal association and syllable dominance do not violate the no-crossing constraint. But correspondence structures may violate it as seen in the metathesis example above.

With correspondence structures in hand, it was shown that the GEN function is *not* MSO-definable. This result means that researchers in OT should study the MSO-definability of GEN as a relation, GEN as a function which maps underlying strings to contenders, or GEN as it is conceived in Harmonic Serialism.

Two other important facts were shown. First, the infinite set of candidates for a given UR is MSO-definable as a language, and in fact is FO-definable. Second, we showed that common processes in phonology can be understood as satisfying language-specific, FO logical constraints over correspondence structures. These analyses do not rely on optimization. They are declarative in the sense that every constraint is inviolable. For example, in a language where the SRs are always fully faithful to the URs, the standard faithfulness constraints in Correspondence Theory would all have to be satisfied. In a language that is not always fully faithful to URs, one or more than one of them, will be replaced by constraints that carve out exceptions to them.

In the OT literature, such constraints bear a strong resemblance to two-level constraints (Kager, 1999, pp.378-381). These were rejected in OT for several theoretical reasons: they are too stipulative, they inappropriately mingle marked structures and repairs, and they are insufficiently phonetically grounded. Nonetheless, it is the case, at least for the simple, common examples studied here, that the right candidate for a given UR can be determined through the satisfaction of statements of FO logic over correspondence structures. Some of these FO statements can be thought of as generating possible candidates (Equations 17.4-17.9) and the faithfulness constraints can be thought of as filtering out the wrong ones (Equations 17.13-17.18). If this result extends to more complex phonologies, then the fact of FO-satisfiability of candidate selection for a given UR is not something that cannot be accounted for by Optimality Theory. This is because global optimization is known to be more powerful than first-order logic (Frank and Satta, 1998; Riggle, 2004; Gerdemann and

Hulden, 2012; Heinz and Lai, 2013; Chandlee *et al.*, 2018; Lamont, 2021, 2022b; Hao, 2024). While the constraints are language-specific, this issue is less important if the language-specific constraints can be learned, and there is promising work in this direction (Strother-Garcia *et al.*, 2017; Vu *et al.*, 2018; Chandlee *et al.*, 2019; Lambert *et al.*, 2021; Rawski, 2021; Payne, 2024). Furthermore, the complete independence of repairs from marked structures is not without issues (Blumenfeld, 2006). As for phonetic groundedness, some cases may be captured with the right representations (see Chapters 18 and 19). In sum, we conclude it is premature to reject constraints over correspondence structures of the sort that we have investigated here.

Our analysis is not comprehensive in the sense that there are many unanswered questions worth pursuing. First and foremost, the examples we considered were simple, and it would be valuable to investigate more complex processes. Future work could also examine long-distance phonological processes such as dissimilation, vowel harmony, and consonantal harmony. Finally, it is also of interest to know the extent to which phonological maps can be expressed with more restrictive logics (such as the conjunction of forbidden substructures) using the correspondence structures discussed here.

Overall this work indicates that Correspondence Theory may have a substantial role to play in phonological theory beyond the framework of Optimality Theory.

Chapter 18

Phonetically Grounded Phonological Representations

CURT SEBASTIAN

18.1 Introduction

The aim of this chapter is to create a means by which phonetic information – mainly in the form of implicational hierarchies – may be incorporated into the phonological component as a set of surface restrictions characterized by logical formulas over phonological representations. The investigation is motivated by the work presented in several chapters of *Phonetically Based Phonology* (Hayes *et al.*, 2004), which proceeds from the view that "phonological constraints can be rooted in *phonetic knowledge* (Kingston and Diehl 1994), the speakers' partial understanding of the conditions under which speech is produced and perceived" (emphasis in original, p. 1). The book argues that Optimality Theory is ideally suited to incorporate the phonetically based markedness. In Hayes *et al.* (2004), each contributor provides an in-depth analysis of one or more phonetic properties, and the way it putitatively manifests in cross-linguistically. The products of these investigations often describes how the specific phonetic parameters can be interpreted as a set of markedness constraints that, when combined with the appropriate faithfulness constraints give rise to factorial typologies that are argued to match known cross-linguistic variation.

This chapter argues that the same insights that phonetic-based marked-

ness bring to Optimality Theory can also be expressed as language-specific, inviolable constraints over phonetically-grounded representations, and that such grammars also account for the observed typological variation. It follows that it may not be necessary for practitioners of phonetically-based phonology to conduct their analyses within the framework of Optimality Theory or its variants. Instead they may consider an alternative analysis, along the lines of what is proposed here.

The analysis in this chapter focuses on the implicational hierarchy created by Hayes and Steriade (2004) to represent voicing difficulty in obstruents. Voicing difficulty is a characterization of the degree of physical difficulty in reconciling the various articulatory gestures and strategies employed to produce the 'active oral tract expansion…necessary to maintain airflow in an obstruent' (Hayes and Steriade, 2004, p. 8). The phonetic explanation is very involved, but for the purposes of this chapter, it will suffice to say that while 'voicing difficulty' is the result of the interaction of numerous phonetic variables. Hayes and Steriade reduce the process to the intersection of two parameters: duration of the oral closure, and the size of the oral cavity during closure (as a result of place of articulation). From these parameters, they generate the following implicational hierarchy to represent the phonetic difficulty of maintaining voicing in obstruents shown below.

(7) Hierarchy of phonetic difficulty for obstruent voicing from least difficult [b] to most difficult [g:].

$$b < d < g < b: < d: < g:$$

From this hierarchy, Hayes and Steriade derive a fixed ranking of markedness constraints shown below.

(8) Fixed ranking of markedness constraints derived from (7).

$$\textbf{*g:} \gg \textbf{*d:} \gg \textbf{*b:} \gg \textbf{*g} \gg \textbf{*d} \gg \textbf{*b}$$

When the faithfulness constraint IDENT(VOICE) is added to this constraint set, the resulting factorial typology exhibits an implicational hierarchy. The existence of any one of the six obstruents under discussion implies the existence of the obstruents which are less difficult (so members to its left in (7)). For example, if a language contains [b:] then the singletons [g d b] will also be present in the language, but not necessarily the geminates

[d: g:]. This is because if [b:] surfaces in the language, IDENT(VOICE) must be ranked above the markedness constraint ***b:**. Since the ranking in (8) is fixed for all languages, IDENT(VOICE) will also be ranked above the constraints ***g:**, ***d:**, and ***b:**.

More generally, the ranking of IDENT(VOICE) with respect to the fixed ranking in (8) divides the scale of voiced obstruents into two groups. Letting x be a variable over the voiced obstruents, it follows that if ***x** is ranked below IDENT(VOICE) then [x] will surface in the language, but if ***x** is ranked above IDENT(VOICE) then [x] will not surface in the language. Consequently, their analysis predicts that there will be languages whose set of voiced obstruents are exactly one of: [] (no voiced obstruents), [b], [b d], [b d g], [b d g b:], [b d g b: d:], and [b d g b: d: g:], depending on precisely where IDENT(VOICE) is ranked with respect to the markedness constraints. Hayes and Steriade report languages which exhibit each of these sets of voiced obstruents.

In this chapter, I show that the same typological variation follows from the same phonetic insights using a grammar of inviolable constraints. In particular, the grammar is comprised of a set of forbidden factors, where the factors are model-theoretic structures which encode representations of phonetic difficulty directly. In the parlance of Chapter 5, this means that the grammars are a restricted form of propositional logic; specifically, they are conjunctions of negative literals. In fact, in order to account for the typological variation above, it is sufficient that the grammars contain only a single factor.

18.2 Incorporating Phonetic Fact into Logical Formalism

Hayes and Steriade (2004, p. 7) provide a schema, that is a series of steps, that lead from phonetic facts to cross-linguistic variation in sound patterns, which is reproduced below.

(9) a. Facts of phonetic difficulty
b. Speaker's implicit knowledge of the facts in (a)
c. Grammatical constraints induced from the knowledge in (b)
d. Sound patterns reflecting the activity of the constraints in (c)

For them, the term 'constraints' here ultimately means OT constraints, which are ranked and violable. However, I maintain that OT constraints are not an essential component for the steps outlined above. The paradigm created by the four statements above does not change under the general interpretation of constraint as some kind of grammatical restriction. In other words, the schema they envision is available to any theoretical approach provided it is clear how the grammars in (c) are derived from (b), and how a theory of such grammars generates a typology (d).

There is a challenge inherent in incorporating the information provided by implicational hierarchies into phonology using the logic and model theory. Hierarchies like the voicing scale do not make statements about individual segments; they use scales of individual segments to make statements about cross-linguistic generalizations, and, in this way, about the alphabets of individual languages. The method by which an individual segment in a string may be assessed by a phonetically-based markedness hierarchy using model theory and logic is not intuitively obvious. For instance, the voicing difficulty scale above relates the claim that if the geminate [b:] occurs in a language then singletons [b d g] should also occur. How would a logical formalism capture and implement such a hierarchy in the form of constraints on representations, using only abstract implicational data? And how would a logical formalism capture and implement a constraint on the instantiation of a particular voiced obstruent itself, without accessing the alphabet of language in question.

This state of affairs gives rise to two difficulties. The first takes the form a question: How can an abstract representation of phonetic facts like the scale in (7) be characterized with constraints on an individual segments? The second leads to this conclusion: in some sense, implicational hierarchies constitute 'meta-constraints' on the phonology of languages. They are broad statements about the structure of phonologies across languages. This raises the question of whether the implementation of these meta-constraints can be characterized by logical formalisms and representations? The initial question is addressed in §18.3. The follow-up is addressed in the discussion.

18.3 Representing Phonetic Difficulty

This section demonstrates one way scales of phonetic difficulty (the difficulty of voicing obstruents in this case) can be incorporated into phonologi-

cal representations using model theory. The critical insight of this chapter is that implicational hierarchies derived from phonetically-motivated markedness scales can be represented with zero or more 'particles' which can be thought of as discrete tokens measuring some quantity of phonetic information. The number of discrete tokens increases as the relevant phonetic measurement increases.

For the scale for phonetic difficulty of voicing (7), the number of tokens increases with difficulty. For instance, the segment [b] would carry one token, [d] would carry two tokens, [g] would carry three tokens and so on. The number of tokens should increase incrementally as the level of voicing difficulty encompassing the entire scale. Model theoretically, these phonetic particles will be elements of the domain which are associated to the appropriate segmental position, which itself is an another element of the domain.

Specifically, in addition to the unary relations such as `sonorant`, `voice`, `long`, `coronal`, `labial`, and `dorsal`, there is an additional unary relation `difficulty`, which is only satisfied by domain elements indicating these phonetic particles. A binary relation α associates these tokens to domain elements satisfying $\neg$`sonorant`$\wedge$`voice`. The model signature in Equation 18.1 where F is a set of features consonants and vowels, including the ones previously mentioned.

$$\mathfrak{R} = \{\texttt{difficulty}, \lhd, \alpha\} \cup F \tag{18.1}$$

Figure 18.1 illustrates $\mathfrak{R}$-structures representing the voiced obstruents [b d g b: d: g:]. In the figure, I abbreviate `difficulty` with `diff`.

Figure 18.2 presents a graphical representation of the word [b:od] with these representations. The proposed representation is not unlike an autosegmental tier. There is a central, skeletal tier where the positions of the segments making up the words are ordered. A distinct level of representation is given to the particles indicating phonetic difficulty. However, unlike a tonal tier (see Chapter 12) or a syllabic tier (see Chapter 15), the elements on this tier need not be ordered, only associated to positions on the skeletal tier.

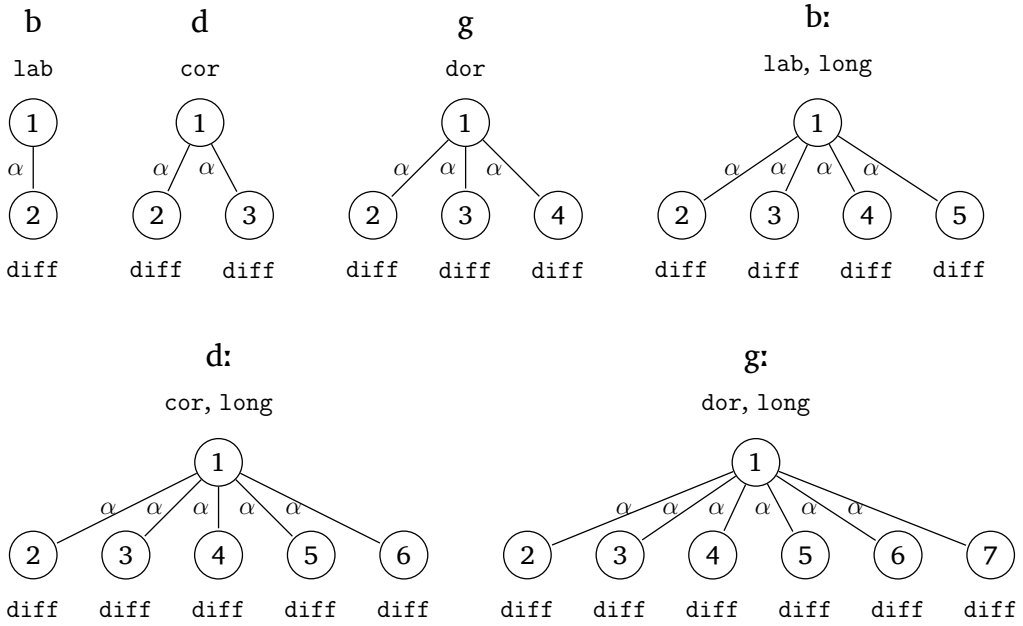


Figure 18.1: Visualization of the $\mathfrak{R}$-structures of the segments [b, d, g, b:, d:, g:] with phonetic difficulty as part of the representation. Each domain element 1 also satsfies other features, e.g. `voiced` and `stop` (not shown).

18.4 Constraints on Phonetic Difficulty

With these representations in place, one can use propositional logic over factors of $\mathfrak{R}$-structures as described in Chapter 5. In particular we are interested in factors that identify segments carrying tokens of phonetic difficulty. Figure 18.3 defines 6 factors that identify positions with different degrees of phonetic difficulty. An important observation is that these factors *are nested*. In other words, 1D is contained within 2D, which itself is contained within 3D and so on. Similarly, the factor 4D contains the factors 3D, 2D, and 1D.

How are these factors interpreted in within a propositional logic? Recall from Chapter 5 that the extension of a factor f is all the $\mathfrak{R}$-structures that contain the factor f. It follows that the propositional sentence 4D would be interpreted as all words containing a position which has four particle of phonetic difficulty. When it comes to numbers, the English language

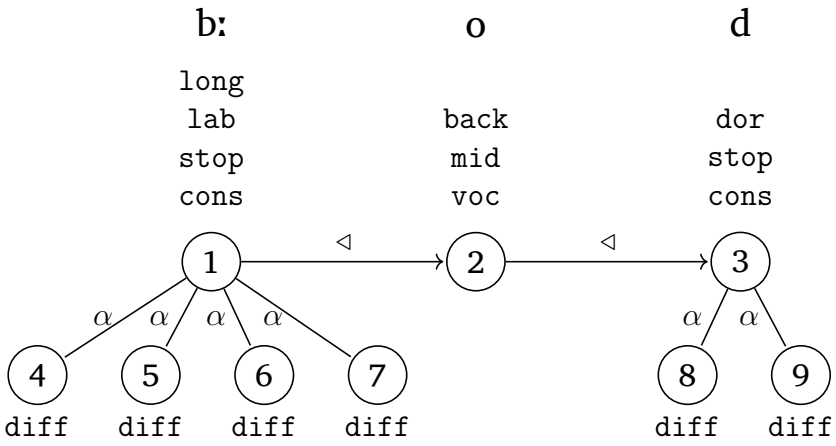


Figure 18.2: The $\mathfrak{R}$-structure of the hypothetical word [b:od].

can be ambiguous so let's be clear that this does not mean "exactly four particles of phonetic difficulty," but "at least fours particle of phonetic difficulty." This is because the factors are contained within in each other as mentioned above. For example, the $\mathfrak{R}$-structure of the word [b:od] in Figure 18.2 satisfies the sentence 4D but the $\mathfrak{R}$-structure of the word [bod] would not.

Conversely, the propositional sentence ¬4D would be interpreted as all words which do *not* contain a position which has four particles of phonetic difficulty. Again, because the factors are contained within in each other, this does not mean "exactly four particles of phonetic difficulty," but "at least four particles of phonetic difficulty." Thus not only would the sentence ¬4D not be satisfied by $\mathfrak{R}$-structure of the word [b:od], it is also not satisfied by by $\mathfrak{R}$-structures for the word [d:od] and [g:od] because those structures contain the factor 4D as well.

Recall from Hayes and Steriade (2004), that there were seven kinds of languages in the typology of languages with voiced obstruents: [] (no voiced obstruents), [b], [b d], [b d g], [b d g b:], [b d g b: d:], and [b d g b: d: g:]. With factors like the ones shown in Figure 18.3, these languages are generated with the sentences of propositional logic shown in Table 18.1.

Under the current analysis, the constraints effectively impose a threshold of phonetic difficulty for individual languages. The implicational hierarchy that was obtained in Optimality Theory via a fixed ranking of constraints is accomplished here representationally via containment. In

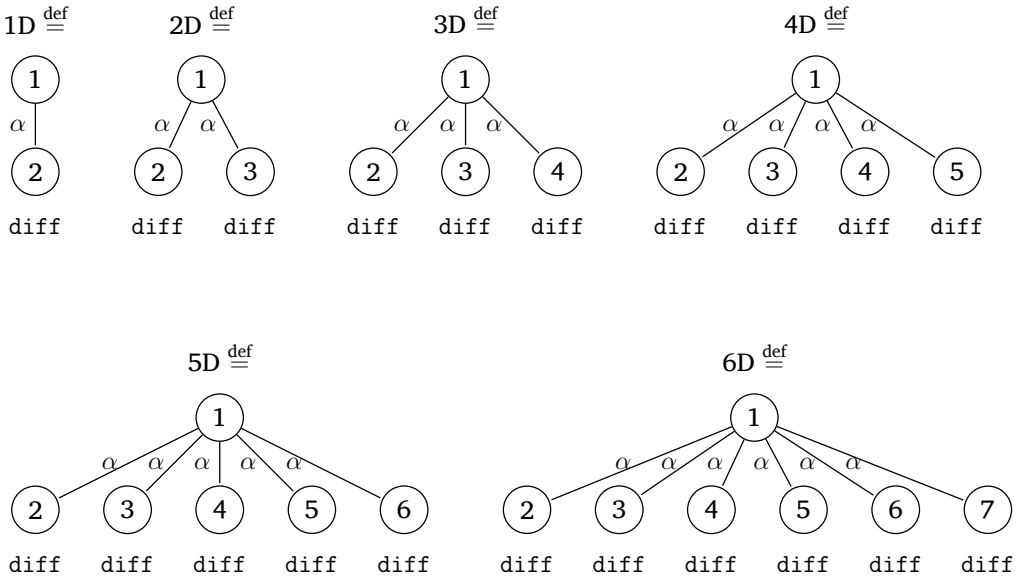


Figure 18.3: Factors 1D, 2D, 3D, 4D, 5D, 6D.

this way, this analysis provides a straight line from the phonetic insight that voiced obstruents are ordered by degree of difficulty through the grammars proposed here to the putative typology which reflects that those degrees of difficulty provide cutoffs for the inventory of voiced obstruents along that scale (see the schema in (9)). It is also striking that this can be achieved with grammars that are the conjunctions of negative literals; indeed, with grammars containing a single negative literal.

18.5 Discussion

There are several aspects of the analysis that warrant further discussion.

First, one may wonder why the sentence `true` in Table 18.1 is given instead of the sentence ¬7D. There is no particular reason other than that the factor ¬7D has not been formally defined. Of course one could define it and then the sentence `true` and ¬7D would be extensionally equivalent as far as voiced obstruents are concerned.

A related question concerns the extension of sentences like ¬2D∧¬4D. Since 4D contains 2D, this expression is redundant and its extension is

Sentence	Voiced obstruents
¬1D	[]
¬2D	[b]
¬3D	[b d]
¬4D	[b d g]
¬5D	[b d g b:]
¬6D	[b d g b: d:]
`true`	[b d g b: d: g:]

Table 18.1: Grammars yielding the typology voiced obstruents.

equivalent to ¬2D. It follows that in the same way there are many rankings of constraints in OT that yield the same language, there are many sentences of propositional logic over $\mathfrak{R}$-structures which are extensionally equivalent. In general, a sentence $\neg n\text{D} \wedge \neg m\text{D}$ will be extensionally equivalent $\neg n\text{D}$ if and only if $n < m$.

A third question has to do with ensuring that the proposed formalism does not yield unwanted inventories such as [g b: d: g:]. I address this question in two parts. First, if the only constraints under consideration are the ones shown above, then it should be clear that obtaining an inventory like [g b: d: g:] is not possible because the factor that effectively forbids [b] will also forbid [d g b: d: g:].

The second part regards other constraints that may be in the system. For example, nothing in principle prevents there from being factors which pick out [b] and [d] as shown in Figure 18.4. The sentence $\neg b \wedge \neg d$ would yield

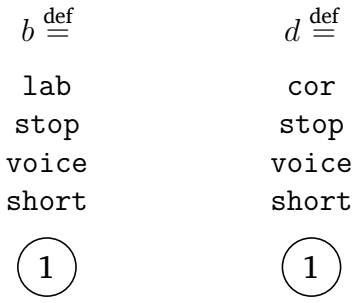


Figure 18.4: Factors b and d.

the unwanted inventory [d g b: d: g:] because no constraints on phonetic difficulty are expressed, only constraints on particular combinations of features.

However, the point of phonetically-grounded phonology is that constraints are *not* formed with arbitrary combinations of features. Therefore, a theory of phonetically-grounded phonology which utilizes the kind of representations and logical formalism here must rule out sentences like $\neg b \wedge \neg d$ on the grounds that they contain literals which have no phonetic grounding. There could be many ways to operationalize this. One way would be to indicate which relations in the model-theoretic signature are phonetically relevant, and to only allow sentences with factors containing those relations. Another way would be to implicate phonetic difficulty with position identity. This is the case for Hayes and Steriade (2004) because the OT constraints against these segments (***b** and ***d**) are placed in a fixed hierarchy and not allowed to be freely ranked. Presumably, they would not allow a constraint such as *[$-$son,$-$long,$-$dorsal,$+$voice] to be freely ranked, which would similarly disrupt their typology.

I leave for future research the question of how best to operationalize phonetic-groundedness using the logical and representational methods employed here. I have made the central point that the vehicle for the phonetic-grounding of constraints offered here accomplishes the same goals as the analysis in Hayes and Steriade (2004). A broader point is that while the representations invoked via $\mathfrak{R}$-structures and the logical language used to express constraints help us understand the nature of phonological constraints, they are not necessarily an exhaustive theory of it.

18.6 Conclusion

Discrete phonetic tokens, in the form of 'phonetic difficulty particles' are a viable means of phonetically-grounding constraints via phonological representations and propositional logic. These representations faithfully represent the values of the voicing difficulty scale and constraints over such representations express the threshold of difficulty the speaker of a language is willing to tolerate. In this way, the phonetic facts are functionally realized in phonological representation.

This analysis suggests lines of further inquiry to me. The first is the question posed in the introduction about the feasibility of characterizing

the 'meta-constraints' on languages generated by implicational hierarchies using logical formalisms. This chapter shows a promising means of incorporating phonetic information directly into the phonology using model-theoretic representations, which could indicate that these 'meta-constraints' could be applied to the alphabets of languages governed by other implicational hierarchies in a similar way.

The second is a more specific and modest objective, and more immediately relevant to the results of this chapter. Zhang (2004) suggests that a combination of sonority and length has an implicational effect on the tone licensing of syllable rhymes. Essentially, longer, more sonorous rhymes are likely to be less restricted with respect to the types of tone they can bear. In Thai, for instance, **CV:O** syllables are more restricted with respect to tone-bearing than **CVR** syllables, even though the rhyme of the former is phonemically represented as being longer than the rhyme of the latter, a situation which generally creates a less restricted tone-bearing environment. Zhang (2004) suggests this apparent asymmetry is due to the fact that in closed syllables (those with obstruent codas), long vowels are phonetically shortened, which renders the rhyme of the **CVR** syllable a more ideal tone-bearing environment, even though long vowels are generally preferred over short. Essentially, tonal variety is licensed by rhyme length, where more length implies more tone-bearing freedom. This seems, in phonetic terms, very analogous to the implicational hierarchy here where the degree of difficulty scaled with the size and character of the inventory of voiced obstruents. It seems very likely that rhyme length could be represented with discrete phonetic tokens, which implies in turn that the constraint and any processes implied by the phonetic facts could be expressed with logical formalisms over such representations.

Chapter 19

Representations of Gradual Oppositions

HYUN JIN HWANGBO

This chapter provides a computational analysis of a vowel lowering process in Danish, in which certain vowels lower adjacent to /r/ within a syllable. This process is also known as r-coloring (Basbøll, 2005). If the rungs of a ladder represent vowel height, then, for these vowels, Danish vowel lowering is a single step to the rung below. This process, while simple and intuitive to describe, has been problematic for both rule-based and constraint-based accounts. The problem, as noted by Schane (1984) and Basbøll (2005), among others, is that representations of vowels as complexes of binary features leads to overly complex descriptions of this kind of process. As I will discuss, a rule-based account requires multiple rules in an opaque ordering, and a constraint-based approach requires multiple, special constraints to handle this opacity.

Schane's approach is to explicitly represent vowel height in terms of degrees of aperture. Basbøll (2005, Chapter 5) provides a similar analysis in terms of degrees of distance from the maximally low and back vowel. These scales formalize the notion of a **gradual opposition** for vowel height in Trubetzkoy's (1969) terminology, which I explain in more detail below. Once these scales have been set, it is possible to formalize the generalization directly: vowels subject to lowering increase in aperture (or decrease their distance to the maximally low and back vowel) by one degree. In this way, these representational choices allow one to provide an analysis which

directly captures the nature of the phonological change.

This chapter provides a First Order (FO) logical transduction with relational structures that represent degrees of aperture directly. With these representations, it is straightforward to express the generalization that Danish vowel lowering is a process that increases aperture by one degree for vowels in the proper context. In this way, this analysis *mirrors* the generalization and satisfies the Mirror Principle for phonological analysis, which expresses a criterion regarding how an analysis within a formalization ought to capture a generalization (Schane, 1984). While I use the aperture-based scale, as opposed to Basbøll's distance-based one, for concreteness, a logical transduction faithful to the representations and generalizations made by Basbøll (2005) is also certainly possible.

On the other hand, it is less clear how rule-based and constraint-based accounts operate over vowels with such representations. As I discuss below, while nothing in principle prevents these formalisms from using these representations and defining mechanisms for operating over them, the formalisms themselves need to be expanded in order to adjust to these representations. In contrast, changing the representations presents no obstacle to the logical methods employed here.

The remainder of this chapter is organized as follows. In §19.1, I review and discuss background material, including salient aspects of Danish vowel lowering, Trubetzkoy's views of phonemic oppositions, and the Mirror Principle. Then in §19.2, I review shortcomings of rule-based and constraint-based accounts of Danish vowel lowering when vowels are represented with binary features, and explain how such analyses violate the Mirror Principle. The next section §19.3 explains how degrees of aperture can be represented directly with relational structures and §19.4 defines a logical transduction which increases the aperture by one for vowels in the appropriate /r/ context. §19.5 discusses some related aspects of the analysis, and §19.6 concludes.

19.1 Background

19.1.1 Danish Vowel Lowering

Basbøll (2005, p. 149) describes Danish vowel lowering as follows.

> The term r-coloring denotes a whole series of sound changes in

> Danish, which are mirrored in synchronic phonological rules, all with the effect that the vowel in question, when it is adjacent to an /r/, becomes 'one step closer' to the low pharyngeal vowel (see Basbøll [1972]).

In this chapter I review some fundamental aspects of this process, but I limit the scope of the analysis here in two ways. First, I choose to focus exclusively on short, front vowels, even though some long and back vowels also undergo this process. Second, I choose to simplify the environment which causes lowering to only post-vocalic /r/. These limitations are sufficient to establish the main points of the analysis and it is left to future research to further develop the analysis without these limitations.

Table 19.1 displays the set of full, short vocalic phonemes in Danish (Basbøll, 2005, p. 50) using descriptions from the IPA.

	Front		Back	
Aperture	Unrounded	Rounded	Unrounded	Rounded
close	i	y		u
close-mid	e	ø		o
open-mid	ɛ	œ		ɔ
open	a		ɑ	ɒ

Table 19.1: Full, short, vocalic phonemes of Danish.

As mentioned, I set aside the back vowels in the analysis and focus on the front vowels. Basbøll (2005, p. 151) presents the input-output relations between full, short vowels affected by r-coloring, which is reproduced in Table 19.2 for the front vowels. The asterisk on the output [ɑ] indicates that

Input	a	ɛ	e	œ	ø
Output	ɑ*	a	ɛ	œ	œ

Table 19.2: Front, full, short vowels which undergo vowel lowering.

this vowel also fuses with /r/ and the /r/ itself is elided. The analysis below does not address this fusion with /r/. Observe that the front vowels /i, y/ are not included in Table 19.2 because they show no lowering adjacent to /r/.

Basbøll (2005, p. 48) presents visualizations of the full vowels of Modern Standard Copenhagen speech (including allophones), where unround and round vowels are separated. While the quadrilateral shapes on top are standard, Basbøll prefers the lower graphs because they do not give the "impression of a corner" (p. 48) near the [a] vowel. An important

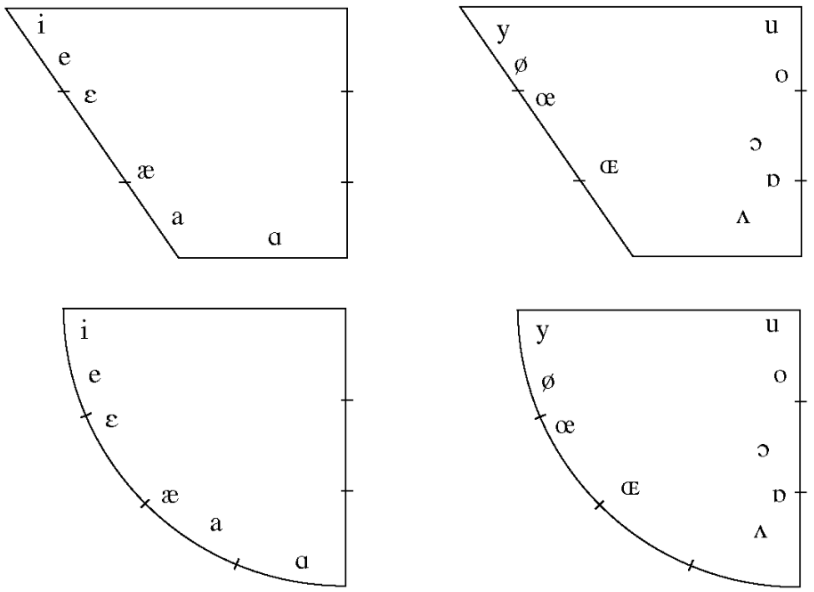


FIGURE 2.2. Full vowels of Modern Standard Copenhagen Speech (from Basbøll and Wagner 1985: 40).

Figure 19.1: Figure 2.2 from Basbøll (2005, p. 48).

consequence of the curved, triangular shape is that as aperture increases, backness also increases, though nonlinearly.

As for the precise nature of the environment that triggers the change in vowel quality, it appears that the /r/ must be adjacent and tautosyllabic with the vowel, but there is variation. While there is evidence that both tautosyllabic pre-vocalic and post-vocalic /r/ inducing lowering, there is also evidence they do so differently. For instance, Basbøll (2005, p. 150, fn. 6) notes that aspects of the syntagmatic patterning of /r/ can be variable "with respect to speaker variables, style level, and lexicon." In this chapter, we are primarily interested in the nature of the change itself, and not the

environment. I agree with Basbøll (2005, p. 150), who writes "The unity of the process, which is in my view a real one and not spurious, lies in the change of the segments in question." Nonetheless, for concreteness, I adopt the first approximation for analysis that front vowels lower when they are immediately followed by /r/ (Hyman, 1975, pp. 68–69). Putting it altogether, the phonological generalization that will be analyzed in the remainder of this chapter is shown below.

Generalization 19.1. Danish front vowel lowering:
Front, non-high, short vowels lower when they are immediately followed by /r/.

19.1.2 Trubetzkoy's Oppositions

Trubetzkoy (1969) identified three kinds of phonemic oppositions: privative, gradual, and equipollent. Privative oppositions are those indicated by the presence versus absence of a property such as voicing or aspiration. Gradual oppositions refer to a category that are described with "various degrees or gradations of the same property (Trubetzkoy, 1969, p.75)." That is, differences among the phonemes are shown along a scale. Vowel height is a good example as it can be described with degrees of aperture as illustrated in Figure 19.1. Equipollent oppositions are neither privative nor equipollent. Place of articulation (labial, interdental, coronal, dorsal) is an example of an equipollent opposition.

While binary features naturally represent privative oppositions, it is less clear whether they are appropriate for gradual and equipollent oppositions. For example, consider vowel height. With a binary feature system, vowel height can be represented with features such as [+high] or [−high] and [+low] or [−low] in the same way as frontness or backness with features such as [+front] or [−front] and [+back] or [−back]. However, if a language has four vowel heights /i, e, ɛ, a/, for example, then both /e/ and /ɛ/ are described as [−high, −low] with this binary feature system. To properly distinguish them, a third feature is needed such as [tense] as shown in Table 19.3. With this feature, /e/ can be described as [−high, −low, +tense] and /ɛ/ as [−high, −low, −tense]. In other words, we need three features in the binary feature system to properly represent the vowels.

Now consider what happens if, as in Danish, there is a vowel lowering process. Since /e/ lowers to [ɛ], the formal representation of the process

Vowel	high	low	tense
i	+	−	
e	−	−	+
ɛ	−	−	−
a	−	+	

Table 19.3: Vowel features of the four height with binary features system

must change the [tense] feature from [+tense] to [−tense]. And since vowel /ɛ/ lowers to [a], the formal representation of the process must change the [low] feature from [−low] to [+low]. Consequently, the question arises: does the representation of vowels with these features in the binary system *mirror* the phonological transformation of vowel lowering?

19.1.3 The Mirror Principle

Schane (1984) introduces the Mirror Principle as a criterion for formalizations. He writes "If one believes that a process or change happens in a certain way, then the notation should not just describe that event but should *reflect* as closely as possible its manner of occurrence" (emphasis added). To illustrate this idea, he presents two rules describing a process where a consonant assimilates to a following vowel. These rules are reproduced below.

(10) a. $\text{C} \rightarrow [+\text{sharp}] / \_\_ \begin{bmatrix} \text{V} \\ +\text{high} \\ -\text{back} \end{bmatrix}$

b. $\text{C} \rightarrow \begin{bmatrix} +\text{high} \\ -\text{back} \end{bmatrix} / \_\_ \begin{bmatrix} \text{V} \\ +\text{high} \\ -\text{back} \end{bmatrix}$

Schane (1984) observes that "Rule (10a) requires independent, unrelated features; (10b) does not." He goes on to explain: "For this particular example, the notation of generative phonology mirrors the nature of the process, and I believe it is fair to say that generative phonology has considered mirroring to be one of the goals of its notation."

19.2 Traditional Analyses

As Schane (1984); Basbøll (2005), and others have observed, the binary feature system in Table 19.3 does not capture, or "mirror" the representation of the vowels and the lowering process, because the features that undergo the transformation do not directly point to the lowering. This is true for both rule-based accounts and for constraint-based accounts.

In a rule-based approach, this lowering process would be given by the rules in (11). Each rule in (11) shows the lowering of each front vowel, and /a/ to [ɑ], /ɛ/ to [a], and /e/ to [ɛ]. Note that the mid vowel /e/ does not change to the low back vowel [ɑ], but to [ɛ], and /ɛ/ lowers to [a] and not to [ɑ]. In other words, the vowels shift only one degree in height. (In the case of /a/ this appears to be a shift from front to back.) Therefore, the rules have to be in the proper order for the correct form to surface. The rules have to be ordered as shown: rule (11a) must apply before rule (11b), and rule (11b) must apply before rule (11c). If the rule was ordered reversed, for example, the mid front vowel /e/ lowers to [ɑ].

(11) a. /a/ $\longrightarrow$ [ɑ] : [−back, +low] $\longrightarrow$ [+back] / __ r

b. /ɛ/ $\longrightarrow$ [a] : [−high, −low, −tense] $\longrightarrow$ [+low] / __ r

c. /e/ $\longrightarrow$ [ɛ] : [−high, −low, +tense] $\longrightarrow$ [−tense] / __ r

This series of rules constitutes an example of counterfeeding opacity (Kiparsky, 1973; Baković, 2007). While the rules in (11) describe the map from underlying to surface form accurately, the rules within the formalism do not capture the process of lowering. That is, each lowering rule changes different features – [back], [low], and [tense] – to obtain a lower vowel in the surface form. In other words, a single lowering process has to be broken down into three different rules, each changing different features. In addition, the rules have to be applied in the order presented above, otherwise the vowel /e/ would lower to [ɑ]. Clearly, the traditional analysis with binary features does not *mirror* vowel lowering in Danish.

Constraint based accounts with binary features face similar difficulties. Consider an account within Harmonic Serialism (HS) (McCarthy, 2010). Hauser *et al.* (2016) proposes the faithfulness constraint, FAITH-UO for opaque processes (see also Hauser and Hughto, 2020). The faithfulness constraint has the form of ID-UO(F)/[αG] which is defined below in (12).

(12) FAITH-UO: ID-UO(F)/[αG] assigns one violation for each value of F that is changed in segments which are [αG] in the underlying representation.

For example, ID-UO(low)/[+tense] assigns a violation when underlying [+tense] vowels are not faithful to [low] features. Thus, the tense vowel [e] is assigned a violation if it changes the low feature. Therefore, the vowel [e] does not lower to [a]. ID-UO(back)/[−low] assigns a violation when an underlying [−low] segment changes its [back] feature. The underlying [−low] segment /ɛ/, is faithful to the [back] feature when lowering to [a]. Lowering from /ɛ/ to [ɑ] would violate this faithfulness constraint because it changes the back feature. In this way, these constraints can account for cases of counterfeeding opacity, such as Danish vowel lowering.

However, the issue here is like the one with the rule-based account: because the constraints implicate different features, they do not capture the fact that Danish vowel lowering is a single process. That is, the constraints requires faithfulness to [low] and [back]. I conclude that, with the binary feature system, either analysis, the rule-based approach or constraint-based, fails to *mirror* the process.

To sum up, counterfeeding chain shifts, where A becomes B, and B becomes C, and C becomes D, and so on, are difficult to account for with binary feature systems. If these chain shifts occur along a scale induced by gradual oppositions, then a representation reflecting that can admit a singular process and eliminate the opacity. The opacity is, in a sense, a byproduct on an inappropriate representation. In the case of Danish vowel lowering, the vowels form a gradual opposition along a scale. Instead of describing the vowel height with binary features as in Table 19.3 above, we want a representation that reflects aperture directly.

19.3 Representing Aperture

This section defines relational structures for words in Danish focusing on the representation of vowels. It shows how the vowel space can be represented according to the scale shown in Figure 19.1.

Table 19.4 divides the vowels (phonemes and one allophone [œ]) by five degrees of aperture. Schane (1984) proposes an aperture particle, a, to denote one degree of aperture. That is, vowels with more aperture

Vowels	Features
i,y,u	0th degree of aperture
e,ø,o	1st degree of aperture
ɛ,œ,ɔ	2nd degree of aperture
a,(ɶ)	3rd degree of aperture
ɑ,ɒ	4th degree of aperture

Table 19.4: Five degrees of aperture. The vowel in parentheses is an allophone.

particles are lower and further back as well. I implement this idea literally with relational structures as follows. There is a unary relation `aperture`, and domain elements satisfying this relation are aperture particles. I additionally adopt a binary relation α which associate aperture particles to domain elements satisfying `vocalic`.

It is interesting to observe that the only two other features needed to fully make distinguish the vowels in Table 19.4 are the features [round] and [back]. These two features will be the unary relations `round` and `back` in the model signature for the relational structures. Specifically, while all vowels described as round in the IPA (see Table 19.1) will satisfy `round`, only the vowels /u,o,ɔ/ will satisfy `back`.

Altogether, the relations I will use to express Danish vowel lowering are shown in the model signature in Equation 19.1 where F is a set of features for non-vocalic Danish segments.

$$\mathfrak{R} = \{\texttt{vocalic}, \texttt{back}, \texttt{round}, \texttt{aperture}, \lhd, \alpha\} \cup F \tag{19.1}$$

Figure 19.2 illustrates $\mathfrak{R}$-structures representing the unround vowels in Danish. Node 1 is labeled with `vocalic` and the nodes from 2 to 5 are labeled with `aperture` (abbreviated as `aper` in the figure). The edges from node 1 to nodes 2, 3, 4, and 5 labeled α indicate the aperture particle is associated to the vowel. The total aperture of the vowel is given by the number of aperture particles associated with it. Describing the round vowels is straightforward. The front round vowels are identical to the ones shown in Figure 19.2 except node 1 in each vowel would also satisfy `round`. The back round vowels are obtained by letting domain element 1 in the vowels with 0, 1 and 2 aperture particles satisfy `round` and `back`.

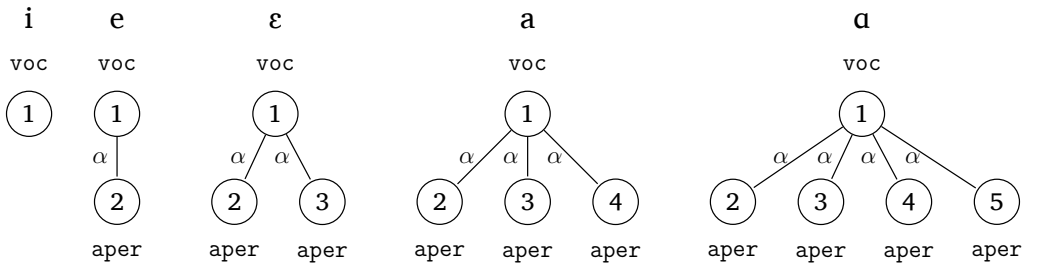


Figure 19.2: Visualization of the $\mathfrak{R}$-structures of unround Danish vowels with aperture particles.

Here is a more word-like example. The $\mathfrak{R}$-structure of a hypothetical word *pok* [pok] is shown Figure 19.3. The domain elements 1, 2, and 3 can be understood as representing the positions of [p], [o], and [k], respectively. Therefore, node 1 is labeled with `labial`, `stop` `cons`, node 2 with `back`, `round`, and `vocalic`, and node 3 with `dorsal`, `stop` and `cons`. The edges from the node 1 to 2 and from the node 2 to 3 are directed edges labeled with $\triangleleft$ indicating the successor relation. Unlike nodes 1 and 3, there are α associations from node 2 to the nodes 4 and 5 which are labeled with `aperture`.

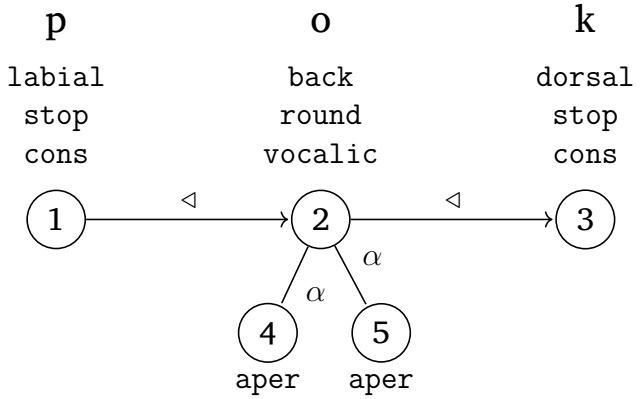


Figure 19.3: The $\mathfrak{R}$-structure of the hypothetical word [pok].

19.4 A Logical Transduction of Danish Vowel Lowering

The generalization we seek to describe is that front, non-high, short vowels lower when they are immediately followed by /r/. As mentioned, this process can be thought of as the increase of one degree of aperture, or the addition of one aperture particle. Adopting the idea that vowels are gradual along the scale of the aperture, I formalize the vowel lowering process with logical formulas in this section.

With the aforementioned representations in mind, I will now turn to transformation of the phonological process of vowel lowering. Any transformation inherently maps an underlying representation to a surface representation. For example, if underlying /er/ undergoes the transformation of vowel lowering it will surface as [ɛr]. In other words, the process adds an aperture particle to the vowel in the surface form. Figure 19.4 shows a graphical representation of the transformation /er/ $\rightarrow$ [ɛr]. I only

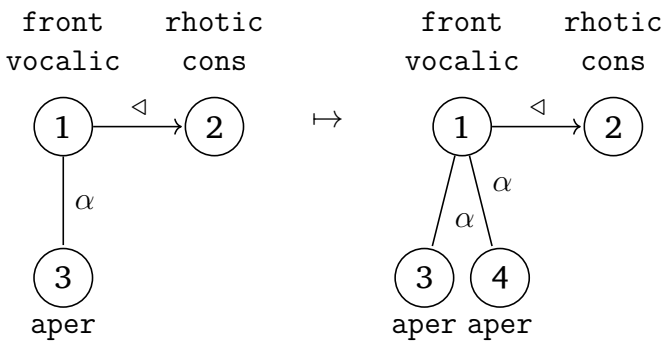


Figure 19.4: An example of transformation from /er/ to [ɛr]

represent the vowel and the following rhotic for convenience. The left side of the arrow represents the input, /er/, and the right side of the arrow represents the output, [ɛr]. Note that there is an additional aperture node on the output, node 4, which indicates that the output vowel is [ɛ] not [e].

I will now show how to construct the output structure from an input structure using formulas of First-Order logic to define the logical transduction as explained in Chapter 3. I will use the example of /er/ to [ɛr] as a running example, and show step by step how the formulas work together to build the $\mathfrak{R}$-structure for [ɛr] from the $\mathfrak{R}$-structure for /er/.

As Figure 19.4 shows, there are more domain elements in the output structure than in the input structure. Consequently, two copysets are required when defining this logical transduction.

$$C \stackrel{\text{def}}{=} \{1, 2\} \tag{19.2}$$

In other words, the workspace to build the output structure has six domain elements to work with, as shown in Figure 19.5. In Figure 19.5, and

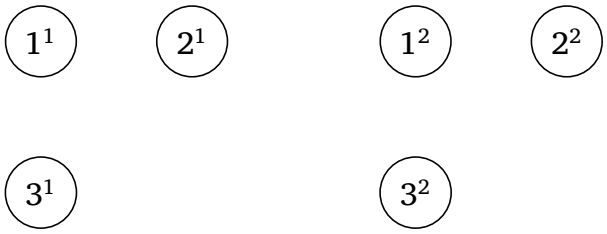


Figure 19.5: The potential domain elements in the output structure.

subsequent figures, the notation x^n indicates the nth copy of the domain element x.

Since there are two copysets, there must be two formulas for each unary relation U which determine whether the first and second copies of the domain elements in the output structure satisfy U, respectively. For the first copy, all of these are faithful. In other words for every unary relation U in $\mathfrak{R}$, we have a formula as shown in Equation 19.3.

$$\phi^1_{\mathtt{U}}(x) \stackrel{\text{def}}{=} \mathtt{U}(x) \tag{19.3}$$

The next formula in Equation 19.4 determines which domain elements are aperture particles in the second copyset of the output. In the second copyset, a domain element satisfies `aperture` only if its counterpart in the input structure satisfies `vocalic`, has at least one aperture particle (and so is not high), and is succeeded by a domain element satisfying `rhotic`.

$$\begin{aligned} \phi^2_{\mathtt{aper}}(x) \stackrel{\text{def}}{=} \quad & \mathtt{vocalic}(x) \wedge \exists(y)[\mathtt{aperture}(y) \wedge x \alpha y] \\ & \wedge \exists(z)[\mathtt{rhotic}(z) \wedge x \lhd z] \end{aligned} \tag{19.4}$$

Since high vowels do not have an aperture particle, Equation 19.4 will not create a new aperture particle for them.

All other unary relations U are defined to be `false` for the second copy. Consequently, given the input $\mathfrak{R}$-structure for /er/ as shown in the left in Figure 19.4, the domain elements in the output structure will have the unary relations as shown in Figure 19.6.

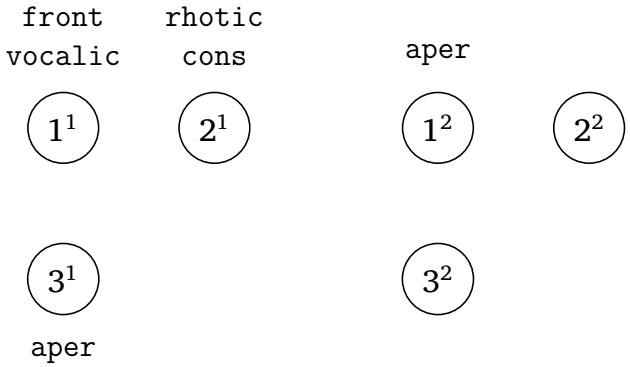


Figure 19.6: The potential domain elements in the output structure and the unary relations they satisfy.

Next I consider the binary relations. There will also be multiple formulas for them. The following equations define whether two domain elements in the first copyset of the output structure are related. Equation 19.5 defines the successor relation and Equation 19.6 defines the aperture relation.

$$\phi^{1,1}_{\lhd}(x,y) \stackrel{\text{def}}{=} \quad x \lhd y \tag{19.5}$$

$$\phi^{1,1}_{\alpha}(x,y) \stackrel{\text{def}}{=} \quad x \; \alpha \; y \tag{19.6}$$

Next I define the formula which effectively connects the new aperture particle to the vowel, which indicates it is lowered. Equation 19.7 says the $alpha$ relation will hold between the first copy of a pre-rhotic vocalic position and its second copy.

$$\phi^{1,2}_{\alpha}(x,y) \stackrel{\text{def}}{=} (x = y) \wedge \phi^{2}_{\texttt{aper}}(x) \tag{19.7}$$

Recall from Equation 19.4 that the second copy of domain element corresponding to the non-high, pre-rhotic vowel satisfies the unary relation `aperture`. In other words, the effect of Equation 19.7 is to connect new aperture particles to their corresponding non-high, pre-rhotic vowels in

the output structure. All the other binary relations are defined as `false` as shown below.

$$\phi^{1,2}_{\triangleleft}(x,y) \stackrel{\text{def}}{=} \phi^{2,1}_{\triangleleft}(x,y) \stackrel{\text{def}}{=} \phi^{2,2}_{\triangleleft}(x,y) \stackrel{\text{def}}{=} \phi^{2,1}_{\alpha}(x,y) \stackrel{\text{def}}{=} \phi^{2,2}_{\alpha}(x,y) \stackrel{\text{def}}{=} \texttt{false} \tag{19.8}$$

Consequently, given the input $\mathfrak{R}$-structure for /er/ as shown in the left in Figure 19.4, the domain elements in the output structure will have the unary and binary relations as shown in Figure 19.7. As defined in

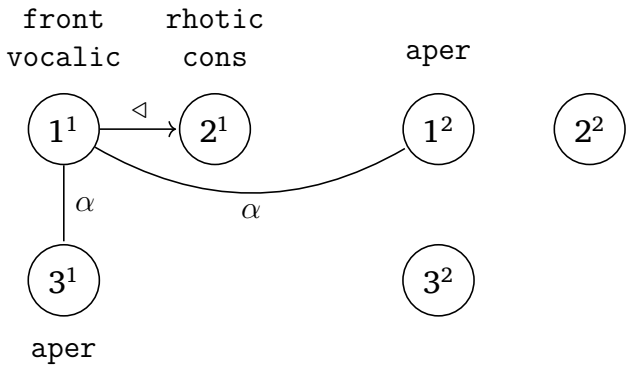


Figure 19.7: The potential domain elements in the output structure and the relations they satisfy.

Equation 19.7, the association is drawn between the node 1^1 and the node 1^2 since node 1 in both copysets corresponds to node 1 in the input structure which satisfies `vocalic` and whose successor element satisfies `rhotic`.

Then, we move to the clean-up phase by way of the licensing functions. There are two licensing formulas, one for each copyset. The only elements to be licensed are the ones that actually matter in the output structure. While every element in copy set 1 ought to be licensed no matter what, this is not the case for domain elements in the second copy set. For example, in Figure 19.7, the elements 2^2 and 3^2 should not be licensed. Essentially, the only elements in copy set 2 to be licensed are the ones that are new aperture particles. These licensing formulas are shown below in Equations 19.9 and 19.10.

$$\phi^{1}_{\texttt{lic}}(x) \stackrel{\text{def}}{=} \quad \texttt{true} \tag{19.9}$$

$$\phi^{2}_{\texttt{lic}}(x) \stackrel{\text{def}}{=} \quad \phi^{2}_{\texttt{aper}}(x) \tag{19.10}$$

The final result for these formulas for the input $\mathfrak{R}$-structure /er/ is represented in Figure 19.8. Only the node 1^2 is left from the second copyset, which is attached to the node 1^1. As a result, the output structure represents the lowered front vowel before the rhotic. It is clearly isomorphic to the $\mathfrak{R}$-structure /ɛr/ shown on the right hand side in Figure 19.4. More gener-

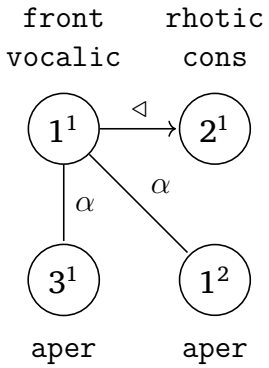


Figure 19.8: The structure output given the $\mathfrak{R}$-structure for /er/.

ally, these formulas provide a logical transduction for the vowel lowering in Danish.

19.5 Discussion

Some aspects of the above analysis merit further discussion. I begin with what the above transduction means for front round vowels and then for the back vowels. For front round vowels, the transduction works as it should. For example, consider the map from underlying form /œr/ to surface form [œr]. In the analysis above, the structure for /œr/ would have two aperture particles associated with a domain element satisfying the unary relations `vocalic` and `round`. Since `round` is faithfully transmitted to the output structure and does not play a role in any other formulas defining the transduction, another aperture particle will be associated to the vowel in the output structure, yielding the allophone [œ].

Like `round`, the unary relation `back` also does not occur in any of the formulas which determine the addition of an aperture particle. The table which Basbøll (2005, p. 151) presents, which shows the relation between the underlying and surface short full vowels that undergo lowering, does

not include /o/, suggesting it does not undergo lowering. It also shows /ɔ/ lowering to [ɒ]. Neither of these is accounted for under the current analysis. In order to exclude /o/, the formula expressed in Equation 19.4 would have to be adjusted. For /ɔ/ lowering to [ɒ], the process would have to add two aperture particles. The latter issue could be accounted for by assuming that /ɔ/ has three degrees of aperture underlyingly, and not two, but I note that in Basbøll's analysis (p. 156), /ɔ/ is the same distance from the maximally low back vowel as the vowels /ɛ,œ/.

Independently, in this analysis, the vowels with four degrees of aperture /ɑ,ɒ/ are not specified for [back]. The advantage for the current analysis was that the back feature does not have to change. Whether it matters or not for the analysis of the back vowels is unclear. For example, the interpretation of domain elements with four aperture particles which satisfy or do not satisfy `back` may be the same. In short, while this analysis focused on short, front vowels, extending or revising the analysis to account for back vowels, as well as other vowels in Danish, is a good project for future work.

Another way in which the analysis above can be improved is to better describe the phonological contexts where rhotics induce lowering. The pre-vocalic rhotic context is currently provided in the third conjunct of Equation 19.4. This conjunct can be revised to ensure that the rhotic is tautosyllabic, perhaps by including syllabic representations in $\mathfrak{R}$-structures (Strother-Garcia, 2019, Chapter 15), and to include post-rhotic contexts as appropriate.

Finally, I turn to the question of whether representing aperture directly as was done here is possible with rule-based and constraint-based frameworks. It is possible, but there is an overhead cost that has to be paid. For example, suppose a rule-based account represents vowels with degrees of aperture as in Table 19.4 and proposes the rule shown below.

(13) $[+\text{vocalic}, n \text{ aperture}] \longrightarrow [n+1 \text{ aperture}] \;/\; \_\_ \text{ r}$

One has to also explain what constitutes a well-formed rule, and how to interpret such rules. Similarly, in OT, if one could provide a marked structure that could be satisfied by the addition of a single degree of aperture, one would have to define faithfulness constraints for adding aperture particles and associating them. While there have been related efforts in developing OT analyses of tonal insertion and association (Yip, 2002), there remains no general theory of constraints in OT, much less

over non-linear structures as is the case here.

In contrast, the model-theoretic relational structures and logical transductions invoked here required no additional overhead. Logic and relational structures let us consider and compare alternative representations within a single analytical framework.

19.6 Conclusion

To conclude, this chapter has analyzed vowel lowering in Danish, which is often considered an opaque phonological process because it exemplifies a counterfeeding chain. With the traditional binary feature system, neither phonological analysis in rule-based accounts nor in constraint-based ones captures or *mirrors* the transformation intuitively. In this chapter, I represented vowels with *aperture particles* instead of binary features for height to account for the *gradual oppositions* of Danish vowels. I provided a FO-definable transduction in terms of these representations, which accurately accounted for the phonological process as well as mirroring it. Essentially, the analysis added a single aperture particle to target vowels in the appropriate environment.

Chapter 20

Logical approximations of lexical strata, cophonologies, and cyclicity

Hossep Dolatian and Jeffrey Heinz

20.1 Introduction

In an abstract sense, morphophonology studies the pronunciations of morphemes based on morphological context. Cross-linguistically, alternations in pronunciation are governed by universal and language-specific principles. These principles refer to morphology, prosody, phonological rule domains (lexical cophonologies or strata), and derivational history. Most work on computational or mathematical morphology/phonology focuses on formalizing an individual aspect of an individual principle. This chapter asks a larger question: *How can we formalize the interaction of all these principles together?* We answer this question with logical transductions and model-theoretic representations of morphophonological data structures. Our answer is that the use of model-theoretic representations provides a flexible, universal mathematical framework for integrating a variety of linear and nonlinear linguistic representations. When these representations are couched in Monadic Second Order (MSO) logic, the system is capable of expressing these principles and their interactions. In fact, we find that even weaker logics may be sufficient. The morphophonology of Armenian

is used to illustrate and explain this analysis.

There are many different theories of morphophonology. Scheer (2011) provides an in-depth historical catalog. Theories are often evaluated in terms of empirical coverage and conceptual elegance. In terms of their formulation, the differences boil down to differences in either combinatorial or organizational properties. Combinatorial differences refer to the individual types of representations or transformations that the theory uses: morphological trees, morphological features, prosodic trees, cophonologies, and so on. Organizational differences refer to how these factors are brought together under some kind of architecture or pipeline.

An important organizational parameter is whether the theory is interactionist like Lexical Phonology (Kiparsky, 1983) vs. non-interactionist like SPE (Chomsky and Halle, 1968). An interactionist theory (also called a cyclic theory) is one where the morphological and phonological components of grammar are interleaved, such that morphology feeds phonology, which then feeds another cycle of morphology. In contrast, non-interactionist theories ban such interleaving: morphology feeds phonology in only one direction. This chapter presents an analysis of interactionist theories. This choice is motivated on the one hand by our belief that of the two kinds of theories, interactionism is the greater challenge to formalize, and on the other by the empirical arguments for cyclicity (Bermúdez-Otero, 2011).

Cyclicity and interactionism are commonly employed in early and contemporary work in phonology and morphology (Scheer, 2011). However, these concepts are difficult to computationally formalize and develop (Sproat, 1992, 208). Intuitively, the problem is because of the following two challenges in computational treatments of phonology and morphology.

2. *Challenges facing cyclic phonology*

   (a) **Linearity**: Most computational systems work over linear representations, while morphophonological analyses often invoke nonlinear representations.

   (b) **Boundedness**: A restrictive grammar cannot use the unbounded application of phonological rules. But the combination of cyclic rule application in phonology and of unbounded processes in morphology imply unbounded rule application.

In this chapter, we utilize logical transductions over model-theoretic representations to address these two challenges. The issue of Linearity

is resolved because model-theoretic representations are flexible enough to operate over a combination of strings, trees, and other data structures. This allows us to provide a computational realization of inherently derivational or transformational phenomena such as resyllabification, prosodic phonology (Nespor and Vogel, 1986; Booij and Lieber, 1993), cyclic levels (Bermúdez-Otero, 2011), cophonologies (Inkelas, 2014), and tree structure (Selkirk, 1982), such that the computational realization is faithful to the representations that linguistic theories posit.

The logical formalization addresses the problem of boundedness, but does not resolve it completely. We enrich our input representations with information on the derivational history of words (cf. Potts and Pullum, 2002). This allows us to effectively compute a single cyclic application and interactionism (Cole, 1995a) with a single logical trandsuction. We can repeat this transduction as many times as needed in order to generate increasingly complex forms.

Nonetheless, cyclicity prevents us from realizing a single model that expresses the totality of the system. Even though our formalization 'works' in that it can model cyclic processes, it comes at the cost of assuming a strictly 'run-time' approach to computing cyclic phonology. An independent apparatus (control system) is needed that will apply the grammar as many times as needed to compute the correct number of cycles. In this sense, the present analysis still suffers from the Unboundedness problem.

This chapter is organized as follows. We briefly go through the main ideas behind cyclic phonology (§20.2.1), and discuss a case study of cyclic phonology in Armenian (§20.2.2). We formalize the components of cyclic phonology in §20.3 We define a large fragment of a logical formalization for generating cyclic phonology, including aspects such as morphological structure, syllabification, prosodic parsing, strata, and cophonologies (§20.4-20.6). In §20.7, we take a step back and evaluate our system and re-evaluate the main challenges that cyclicity causes for any computational formalization.

20.2 Background

This section reviews the theoretical literature and main architectural assumptions in cyclic theories of phonology (§20.2.1). Then we present an in-depth case study of Armenian, taken from the first author's theoretical

work on the language (§20.2.2). We use Armenian not because Armenian cyclic phonology is unique, but simply because it is a non-trivial case which helps illustrate both coarser and finer aspects of the formalization. Readers are encouraged to apply the formalization used here for Armenian to other cases of cyclic phonology.

20.2.1 What are cyclicity and strata

Interactionist theories were primarily developed to capture phonological effects on morphology, such that the form of some complex word $A+B$ was affected by the phonology of the base word A. Alongside such cyclic effects, interactionist theories likewise try to capture affix-specific phonological effects. Cross-linguistically, it is common to find that some sequences of morphemes trigger a different set of phonological rules than some other sequence of morphemes.

To illustrate, consider bound roots and stress shift in English (Siegel, 1974; Allen, 1979). The word *sensitive* is formed by a bound root *sens-* and a suffix *-itive*. It is well-known that some suffixes like *-ity* cause stress shift: *sénsitive* and *sènsitív-ity*. Such suffixes are called Level 1 suffixes or stem-level suffixes. In contrast, suffixes like *-ness* do not trigger stress shift: *sśensitive-ness*. Such suffixes are called Level 2 or word-level suffixes. See Newell (2021) for a review. A common analysis of these patterns in English utilizes cyclicity and strata. A derivation of *sensitivity* and *sensitiveness* is shown in Table 20.1. Generating the word *sénsitive-ness* goes through two cycles. In Cycle 1 (stem-level), the base form is generated: *sens-itive*. This word forms a prosodic word (PWord), and the stem-level (Level 1) cophonology places stress. In the next cycle (word-level; Cycle 3 in the figure), the suffix *-ness* is added and stays outside the base's Prosodic Word. The word-level (Level 2) cophonology applies which has no stress shift rule. In contrast, for *sènsitív-ity*, the word undergoes two stem-level cycles: one for the base, and one for adding *-ity*. Stress shift applies across the stem-level cycles, and the Prosodic Word expands. A vacuous word-level cycle then applies.

The organizational aspects of the derivation in Table 20.1 are largely independent of the combinatorial aspects, which can be modeled in different theoretical frameworks, whether rule-based or constraint-based.

When formalizing such a derivation one recognizes that cyclic deriva-

Input		/ sense, -itive, -ity /	/ sense, -itive, -ness /
Cycle 1			
MORPHOLOGY	Stem Formation	sens-itive	sens-itive
PROSODY	Syllabification	sen.si.tive	sen.si.tive
	PWord Formation	$(\text{sen.si.tive})_w$	$(\text{sen.si.tive})_w$
LEVEL 1 PHONOLOGY	Stress Assignment	sén.si.tive	sén.si.tive
Cycle 2			
MORPHOLOGY	/-ity/ Affixation	sén.si.tive /-ity/	
PROSODY	Syllabification	sén.si.ti.vi.ty	
	PWord Formation	$(\text{sén.si.ti.vi.ty})_w$	
LEVEL 1 PHONOLOGY	Stress Assignment	sèn.si.tí.vi.ty	
Cycle 3			
MORPHOLOGY	/-ness/ Affixation		sén.si.tive /-ness/
PROSODY	Syllabification		sén.si.tive.ness
	PWord Formation		$(\text{sén.si.tive})_w$.ness
LEVEL 2 PHONOLOGY	Stress Assignment		

Table 20.1: Cyclic derivation of the English words 'sensitivity' and 'sensitiveness.' Prosodic word boundaries are only shown in the PWord Formation step.

tions involve two types of implicit steps. The first implicit step is that "something" has to tell us what morphology to add in each cycle. In other words, aspects of the derivational history need to be present. This information is buried in the underlying form of the input. The second implicit step is that, after the morphology, "something" has to tell us what prosodic or phonological rules to apply based on the newly generated morphological structure. This information is buried in the morphological derivation from each cycle. Throughout this paper we refer to the first implicit step as the **operation** and the second one as **examination**.

The implicit steps of operation and examination become more apparant when a logical formalization of cyclic phonology is developed. For now, the main idea is that the operation step keeps track of one's place in the derivational history of a word. The examination step involves identifying

the morphological information to apply the correct prosodic/phonological rules. With this in mind, we turn to the main case study from Armenian.

20.2.2 Illustrating cyclicity with Armenian

Armenian is an agglutinative Indo-European language. We briefly introduce aspects of Armenian prosody, morphology, and phonology which we later formalize, which showcase the role of cyclicity and phonological rule domains.

Armenian has final stress as shown in (3) (Vaux, 1998). If the final vowel is a schwa, stress is placed to the penultimate full vowel (3b). When stressed high vowels lose stress, they are reduced in a process of destressed reduction (4). The high vowel is generally deleted (4b). But if deletion would create an unsyllabifiable consonant cluster, then the high vowel is reduced to a schwa (4d).

3. (a) kʰórɾd͡z ‘work’
   (b) kʰórɾd͡z-ə ‘work-DEF’

4. (a) tʰeʁín ‘yellow’
   (b) tʰeʁn-orág ‘yellowish’
   (c) hín ‘old’
   (d) hən-utʰjún ‘oldness’

Destressed reduction targets high vowels which had stress in the base but not in derived forms. Thus, unstressed vowels in the base do not reduce in derived forms (5a). Furthermore, reduction can apply an unbounded number of times based on how much morphology we have. It can apply in roots (5b), suffixes (5c), and in compounds (5d).

5. (a) məχitʰáɾ ‘comforter’
   məχitʰɑɾ-él ‘to comfort’
   *məχtʰɑɾ-él
   (b) d͡zín ‘birth’
   d͡zən-únʰt ‘birth’
   d͡zən-ənʰt-ɑgán ‘generative’

(c) ásk ‘nation’
ɑsk-ɑjín ‘national’
ɑsk-ɑjn-u^{h}tjún ‘nationalist’

(d) k^{h}íɾ ‘writing’
k^{h}əɾ-íʧ ‘pen’
dúp ‘box’
k^{h}əɾ-ʧ-ɑ-dúp ‘pencil-box’

Altogether, only high vowels which were stressed at some earlier point in the derivation undergo reduction. The reduction facts provide evidence that stress is assigned cyclically.

Morphologically, not every affix triggers reduction. Derivational suffixes and compounding trigger it, while inflectional suffixes do not (6 and 7).

6. k^{h}íɾ ‘writing’
k^{h}əɾ-íʧ ‘pen’
k^{h}əɾ-iʧ-é ‘pen-ABL’

7. t^{h}eʁín ‘yellow’
t^{h}eʁn-ál ‘to yellow’
t^{h}eʁin-óv ‘yellow-INS’

Dolatian analyzes reduction with Lexical Phonology (Dolatian, 2020a) and Stratal OT (Dolatian, 2020b). Different types of morphology create different phonological domains. Free-standing roots and derivational morphology create morphological stems (MStems, MS).[1] These trigger the stem-level (SLevel) phonology of stress and reduction. In contrast, inflectional morphology creates morphological words (MWords, MW) and triggers word-level phonological stress assignment, which does not induce reduction. To illustrate, consider Figure 20.1, which shows three tree structures associated with the the inflected term *k^{h}əɾ-iʧ-e* in (6). One tree structure shows the morphological derivation, one showcases the different

[1]To illustrate morphological functions, we assume that free-standing roots get their syntactic category via a zero morpheme N (Marantz, 2007). We don't use lowercase letters *n* to avoid ambiguity. For simpler illustration, overt derivational and inflectional suffixes are labelled as DER and INF.

phonological rule domains of this word, and one presents the prosodic structure.[2]

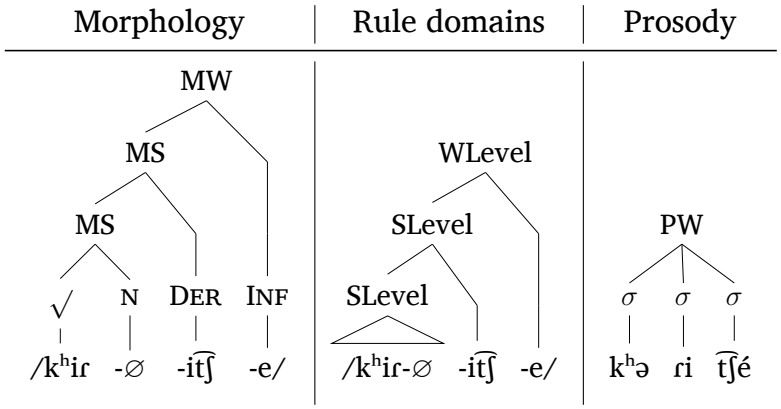


Figure 20.1: Morphological and prosodic structures and rule domains for [kʰəɾ-i͡tʃ-e] 'pen-ABL'. Prosodic word boundaries are only shown in the PWord Formation step.

Figure 20.2 illustrates a derivation for the inflected form *kʰər-i͡tʃ-e*. The root undergoes one cycle of the stem-level phonology to get stressed: *kʰ́ɾ*. The derivational suffix triggers a new cycle of the stem-level phonology to get stressed and to reduce the root: *kʰər-i͡tʃ́*. Between stress shift and reduction, we mark the destressed vowel with the diacritic *ĭ*. Finally, the inflectional suffix triggers a cycle of the word-level phonology to get stressed but without causing reduction: *kʰər-i͡tʃ-é*. This last cycle generates a prosodic word (PWord).

In sum, the Armenian data motivates both morphological and prosodic structure. These structures cyclically create phonological domains. The next section formalizes a stratal architecture for computing cyclic phonology and morphology.

20.3 Components of cyclic phonology

As with any computational formalism, logical transductions require precise and well-defined representations and operations. In order to formalize

[2]Between syllables and prosodic words, Dolatian (2020a,b) uses an additional prosodic constituent called the prosodic stem (Downing, 1999). We set this aside here.

Input		/kʰiɾ, -∅, -it͡ʃ, -e/
Cycle 1		
MORPHOLOGY	Stem Formation	kʰiɾ -∅
PROSODY	Syllabification	kʰiɾ
STEM LEVEL PHONOLOGY	Stress Assignment	kʰíɾ
	Reduction	
Cycle 2		
MORPHOLOGY	Affixation	kʰíɾ-/it͡ʃ/
PROSODY	Syllabification	kʰí.ɾit͡ʃ
STEM LEVEL PHONOLOGY	Stress Assignment	kʰǐ.ɾít͡ʃ
	Reduction	kʰə.ɾít͡ʃ
Cycle 3		
MORPHOLOGY	Affixation	kʰə.ɾít͡ʃ-/e/
PROSODY	Syllabification	kʰə.ɾí.t͡ʃe
	PWord Formation	(kʰə.ɾí.t͡ʃe)$_w$
WORD LEVEL PHONOLOGY	Stress Assignment	kʰə.ɾǐ.t͡ʃé
Output		kʰəɾit͡ʃé

Table 20.2: Derivation table for the cyclic derivation of the inflected form [kʰəɾit͡ʃe] 'pen-ABL'.

cyclic phonology, we first set up the components of a cyclic architecture. Using the Armenian word *kʰər-itʃ-e* 'pen-ABL' as a running example, we formalize the individual morphological, prosodic, and phonological operations across 3 cycles (§20.4-20.6). This section provides an overview.

In the previous section §20.2.2, we used an informal illustration in the form of a derivational table (20.2). This table encodes steps or components for morphology, prosody, and phonology where the operation and examination steps were left implicit. In order to formalize cyclic phonology, we develop an elaborated architecture in (20.2) and make the operation and examination steps explicit.

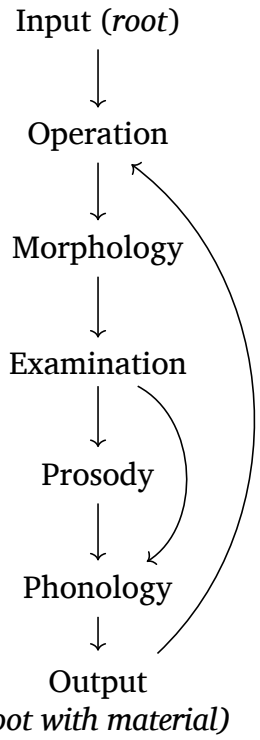


Figure 20.2: Elaborated architecture for cyclic phonology.

The figure in (20.2) shows the three expected modules of **Morphology**, **Prosody**, and **Phonology**. They are intermingled with two additional components: the **Operation** and the the **Examination**. These two components are used to streamline the choice of morphological, prosodic, and

phonological functions. Together, these 5 components comprise a single cycle. The output of this cycle is then fed back to itself to start a new cycle.

In a cycle, the first component is the **Operation**. It encodes the word's derivational order, i.e., a log of what morphological functions have applied or will be applied. This feeds the **Morphology** component which adds morphological structure. The morphology is then examined to find global properties of the input in the **Examination** stage. These properties determine what prosodic parse to apply, and what phonological rule domains to trigger. These properties are encapsulated into a global constant called the SETTINGS. With the properties in place, we then apply the **Prosody** and **Phonology**.

Before going further, we wish to emphasize that although we use procedural language, the logical methods here are declarative. For example, we often use procedural terms like 'then.' This is only for ease of exposition. There is no fixed ordering between logical statements within a single transduction. While five distinct transductions are specified above, one for each component, it is well-known that logical transductions are closed under composition. In other words, there is a single logical transduction that computes a single cycle, and within this transduction, there is no 'ordering' per se.

Having outlined our formalization of the cyclic architecture, the next sections formalize the logical transductions which comprise these 5 components. Using *kʰər-it͡ʃ-e* 'pen-ABL' as our running example, we showcase these transductions over the course of three cycles. Because there are many components to the derivation, it is burdensome to present the model-theoretic signature for the structure at this time. Each section adds more unary labels and binary relations to our model. We provide the signature with some notes in an appendix to this chapter. In addition, for each transduction, we only provide a fragment relevant to the running example. As we explain, the remainder of the transduction must be filled in on the basis of other aspects of the grammars (such as other morphemes).

20.4 First cycle: Generating a stem

In the first cycle, the input is an unsyllabified and unstressed root $\sqrt{\text{kʰir}}$. The root lacks any syntactic category. The output is a fully syllabified and stressed stem *kʰír* in a morpho-phonological structure. To formalize

this transformation, we first explain the input structure (§20.4.1). We process the derivational order (Operation list) in order to determine what morphological rules to apply (§20.4.2). We then apply the morphology (§20.4.3), examination (§20.4.4), prosody of syllabification (§20.4.5), and phonological rule domains (§20.4.6).

20.4.1 Morphological and phonological representations

As said, the input to the first cycle is a simple root $\sqrt{\text{k}^{\text{h}}\text{ir}}$. Phonologically, the segments of the root are in a linear order based on immediate successor. Morphologically, the segments are dominated by a root morpheme MRoot. Essentially, the linguistic representation combines a morphological tree with a linear phonological sequence of segments. The model-theoretic representation expresses this faithfully with a successor relation, dominance relation, and unary relations to identify the various positions.

As the cycle unfolds, this morphophonological structure will be modified. In order to know how to modify it, the model-theoretic representation includes two additional types of structures: the SETTINGS and the **Operation List**. These structures encapsulate information about what morphological and phonological functions to apply.

Figure 20.3 visualizes this model-theoretic structure. We adopt a different set of conventions for the illustrations of model-theoretic structures than the other chapters in this book in order to facilitate reading them. We explain this figure and these conventions in more detail below. In general we use the language of graphs where **nodes** correspond to domain elements in the structure and labeled **edges** connecting nodes correspond to binary relations in the structure.

For the phonology, the positions representing the segments *k,i,r* satisfy the unary relations $\texttt{k}, \texttt{i}, \texttt{r}$ respectively.[3] Each position likewise has the label `segment` in order to distinguish them from other kinds of structure in the representation. As for the morphology, the root has the labels $\texttt{morpheme}(x)$ and $\texttt{MRoot}(x)$. In general, the unary relations `segment` and $\texttt{morpheme}(x)$ are not shown in the figures.

Instead of labeling the nodes in the graph with the indices of the domain elements, we label the nodes of the graph with the most important unary

[3] We use segmental labels instead of features to facilitate discussion, but as discussed in Chapter 2, phonological features could be used instead.

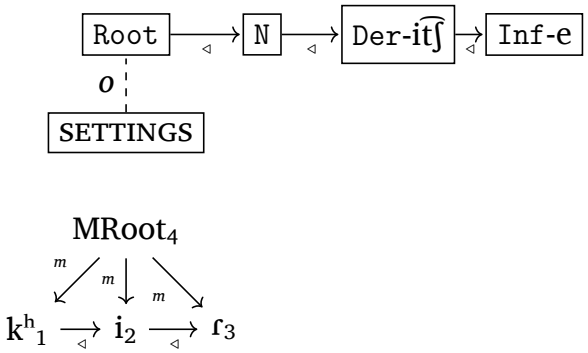


Figure 20.3: Input to Cycle 1: *kʰir* 'write' (root).

relation it satisfies. We include the indexing of domain element as a subscript to this label.

The segments are connected via the binary successor relation. The successor relation is specialized for segments only: succ:seg(x, y) (shown as ◁-labeled edges in Figure 20.3). The root is connected to its segments via the binary relation of morphological dominance MDom(x, y) (shown as *m*-labeled edges in Figure 20.3).[4]

Above the tree, the SETTINGS node is a **constant** (shown as a rectangle). In relational structures, a constant is a domain element with a unique name present in every structure. As will be explained, this domain element will satisfy various properties in the course of the cyclic derivation in order to fulfill the examination step.

The SETTINGS is connected to a sequence of nodes called the **Operation List**. The list encodes the derivational sequence to generate the word. Each of these nodes represents an **Operation** (shown in rectangles). Nodes which represent operations satisfy the unary relation oper(x). Operation nodes are connected via the binary relation of successor, specialized for operation nodes: succ:oper(x). The main label for each operation node is the name of the type of morphological function that it triggers. In the running example in Figure 20.3, the Operation List carries the following information:

1. it begins with root: Op:Root(x),
2. then there is zero-nominalization: Op:N(x),

[4]Dolatian (2020a) includes a morph level between segments and morphemes. We omit it for brevity.

3. followed by overt nominalization with the suffix *-itʃ*: Op:Der-it$\widehat{\text{ʃ}}(x)$,
4. and finally inflection occurs with the suffix *-e*: Op:Inf-e(x).

The SETTINGS is connected to the first operation via the binary relation of operate_at(SETTINGS, y) (shown as an *o*-labeled edge). There are finitely many syntactic categories and so there are finitely many primitive unary relations like Op:N(x) (such as Op:A(x) for adjectives). There are also finitely many derivational and inflectional morphemes like Op:Der-it$\widehat{\text{ʃ}}(x)$ and Op:Inf-e(x). Each has its own unary relation in the signature of the relational structure.

In the diagrams, we omit the prefix "op:" from these operation labels. In general, we do not show indexes for the SETTINGS and operation nodes.

20.4.2 Operation: Encoding derivational history

With this input, the first cycle begins with the Operation component (see Figure 20.2). Essentially, the Operation component advances the Settings to the next operation in the operation list. Formally, the Operation component is a logical transduction that uses a copy set of size 1. The input and output of this transduction for the structure in Figure 20.3 are shown below in Figure 20.4. In the input, SETTINGS constant is connected to the first operation node: Op:Root. In the output, the SETTINGS constant is reconnected with the subsequent operation node: Op:N. In the diagram, the indices are preprended with a number indicating the copy set. We adopt a convention where the "copy set" of the input is 0.

In the Operation step, all labels and binary relations are faithfully outputted except for the operate_at(x, y) relation. Beyond this section, we do not show the formulas for faithfully outputting elements. They all follow the same format as below. For all unary relations u$\in U$, and for all binary relations b in $B/\{$operate_at$\}$, we have the following formulas.

$$\phi^{1}_{\mathtt{u}}(x) \stackrel{\text{def}}{=} \mathtt{u}(x) \tag{20.1}$$

$$\phi^{1,1}_{\mathtt{b}}(x, y) \stackrel{\text{def}}{=} \mathtt{b}(x, y) \tag{20.2}$$

To move along the operation list, we use the predicate defined in (20.3) to find the current node connected to the SETTINGS, and the predicate in (20.4) to find the subsequent operation node. The SETTINGS node is then

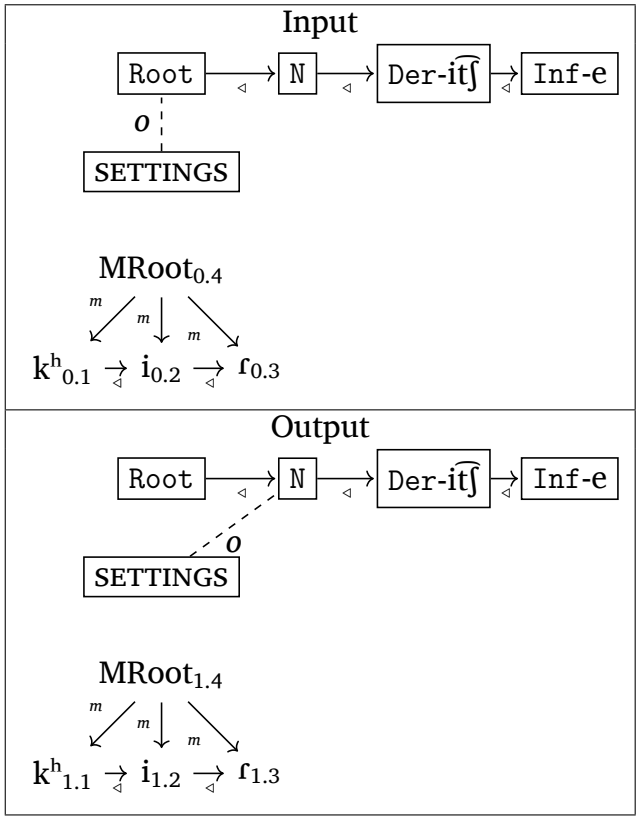


Figure 20.4: Operation: Proceeding on the operation list for the root k^hir

made to point to the subsequent node in the definition of $\phi_{\texttt{operate\_at}}$ (20.5).

$$\texttt{Op:curr}\,(x) \stackrel{\text{def}}{=} \texttt{Op}(x) \wedge \texttt{operate\_at}(\textsc{settings}, x) \quad (20.3)$$

$$\texttt{Op:next}\,(x) \stackrel{\text{def}}{=} \texttt{Op}(x) \wedge \exists y[\,\texttt{Op:curr}\,(y) \wedge \texttt{succ:Oper}(y, x)] \quad (20.4)$$

$$\phi^{1,1}_{\texttt{operate\_at}}(x, y) \stackrel{\text{def}}{=} x = \textsc{settings} \wedge \texttt{Op:next}\,(y) \quad (20.5)$$

The logical transduction defined above points the SETTINGS constant to a new operation and leaves everything else the same. The next stage is to apply the morphological function designated by this new operation. Every cycle starts with this logical transduction for the Operation step, defined by the above formulas. Note that this means that if the SETTINGS currently points to the last item in the list, then application of this transduction results in the SETTINGS not being related to anything. In principle, this could be used as a flag to know that the derivation has ended.

Turning now to the licensing functions, we adopt a simple strategy here and for all subsequent transductions: unliscensed elements are ones that satisfy no unary relations. If U is the set of unary relations, we can write the following equation for all copies c in the copy set C.

$$\phi^{c}_{license}(x) = \bigvee_{u \in U} u(x) \quad (20.6)$$

Consequently, elements satisfying no unary relation are eliminated.

20.4.3 Morphology: Morphological functions

A language consists of a finite number of possible morphological processes which are applied in some order. We formalize these morphological processes as logical transductions that reference the operation list. We examine the label of the current operation node, and we apply the *type* of morphological process which this node is labeled for. For our running example, the morphology adds a zero affix -∅ to create an MStem. We define a logical transduction with a copy set of size at least 3 for reasons explained below. Figure 20.5 illustrates the application of this transduction to the running example.

We use the predicate in (20.7) to check if the current operation node has the label Op:N for zero-nominalization. We also use the predicate $\texttt{MTop}\,(x)$

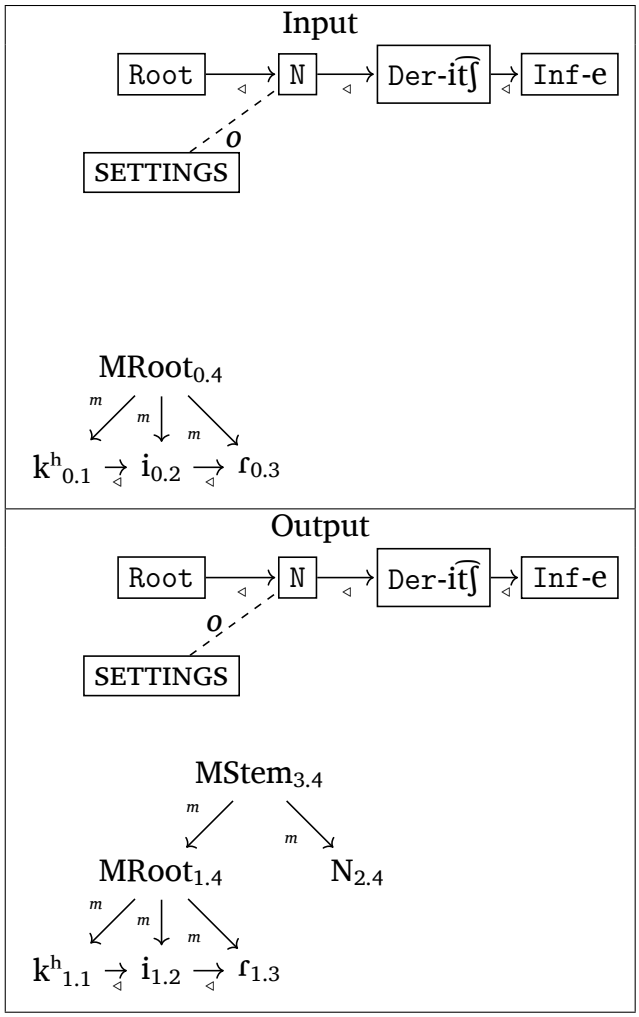


Figure 20.5: Morphology: Adding a covert nominalizer in *kʰirˑ-∅*.

(20.8) to find the morphologically-topmost node in the input, i.e., the morphological node which is at the top of the tree ($\text{MRoot}_{0.4}$).

$$\texttt{CrntOp:N} \overset{\text{def}}{=} \exists y[\texttt{Op}(y) \wedge \texttt{Op:curr}\,(y) \wedge \texttt{Op:N}(y)] \tag{20.7}$$

$$\texttt{MTop}\,(x) \overset{\text{def}}{=} \texttt{MNode}(x) \wedge \neg\exists y[\texttt{MDom}(y, x)] \tag{20.8}$$

In general, $\texttt{CrntOp:N}$ is just one of several that are defined based on the finite number of derivational and inflectional morphemes categories. For example, there is a zero suffix which turns roots into adjectives, as in /uɾɑχ/ 'happy,' and thus there is a predicate $\texttt{CrntOp:A}$ defined as in Equation 20.7 but using the atomic relation Op:A instead Op:N. Generally, if M denotes the finite set of morphological derivational and inflectional operations, then for all $\texttt{M} \in M$, there is a predicate $\texttt{CrntOp:M}$ defined as above where Op:N is replaced with Op:M.

Given these predicates, we then output the base *kʰir* in Copy 1. This is accomplished by making the copy 1 formulas fully faithful as discussed with respect to Equations 20.1 and 20.2. We output the suffix morpheme $N_{2.4}$ in Copy 2 and the new $MStem_{3.4}$ in Copy 3 as output correspondents of the morphologically-topmost node $MRoot_{0.4}$. The morpheme has the labels N because it assigns the syntactic category, while the MStem has the label MStem. The MStem dominates its suffix and the base's MRoot.

While the discussion so far illustrates the application of the transduction to the specific example, the transduction must be written in a way to accomodate all derivational and inflectional morphemes, of which there are only finitely many. We therefore construct a finite set of predicates for the different syntactic categories as shown below. For all $\texttt{M} \in M$, we define each $\phi^2_{\texttt{M}}(x)$ as follows.

$$\phi^2_{\texttt{M}}(x) \overset{\text{def}}{=} \texttt{MTop}\,(x) \wedge \texttt{CrntOp:M} \tag{20.9}$$

(20.10)

Consequently, $\phi^2_{\texttt{N}}(x)$ is defined as $\texttt{MTop}\,(x) \wedge \texttt{CrntOp:N}$. This formula is responsible for labeling node $N_{2.4}$ in Figure 20.5.

We use the third copy to indicate the node labeled $\phi_{\texttt{MStem}}$ in the output. In Figure 20.5, this is the node indexed $N_{3.4}$. The fact this node is labeled

$\phi_{\texttt{MStem}}$ instead of $\phi_{\texttt{MWord}}$ is because zero nominalization Op:N is a derivational, and not an inflectional suffix. We partition the set M into two disjoint subsets MD for the derivational morphemes, and MI for the inflectional morphemes. The former create stems, and the latter create words, as indicated in Equations 20.11 and 20.12, respectively below.

$$\phi^{3}_{\texttt{MStem}}(x) \stackrel{\text{def}}{=} \quad \texttt{MTop}\,(x) \wedge \bigvee_{\texttt{M}\in MD} \texttt{CrntOp:M} \tag{20.11}$$

$$\phi^{3}_{\texttt{MWord}}(x) \stackrel{\text{def}}{=} \quad \texttt{MTop}\,(x) \wedge \bigvee_{\texttt{M}\in MI} \texttt{CrntOp:M} \tag{20.12}$$

The formulas below establish the morphological dominance relations. The way we have set things up, the 3rd copy of the morphological topmost input node will dominate both the 2nd and 1st copy of the morphological topmost input node.

$$\phi^{3,2}_{\texttt{MDom}}(x,y) \stackrel{\text{def}}{=} \quad \texttt{MTop}\,(x) \wedge \texttt{MTop}\,(y) \tag{20.13}$$

$$\phi^{3,1}_{\texttt{MDom}}(x,y) \stackrel{\text{def}}{=} \quad \texttt{MTop}\,(x) \wedge \texttt{MTop}\,(y) \tag{20.14}$$

With regards to copy sets, 2 and 3, there are additional unary and binary relations we have not yet discussed. Generally, such formulas will be set to false. For example, $\phi^{2,3}_{\texttt{MDom}}(x,y) \stackrel{\text{def}}{=} \texttt{false}$. Formally, for all unary relations $\texttt{u}\in U$, and for all binary relations b in $B/\{\texttt{operate\_at}\}$ which have not been previously defined, and whenever $c > 1$ or $c' > 1$, we define them as below (cf. Equations 20.1 and 20.2).

$$\phi^{c}_{\texttt{u}}(x) \stackrel{\text{def}}{=} \quad \texttt{false} \tag{20.15}$$

$$\phi^{c,c'}_{\texttt{b}}(x,y) \stackrel{\text{def}}{=} \quad \texttt{false} \tag{20.16}$$

To conclude this section, we return to the issue of the size of the copy set. At the beginning of this section, we wrote that the size of the copy set is "at least 3" but we did not specify its exact size. What is the size of the copy set of the logical transduction discussed here that adds a morpheme? In the running example, the added morpheme's phonetic content was null. As a result, the output only needed to grow in size by two elements (nodes 2.4 and 3.4 in Figure 20.5). This is why we said that the copy set is at least size 3. However, if the added morpheme's phonetic content is not

null, then the output structure would include more nodes to represent this phonetic content. The reason why the copy set is "at least 3" is that we utilize additional copies to provide this phonetic content. In principle, the derivational and inflectional affixes could have phonetic content of any length. However, because there are finitely many derivational and inflectional affixes there is a largest one. Let it's size be n. Then the size of the copy set of this logical transduction will be $n + 3$. Here, where the affix was covert with size 0, three copies were sufficient. We explain this value in §20.5.1, which discusses how the formulas in this transduction accomodate the derivational suffix [it͡ʃ] in the second cycle of the running example.

20.4.4 Examination: What to parse and what rules to apply

Following the Morphology stage, the next stage is Examination. Examination is a logical transduction with a copy set of size 1. In Examination, we examine the morphological structure and determine what prosodic parses and what phonological rules to later apply. This is done by examining the morphological and prosodic context of the morphologically-topmost node. Based on this context, we encapsulate properties onto the SETTINGS node. We show the input and output of the running example in Figure 20.6 below. For convenience, we reindex the domain elements between stages. For example, the node labeled MStem now has the index 0.6 in the input side of Figure 20.6 instead of 2.4, which was its index in the output structure of Figure 20.5.

For the root *kʰir*, the morphology created an MStem and not an MWord. Thus we should not create a PWord in Cycle 1. We should apply the stem-level phonology, not the word-level phonology. In linguistic theory, the separation of stem-level and word-level phonology is formalized with separate cophonologies or levels (Inkelas, 2014). We formalize cophonologies with the unary relations `Cophon:SLevel` and `Cophon:WLevel`.

The formulas below examine whether the topmost node in the morphological tree satsfies `MStem` or `MWord` to deterimine whether the SETTINGS

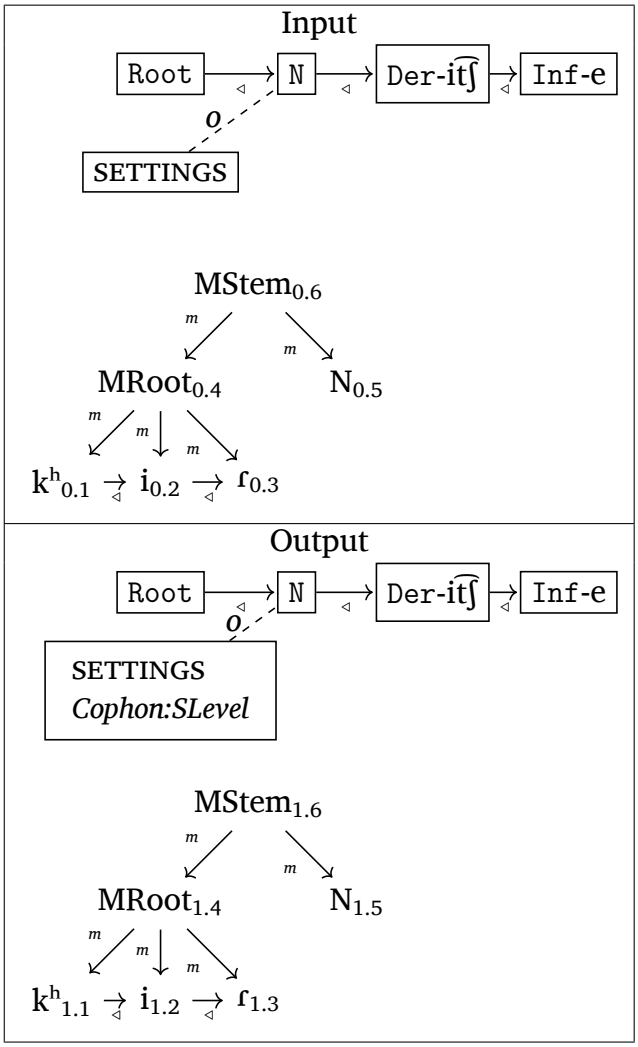


Figure 20.6: Examination: Setting the instructions for the prosody and phonology *kʰir*.

node satisfies Cophon:SLevel or Cophon:SLevel, respectively.

$$\phi_{\texttt{Cophon:SLevel}}(x) \stackrel{\text{def}}{=} \quad x = \textsc{settings} \wedge \texttt{MTop}(x) \wedge \texttt{MStem}(x) \tag{20.17}$$

$$\phi_{\texttt{Cophon:WLevel}}(x) \stackrel{\text{def}}{=} \quad x = \textsc{settings} \wedge \texttt{MTop}(x) \wedge \texttt{MWord}(x) \tag{20.18}$$

The output side of Figure 20.6 labels the inside of SETTINGS box with the unary relation it satisfies. For *kʰir*, the SETTINGS inherits the stem-level feature from the topmost node $MStem_{0.6}$.

Generally, there can be other properties to encode on SETTINGS. These will be defined when they are introduced. With Examination completed, we move on to the Prosody and Phonology.

20.4.5 Prosody: Syllabification

In the Prosody component, we apply syllabification, resyllabification, syllable linearization, and the generation of higher-level prosodic constituents like the prosodic word. These processes are each modeled by their own logical transduction which are composed into a single transduction. This is illustrated in Figure 20.7. For the stem *kʰir* in Cycle 1, the only relevant

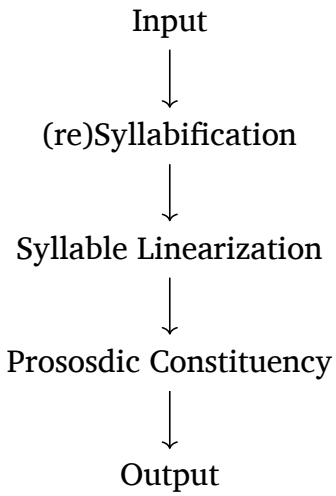


Figure 20.7: Elaborated architecture for the Prosody component of the cycle.

prosodic process is syllabification.[5] Therefore, in this section, we only show syllabification, and the other aspects of the Prosody stage are introduced later.

Generally, the syllable structure of Armenian is fairly typical. The maximal syllable is CjVCCC. In this chapter, we set aside complex codas, but see Dolatian (2020a) for a treatment of them.

Formally, syllabification is a function with a copy set of size 2. The input and output are shown for the running example in Figure 20.8 below, omitting the settings, operation list, and morphological structure. As shown, syllabification creates a single syllable over the stem *kʰir*.

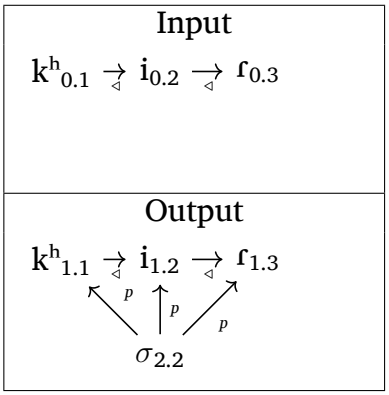


Figure 20.8: Prosody: Syllabifying the stem *kʰir*.

Strother-Garcia (2019) examined different ways syllable structure and syllabification can be formalized using model-theoretic representations and logic. In particular, she used a single binary dominance relation, and included domain elements that satisfied unary relations for syllabic roles such as onset, nucleus, and coda (see also Chapter 15). We formalize a different approach. Specifically, the syllable prosodically dominates the segments via different types of prosodic dominance relations which are specialized for the type of syllable position: `PDom:syll_nuc`, `PDom:syll_ons`, and `PDom:syll_coda`. In our graphs such as in Figure 20.8, these relations are shown as *p*-labeled edges.

[5] As said in footnote (2), Dolatian (2020a) has MStems mapped to prosodic stems (PStem). For *kʰər-itʃ-e*, a PStem is created and modified in Cycles 1 and 2. We omit the PStem for brevity.

The first step in syllabification is to faithfully output all labels and relations in Copy 1 (not shown). There will be an exception to this to account for resyllabificaton, which will be discussed later in §20.5.2. Following this, syllabification applies only to segments which are unsyllabified in the input. The predicate `unsyll_0` in (20.19) checks if some segment is unsyllabified in the input. (Recall that we use 0 to denote the input 'copy.') We likewise define the predicate `unsyll_1` in (20.20) to check if a segment is unsyllabified over the output in Copy 1. The use of these two separate predicates is to simplify resyllabification in later cycles (§20.5.2). They both check if a segment x lacks a prosodic dominance relation to some syllable y.

$$\texttt{unsyll\_0}\,(x) \stackrel{\text{def}}{=} \neg\exists y[\texttt{syll}(y) \wedge [\texttt{PDom:syll\_ons}(y,x) \vee \texttt{PDom:syll\_nuc}(y,x) \vee \texttt{PDom:syll\_coda}(y,x)]] \quad (20.19)$$

$$\texttt{unsyll\_1}\,(x) \stackrel{\text{def}}{=} \neg\exists y[\phi^{1}_{\texttt{syll}}(y) \wedge [\phi^{1,1}_{\texttt{PDom:syll\_ons}}(y,x) \vee \phi^{1,1}_{\texttt{PDom:syll\_nuc}}(y,x) \vee \phi^{1,1}_{\texttt{PDom:syll\_coda}}(y,x)]] \quad (20.20)$$

These predicates will faciltate writing how the prosodic structure in the output is generated. For every position $1.x$ that is an unsyllabified vowel in Copy 1, a syllable node is created in Copy 2 (20.21). Any unsyllabified vowel at position $1.x$ will stand in the `PDom:syll_nuc` relation to the newly created syllable $2.x$ (20.22). As for onset formation (20.23), any unsyllabified consonant $1.y$ which precedes some vowel $1.z$ in Copy 1 becomes related to the syllable $2.z$. Similarly for coda formation (20.24), any consonant at position $1.y$ which succeeds a vowel $1.z$ in Copy 1 is related to the syllable $2.z$ provided the consonant $1.y$ is not already serving

as the onset of some syllable.

$$\phi^{2}_{\texttt{syll}}(x) \stackrel{\text{def}}{=} \phi^{1}_{\texttt{vowel}}(x) \wedge \texttt{unsyll\_1}\,(x) \tag{20.21}$$

$$\phi^{2,1}_{\texttt{PDom:syll\_nuc}}(x,y) \stackrel{\text{def}}{=} \phi^{2}_{\texttt{syll}}(x) \wedge \phi^{1}_{\texttt{vowel}}(y) \wedge \texttt{unsyll\_1}\,(y) \wedge (x = y) \tag{20.22}$$

$$\phi^{2,1}_{\texttt{PDom:syll\_ons}}(x,y) \stackrel{\text{def}}{=} \phi^{2}_{\texttt{syll}}(x) \wedge \phi^{1}_{\texttt{cons}}(y) \wedge \texttt{unsyll\_1}\,(y) \wedge \exists z[\phi^{1}_{\texttt{vowel}}(z) \wedge \phi^{1,1}_{\texttt{succ:seg}}(y,z) \wedge \phi^{2,1}_{\texttt{PDom:syll\_nuc}}(x,z)] \tag{20.23}$$

$$\phi^{2,1}_{\texttt{PDom:syll\_coda}}(x,y) \stackrel{\text{def}}{=} \phi^{2}_{\texttt{syll}}(x) \wedge \phi^{1}_{\texttt{cons}}(y) \wedge \texttt{unsyll\_1}\,(y) \wedge \exists z[\phi^{1}_{\texttt{vowel}}(z) \wedge \phi^{1,1}_{\texttt{succ:seg}}(z,y)] \wedge \phi^{2,1}_{\texttt{PDom:syll\_nuc}}(x,z)] \wedge \neg\exists u[\phi^{2}_{\texttt{syll}}(u) \wedge \phi^{2,1}_{\texttt{PDom:syll\_ons}}(u,y)] \tag{20.24}$$

Following syllabification, there is resyllabification, syllable linearization, and prosodic constituents are constructed (see Figure 20.7). For monosyllabic *kʰir*, these steps are vacuous. We explain these steps in detail in later sections.

20.4.6 Phonology: Stem-level rule of stress

After Prosody, the next stage is the Phonology. There are two phonological processes in this component: stress assignment and reduction. For the stem *kʰir*, the stem-level rule of stress applies. Reduction does not apply because there are no destressed vowels. Reduction is formalized in Cycle 2 (§20.5). The input and output are shown below in Figure 20.9. The stressed vowel is in bold.

Stress is a transduction with a copy set of size 1. Morphologically, stress is triggered if we are in the stem-level or word-level domains. The predicate (20.25) checks if we are in the right cophonology by examining the SETTINGS. We use this predicate as a condition on the application of stress.

$$\texttt{StrDom} \stackrel{\text{def}}{=} \texttt{Cophon:SLevel}(\textsc{settings}) \vee \texttt{Cophon:WLevel}(\textsc{settings}) \tag{20.25}$$

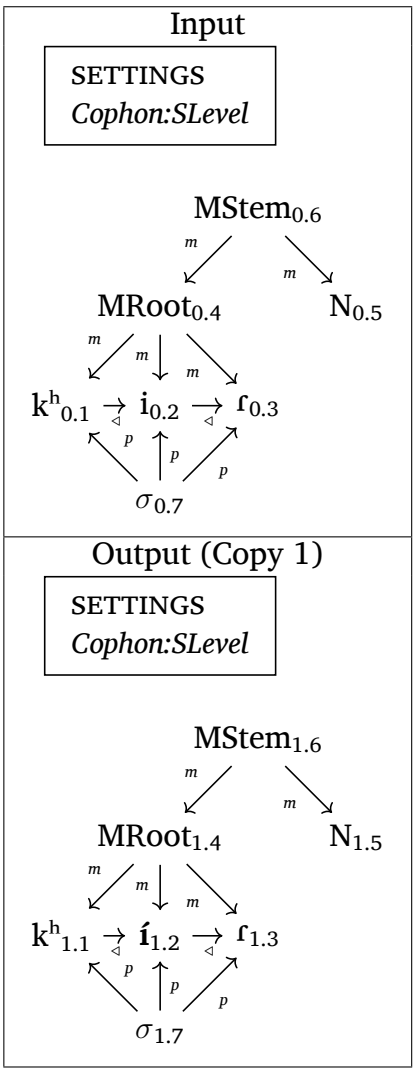


Figure 20.9: Phonology: Applying stem-level stress assignment in k^hir.

As said in §20.2.2, Armenian places stress on the rightmost full vowel. This is either the final syllable if it's a non-schwa; otherwise the penultimate syllable. We use the following predicates to find these phonological contexts, i.e., full vowels, final syllables, and penultimate syllables which precede a schwa-headed syllable.[6]

$$\texttt{vowel:full}\,(x) \stackrel{\text{def}}{=} \texttt{vowel}(x) \wedge \neg\texttt{schwa}(x) \quad (20.26)$$

$$\texttt{syll:final}\,(x) \stackrel{\text{def}}{=} \texttt{syll}(x) \wedge \neg\exists y[\texttt{succ:syll}(x,y)] \quad (20.27)$$

$$\texttt{syll:schwa}\,(x) \stackrel{\text{def}}{=} \texttt{syll}(x) \wedge \exists y[\texttt{schwa}(y) \wedge \texttt{PDom:syll\_nuc}(x,y)] \quad (20.28)$$

$$\texttt{syll:pen-pre-schwa}\,(x) \stackrel{\text{def}}{=} \texttt{syll}(x) \wedge \exists y[\texttt{succ:syll}(x,y) \wedge \texttt{syll:final}\,(y) \wedge \texttt{syll:schwa}\,(y)] \quad (20.29)$$

With these predicates, we now apply stress. We faithfully output all labels and relations except for those which involve stress (not shown). Then we find the rightmost full vowel (20.30), i.e., a vowel x which (1) is full, (2) part of the syllable y which is (3) either the final syllable or precedes a schwa-headed syllable. The position x is stressed by getting the label $\texttt{stressed}(x)$ (20.31). Crucially, the output function (20.31) checks that we are in the right morphologically-induced stress domain. This transduction is however incomplete. In Cycle 2 (§20.5.3), we add an output function to mark destressed vowels.

[6]Some of these predicates check for the linear order of syllables via the binary relation of syllable-based immediate successor: $\texttt{succ:syll}(x,y)$, which we will discuss later in Cycle 2 (§20.5.2).

$$\texttt{vowel:rightFull}(x) \stackrel{\text{def}}{=} \texttt{vowel:full}(x) \wedge \exists y[(\texttt{syll:final}(y) \vee \texttt{syll:pen-pre-schwa}(y)) \wedge \texttt{PDom:syll\_nuc}(y,x)] \tag{20.30}$$

$$\phi^1_{\texttt{stressed}}(x) \stackrel{\text{def}}{=} \texttt{StrDom} \wedge \texttt{vowel:rightFull}(x) \tag{20.31}$$

Other aspects of the co-phonologies are treated in later sections.

This completes the first stem-level cycle for the free-standing noun *kʰír*. The output of this cycle is next fed back to the cyclic architecture in order to start a new cycle of morphology and phonology.

20.5 Second cycle: Generating a derivative

The first cycle showcased the basis of our logical formalization. In the second cycle, we provide further details on those logical transductions to account for the creation of overt morphological structure (§20.5.1), resyllabification (§20.5.2), and the application of cyclical phonological rules (§20.5.3).

20.5.1 Operation and Morphology: Adding a derivational suffix

Given the output of the first cycle as input to Cycle 2, the transductions illustrated in Figure 20.2 are repeated, beginning with the Operation component, articulated in §20.4.2. This transduction shifts SETTINGS to point to the subsequent operation. In the running example, it now points to the operation labeled Op:Der-it$\widehat{\text{ʃ}}(x)$, which means this cycle will add the derivational suffix *-itʃ*. The input and output for the Operation component of cycle 2 is shown in Figure 20.10.

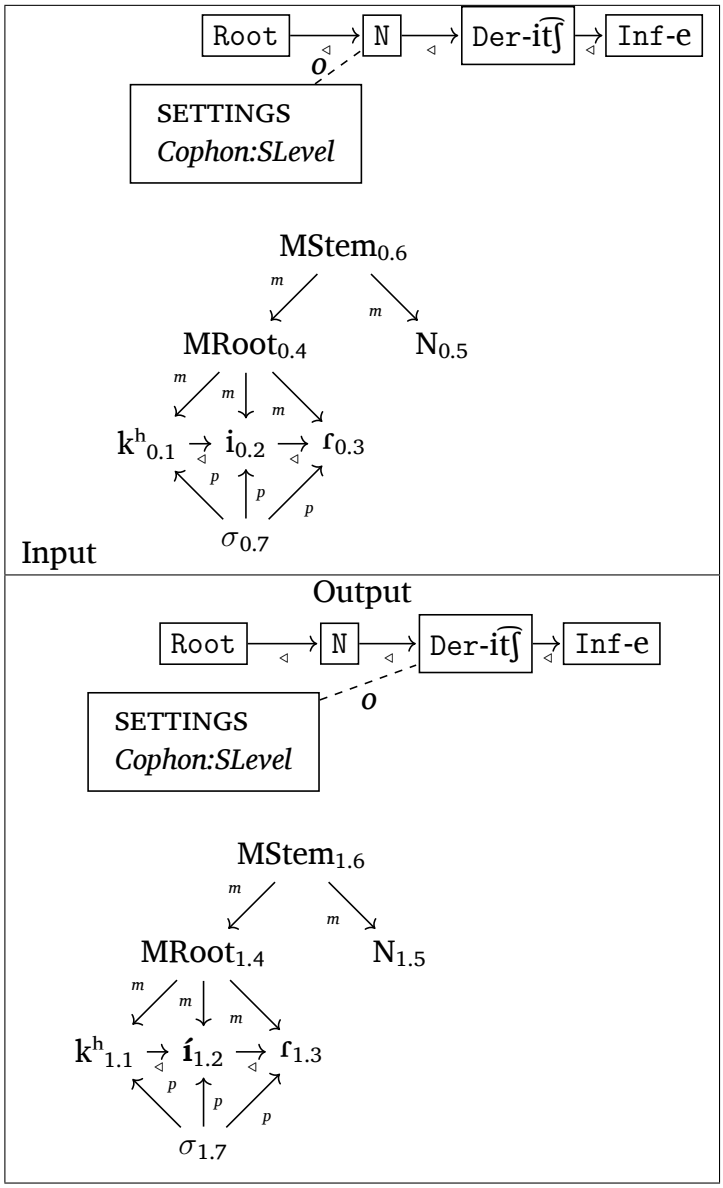


Figure 20.10: Operation: The input and output of the operation component of cycle 2 for the stem *kʰir*.

The output of the Operation component is fed to the Morphology component. While the Morphology component in Cycle 1 added a covert affix -∅, the Morphology component in Cycle 2 adds an overt derivational suffix

-itʃ. The input and output of the Morphology component in Cycle 2 is shown in Figure 20.11.

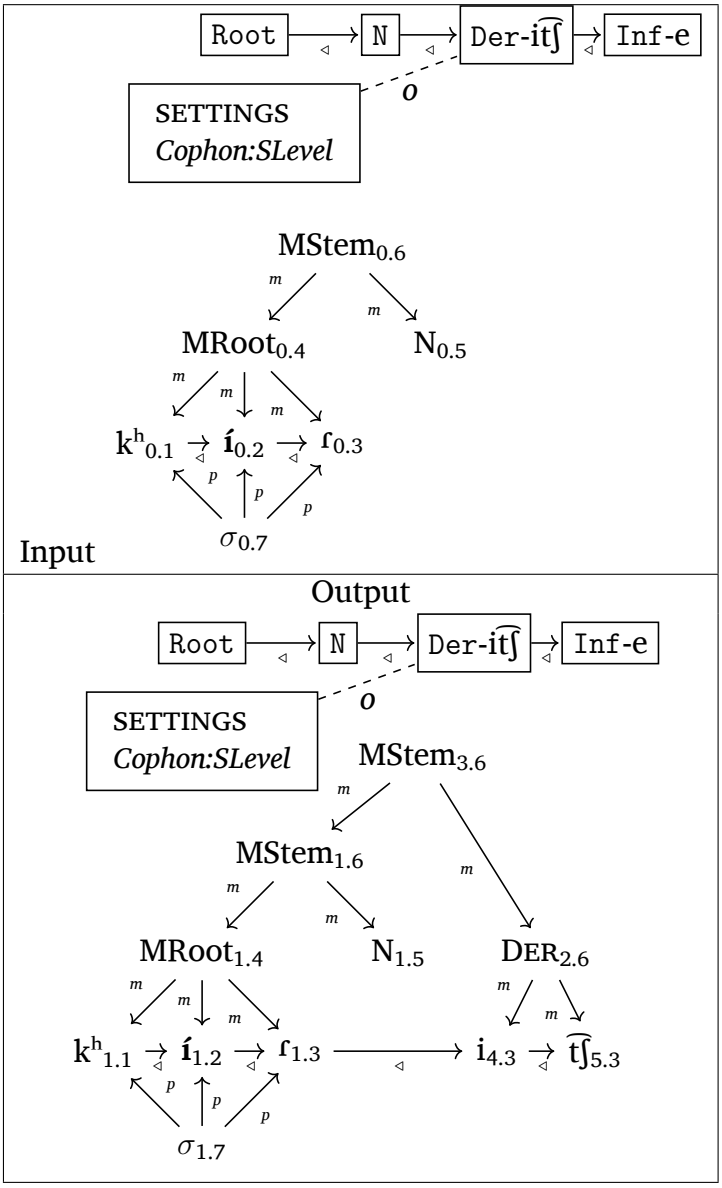


Figure 20.11: Input and output structures for the addition of the derivational suffix *-itʃ*.

In the first cycle, we noted that the size of the copy set for the logical

transduction would be equal to $n + 3$ where n was the size of the largest derivational or inflectional affix. Since the derivational suffix *-itʃ* is of size 2, the part of the logical transduction that addresses its addition will use a copy set up to and including 5. As in the case of zero-affixation, copies 2 and 3 are reserved for positions in the morphological structure indicating the affix and the new stem. Copies 4 and 5 will be used to represent the phonetic content of the derivational suffix *-itʃ*.

The current transduction utilizes the predicate `CrntOp : Der-`itʃ , which is true only if there is an instruction to add the derivational suffix *-itʃ* (see §20.4.3). This predicate is used to output the base faithfully (not shown). As explained in §20.4.3, equations 20.9 and 20.11 on page 338 place the suffix node and MStem node in copies 2 and 3 as output correspondents of the morphologically-topmost node in the input structure (the node $MStem_{0.6}$) in Figure 20.11). Similarly, the formulas which establish the dominance relations (Equations 20.13 and 20.14 on page 339) are also at work here.

Since the suffix *itʃ* has length 2, the next two copies 4 and 5 are used to represent this suffix's phonetic content. Specifically, the phonetic content of the suffix *itʃ* is defined to be output correspondents of the input-final segment (20.33 and 20.34). In Figure 20.11, this is the node indexed 0.3. This segment is picked out by the predicate `final` (x) defined below. The suffix morpheme dominates these suffix segments via morphological dominance in the output structure (20.35 and 20.36). The suffix segments are connected via successor (20.37), and connected with the base via successor (20.38).

$$\texttt{final}\ (x) \stackrel{\text{def}}{=} \neg\exists y[\texttt{succ:seg}(x,y) \quad (20.32)$$

$$\phi^4_{\texttt{i}}(x) \stackrel{\text{def}}{=} \texttt{final}\ (x) \wedge \texttt{CrntOp} : \texttt{Der-it}\widehat{\int} \quad (20.33)$$

$$\phi^5_{\texttt{t}}(x) \stackrel{\text{def}}{=} \texttt{final}\ (x) \wedge \texttt{CrntOp} : \texttt{Der-it}\widehat{\int} \quad (20.34)$$

$$\phi^{2,4}_{\texttt{MDom}}(x,y) \stackrel{\text{def}}{=} \texttt{MTop}\ (x) \wedge \texttt{final}\ (y) \wedge \texttt{CrntOp} : \texttt{Der-it}\widehat{\int} \quad (20.35)$$

$$\phi^{2,5}_{\texttt{MDom}}(x,y) \stackrel{\text{def}}{=} \texttt{MTop}\ (x) \wedge \texttt{final}\ (y) \wedge \texttt{CrntOp} : \texttt{Der-it}\widehat{\int} \quad (20.36)$$

$$\phi^{4,5}_{\texttt{succ:seg}}(x,y) \stackrel{\text{def}}{=} \texttt{final}\ (x) \wedge \texttt{final}\ (y) \wedge \texttt{CrntOp} : \texttt{Der-it}\widehat{\int} \quad (20.37)$$

$$\phi^{1,4}_{\texttt{succ:seg}}(x,y) \stackrel{\text{def}}{=} \texttt{final}\ (x) \wedge \texttt{final}\ (y) \wedge \texttt{CrntOp} : \texttt{Der-it}\widehat{\int} \quad (20.38)$$

As before, any other unary and binary relation which refer to a copy $c > 1$ that have not been specified are set to `false` as described in Equations 20.15 and 20.16.

In sum, overt affixation further specifies aspects of the transduction of the morphological component specified in §20.4.3 to generate and linearize the segmental make-up of the overt affix and place them in a dominance relation with morphological nodes. In the case of the suffix *it͡ʃ*, because it is of length 2, we used copies 4 and 5 for this purpose. Affixes of size 3 would also make use of copy 6. Thus the size of the copy set for the transduction for the morphological compenent is $n + 3$ where n is the size of the largest affix.

While the above formulas work for the particular derivational suffix *-it͡ʃ*, they will need to be generalized to be sensitive to other morphemes as well to accurately specify the transduction we have in mind. To explain, recall that M denotes the finite set of morphological derivational and inflectional operations. If there is another suffix $\texttt{M} \in M$ that begins with the vowel [i] then the equation $\phi^4_{\texttt{i}}(x)$ would have to be defined as $\texttt{final}\ (x) \wedge$ $\big(\ \texttt{CrntOp} : \texttt{Der-it}\widehat{\int}\ \vee\ \texttt{CrntOp} : \texttt{M}\ \big)$. More generally, we have to partition M according to the segmental makeup of the morphemes. If $M1[i]$ denotes the suffixes that begin with [i] and if $M2[\widehat{\text{tʃ}}]$ denotes the suffixes whose second segment is [t͡ʃ], then formulas 20.33 and 20.34 above would be

replaced with formulas 20.39 and 20.40 shown below, respectively.

$$\phi^4_{\texttt{i}}(x) \stackrel{\text{def}}{=} \quad \texttt{final}\,(x) \wedge \bigvee_{\texttt{M}\in M1[i]} \texttt{CrntOp:M} \tag{20.39}$$

$$\phi^5_{\widehat{\texttt{tʃ}}}(x) \stackrel{\text{def}}{=} \quad \texttt{final}\,(x) \wedge \bigvee_{\texttt{M}\in M1[\widehat{\texttt{tʃ}}]} \texttt{CrntOp:M} \tag{20.40}$$

(20.41)

Likewise, the dominance relations expressed in formulas 20.35 and 20.36 have to be generalized because there are potentially other morphemes in the lexicon that have phonetic content and are of length 1 or of length 2. We again partition M into morphemes of length 1 and 2 into the sets $Mlen1$ and $Mlen2$, respectively. Then formulas 20.35 and 20.36 above would be replaced with Equations 20.42 and 20.43 below.

$$\phi^{2,4}_{\texttt{MDom}}(x,y) \stackrel{\text{def}}{=} \quad \texttt{MTop}\,(x) \wedge \texttt{final}\,(y) \wedge \bigvee_{\texttt{M}\in Mlen1} \texttt{CrntOp:M} \tag{20.42}$$

$$\phi^{2,5}_{\texttt{MDom}}(x,y) \stackrel{\text{def}}{=} \quad \texttt{MTop}\,(x) \wedge \texttt{final}\,(y) \wedge \bigvee_{\texttt{M}\in Mlen2} \texttt{CrntOp:M} \tag{20.43}$$

(20.44)

Similar generalizationse should be made for the formulas defining successor relations as well.

20.5.2 Examination and Prosody: Stem-levels and resyllabification

The Examination component applies after the Morphology component. The new topmost morphological node is the suffix MStem. Just like in Cycle 1, the SETTINGS gets the stem-level cophonology label via percolation from the topmost MStem node. The transduction presented in §20.4.4 needs no further specification, and the change itself is vacuous, and so we move on to the next component, Prosody.

As outlined in §20.4.5, the the Prosody component will syllabify the suffix, and resyllabify the base's coda. Figure 20.12 shows the input and output structures for this process. This figure only shows the segments and syllables and leaves out the other parts of the input and output structures.

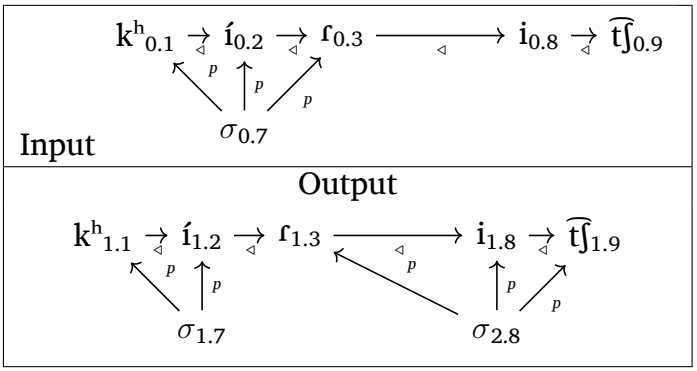


Figure 20.12: Applying resyllabification to the derived stem *kʰɪ́r-itʃ*.

In §20.4.5, copy 1 contained a faithful copy of all the input relations, both segemental, and prosodic. For example, labels are carried over to copy 1 because of formulas such as $\phi^1_{\texttt{seg}}(x) \stackrel{\text{def}}{=} \texttt{seg}(x)$ and $\phi^1_{\texttt{syll}}(x) \stackrel{\text{def}}{=} \texttt{syll}(x)$. Similarly, binary relations that hold in the input structure are also carried over with formulas expressing faithfulness like the ones shown below.

$$\phi^{1,1}_{\texttt{PDom:syll\_nuc}}(x,y) \stackrel{\text{def}}{=} \qquad \texttt{PDom:syll\_nuc}(x,y) \tag{20.45}$$

$$\phi^{1,1}_{\texttt{PDom:syll\_ons}}(x,y) \stackrel{\text{def}}{=} \qquad \texttt{PDom:syll\_ons}(x,y) \tag{20.46}$$

In §20.4.5, however, we noted there was an exception to copy 1 being a completely faithful copy to the input structure in order to account for resyllabification. This exception has to do with codas, so that they can be resyllabified as onsets. The exception effectively 'unsyllabifies' segments preceding vowels which were previously identifed as codas. Equation 20.47 says a segment which is a coda in the input structure will remains a coda in copy 1 *unless* it precedes a vowel.[7]

$$\phi^{1,1}_{\texttt{PDom:syll\_coda}}(x,y) \stackrel{\text{def}}{=} \qquad \texttt{PDom:syll\_coda}(x,y) \wedge \neg\exists z[\texttt{vowel}(z) \wedge \texttt{succ:seg}(y,z)] \tag{20.47}$$

As before in §20.4.5, copy 2 is used to introduce new syllable nodes for unsyllabified material. This part of the logical transduction is the same as

[7]Dolatian (2020a) provides additional conditions for resyllabification, such as when new consonants precede or follow the base's root, when new vowels precede the root, or in compound prosody. We set these aside.

what was presented in §20.4.5. The same formulas which syllabified the root in the cycle 1 will also successfully syllabify the overt affix here. This is because the formulas for copy 2 elements are defined in terms of the structure present in Copy 1. In particular, in the running example, because the node labeled [ɾ] indexed at 1.3 is no longer syllabified, Equations 20.21 to 20.24 will ensure that it will be syllabified as an onset.

After (re)syllabification, the old and new syllables are ordered via a the successor relation among syllables: $\texttt{succ:syll}(x, y)$. Syllable linearization is a transduction that uses a copy set of size 1. We show the input and output below.

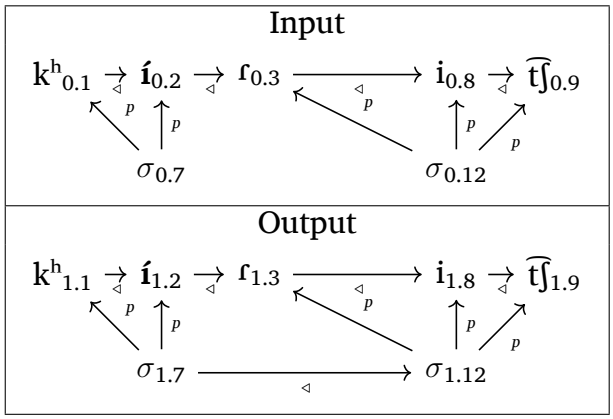


Figure 20.13: Prosody: Ordering syllables in a derived stem *kʰíɾ-it͡ʃ*

Syllables are ordered based on the general precedence relations of their nuclei. We first generalize the immediate successor relation ($\lhd$) of segments into general long-distance precedence ($<$). Chapter 2 explained how this can be accomplished with MSO logic in Equation 2.14 on page 47. With general precedence we can define the next-vowel relation (Equation 20.48) which related successive vowels on the vowel tier. Syllables can then be ordered based on the location of their nuclei in the vowel tier

(Equation 20.49).

$$\lhd_{\texttt{V}}(x,y) \stackrel{\text{def}}{=} \quad \texttt{vowel}(x) \wedge \texttt{vowel}(y) \wedge x < y \wedge \neg\exists z[\texttt{vowel}(z) \wedge x < z < y] \tag{20.48}$$

$$\phi^{1,1}_{\texttt{succ:syll}}(x,y) \stackrel{\text{def}}{=} \quad \texttt{syll}(x) \wedge \texttt{syll}(y) \wedge \exists u,v\,[u \lhd_{\texttt{v}} v \wedge \texttt{PDom:syll\_nuc}(x,u) \wedge \texttt{PDom:syll\_nuc}(y,v)] \tag{20.49}$$

As mentioned, general precedence requires MSO logic by using quantifiers over sets. Alternatively as discussed in Chapter 2, general precedence could be set as a primitive relation in our input and the successor relations could be derived with only First Order (FO) logic.[8] Having formalized resyllabification and syllable ordering, we now turn to cyclic phonological rules.

20.5.3 Phonology: Cyclic reduction

The Phonology component follows the Prosody component. As with Cycle 1, stem-level cophonology applies. For Cycle 2, both stem-level rules, stress assignment and reduction, apply.

The Phonology component consists of two logical transductions which apply in a sequence: stress assignment and reduction. The tranduction for stress assignment was presented in §20.4.6, and it applies here again, placing stress on the suffix vowel *-ítʃ*. We introduce an atomic relation `destressed` and include its specification as part of the stress assignment function. Then the transduction for reduction transforms destressed vowels into schwas.

Figure 20.14 shows the segmental part of the input and output structures for the reduction process in the Phonology component, as applied to the running example.

In addition to assigning stress as described in §20.4.6, we further specify stress assignment to also indicate which vowels are destressed. Equation 20.50 says that a position x satisfies the atomic formula `destressed`

[8] Because syllable sizes are bound, it is possible that syllable linearization may be achievable with quantifier-free logic. See Chapter 22 for Quantifer-Free logic and Strother-Garcia (2018a) and Strother-Garcia (2019) for application to syllabification.

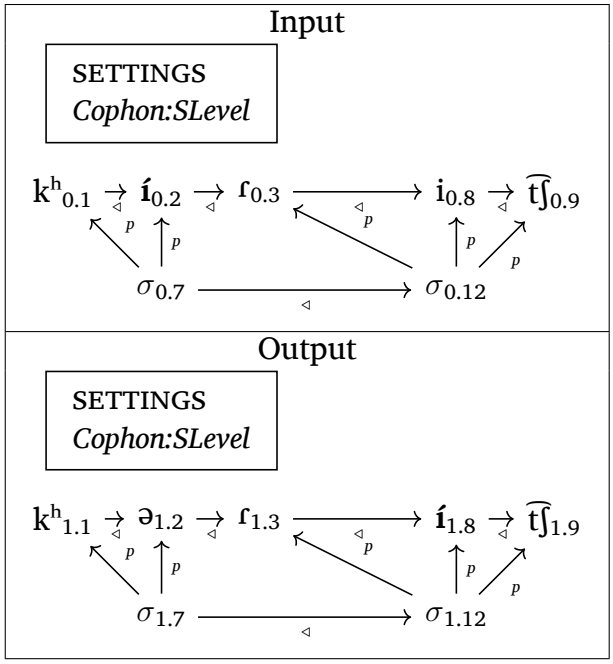


Figure 20.14: Phonology: Applying stress assignment and reduction to the derived stem *kʰɨ̌ɾ-ítʃ*.

in the output if and only if x is a stressed vowel in the input but not the output.

$$\phi_{\texttt{destressed}}(x) \stackrel{\text{def}}{=} \qquad \texttt{stressed}(x) \wedge \neg\phi_{\texttt{stressed}}(x) \tag{20.50}$$

No position would satisfy this formula in cycle 1 because no input would include any stressed vowels.

Following the application of stress assignment, vowel reduction occurs. Reduction is modeled with a separate logical transduction that uses a copy set of size 1. Figure 20.15 shows the input and output structures to reduction for the running example. Figure 20.15 only shows the segments, syllable nodes, and the SETTINGS constant. In the input structure in Figure 20.15, the destressed and stressed vowels are shown in bold.

Input	$\mathrm{k^h}_{0.1} \to \mathbf{ĭ}_{0.2} \to \mathrm{ɾ}_{0.3} \longrightarrow \mathbf{í}_{0.8} \to \widehat{\mathrm{tʃ}}_{0.9}$
Output	$\mathrm{k^h}_{1.1} \to \mathrm{ə}_{1.2} \to \mathrm{ɾ}_{1.3} \longrightarrow \mathrm{í}_{1.8} \to \widehat{\mathrm{tʃ}}_{1.9}$

Figure 20.15: Phonology: Applying vowel reduction to the input structure *kʰĭr-ít͡ʃ* to produce the output *kʰər-ít͡ʃ*.

As explained in §20.2.2, destressed high vowels are either deleted or reduced to a schwa. The transduction first checks that we are in the right morphological domain of reduction (20.51), i.e., that we are in the stem-level domain based on the features of the SETTINGS constant. With this domain condition, we turn destressed high vowels into schwas (20.52). Any underlying schwas are faithfully outputted.

$$\texttt{ReducDom} \stackrel{\text{def}}{=} \qquad \texttt{Cophon:SLevel}(\textsc{settings}) \tag{20.51}$$

$$\phi_{\texttt{schwa}}(x) \stackrel{\text{def}}{=} \qquad \texttt{ReducDom} \wedge [\texttt{schwa}(x) \vee [\texttt{destressed}(x) \wedge \texttt{high}(x)]] \tag{20.52}$$

(20.53)

There are some additional finer points regarding reduction which are set aside here. Readers are referred to Dolatian (2020a) for a complete formalization. Nonetheless, the reduction transduction specified here captures the main features of reduction in Armenian.

This completes the second cycle. The logical formalism exactly captures the fact that reduction is a cyclic process that is derived via multiple cycles of stress shift and reduction. In the next cycle, we show how this formalism separates the stem-level domain from the word-level domain.

20.6 Third cycle: Word-level phonology

The output of the second cycle *kʰər-ítʃ* is fed to a new third cycle to form an inflected word *kʰər-ítʃ-é*. The main aspects of this cycle are creating a different overt affix (§20.6.1), parsing prosodic words (§20.6.3), and applying a separate morphologically-induced cophonology (§20.6.4).

20.6.1 Operation and Morphology: Adding an inflectional suffix

In the third cycle, the Operation and Morphology components do the tasks of generating an overt inflectional affix. The operation component advances the operation list so that the SETTINGS points to the final operation node: `Op`:`Inf`-e. This transduction uses the same functions from Cycle 1 (§20.4.2). Then the Morphology component adds the inflectional suffix *-e*. Figure 20.16 shows the input and output structures for the Morphology component.

As in previous Morphology stages, the predicate `CrntOp : Inf-e` checks if the current operation node indicates that an inflectional suffix *-e* should be added. With this condition, the transduction generates the inflectional suffix *-e*. Following the convention established in the first two cycles, copies 2 and 3 are used to for the affix and MWord nodes in the morphological structure (Equations 20.9 and 20.12). The phonetic material *-e* will be placed in copy 4. Following the discussion at the end of §20.5.1, let $M1[e]$ be the subset of morphemes which begin with [e]. Equation 20.54 ensures that t

$$\phi^4_{\mathsf{e}}(x) \stackrel{\text{def}}{=} \qquad \texttt{final}\,(x) \wedge \bigvee_{M \in M1[e]} \texttt{CrntOp : M} \tag{20.54}$$

Finally all of the aforementioned nodes are connected via successor and morphological dominance with the formulas specified in §20.5.1.

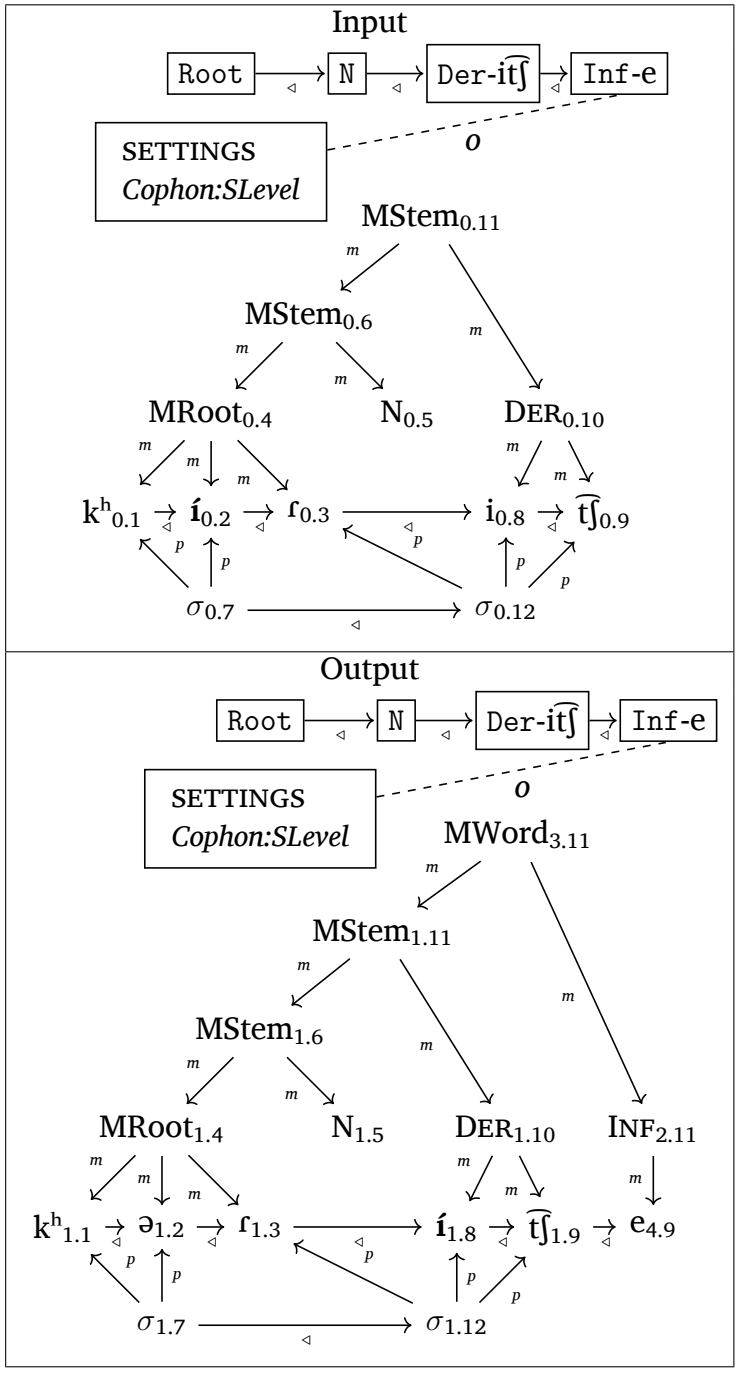


Figure 20.16: Morphology: Adding an overt inflectional suffix *kʰər-ítʃ-e*.

20.6.2 Examination: Parsing instructions

The output of the Morphology in cycle 3 is a morphological word. Unlike stems, words trigger the word-level cophonology and they are parsed into prosodic words. In order to apply these processes, in the Examination stage, the SETTINGS constant is given the word-level cophonology label. It is likewise given a parsing instruction to trigger PWord formation. Figure 20.17 shows the input and output structures of the Examination component as applied to the running example.

The label for the word-level cophonology is percolated via the output function in (20.55). As for the parse instructions (20.56), this label is generated on the SETTINGS if the morphologically topmost node is an MWord which is not already parsed into a PWord. To encode this property, we use the binary relation of $\texttt{Match:word}(x, y)$ which associates an MWord with a matching PWord. This relation is explained further in the Prosody stage (§20.6.3).

$$\phi^1_{\texttt{Cophon:WLevel}}(x) \stackrel{\text{def}}{=} x = \textsc{settings} \wedge \exists y[\ \texttt{MTop}\ (y) \wedge \texttt{Cophon:WLevel}(y)] \quad (20.55)$$

$$\phi^1_{\texttt{Parse:MWord}}(x) \stackrel{\text{def}}{=} x = \textsc{settings} \wedge \exists y[\ \texttt{MTop}\ (y) \wedge \texttt{MWord}(y) \wedge \neg\exists z[\texttt{PWord}(y) \wedge \texttt{Match:word}(y, z)]] \quad (20.56)$$

20.6.3 Prosody: Generating prosodic words

After Examination, the Prosody component applies. The Prosody component involves resyllabification and generating higher-level prosodic structure. The suffix is first resyllabified with the base, using the functions completely specified in Cycle 2 (§20.5.2), which effectively resyllabfies *.kʰə.ɾíʧ.*⟨*e*⟩ as *.kʰə.ɾí.ʧe*.

The final transduction making up the Prosody component is Prosodic Constituency (Figure 20.7). The labels on the SETTINGS indicate whether any prosodic constituents should be generated. In the running example, the SETTINGS constant has the label `Parse:MWord`, which is interpreted as an instruction to generate a PWord. In terms of the relational structure, this PWord dominates the syllable nodes. Additionally, this PWord is related to the MWord via a special binary relation $\texttt{Match:word}(x, y)$ that is specialized

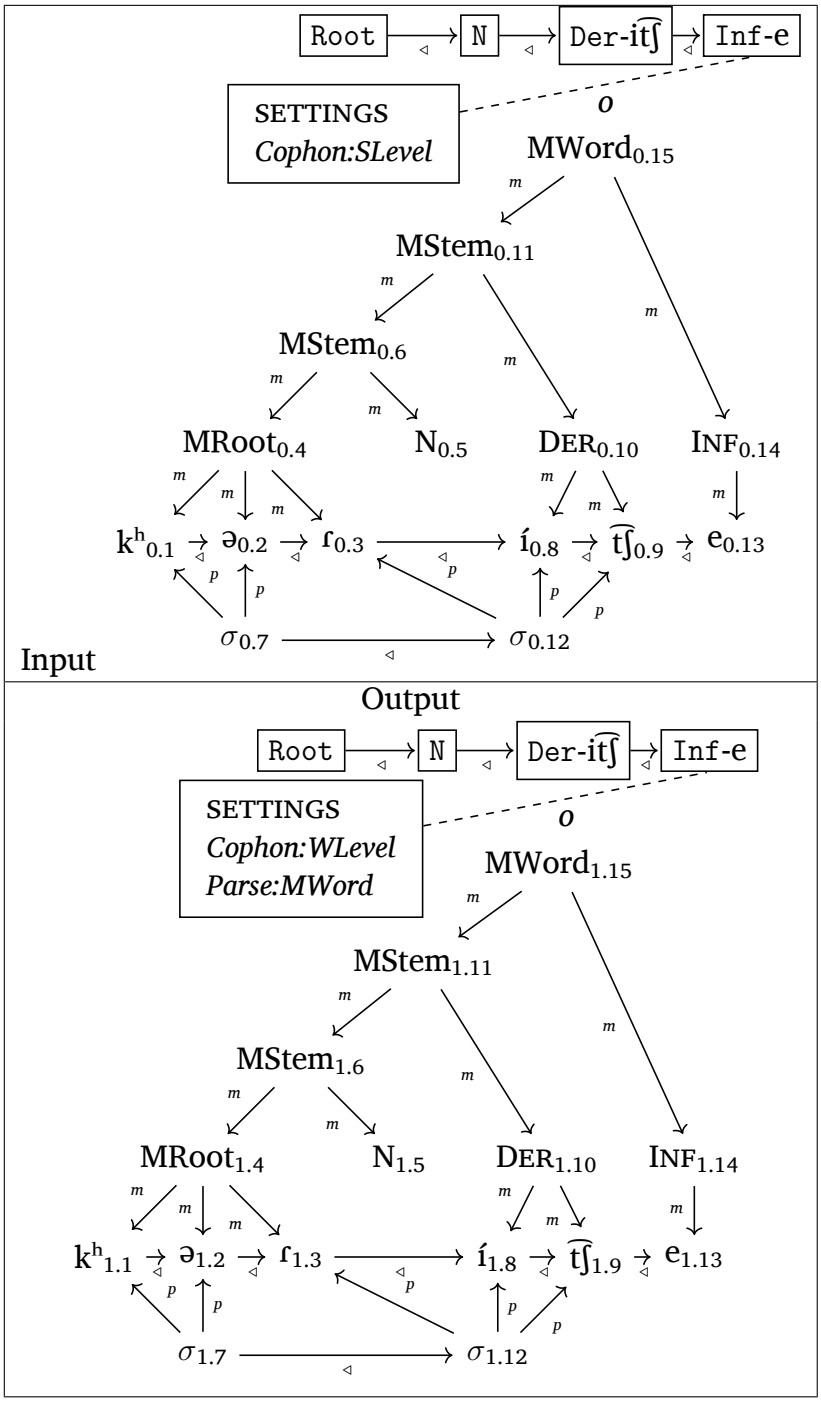


Figure 20.17: Examination: Percolating word-level cophonology and prosodic parse for *kʰər-ítʃ-e*.

for prosodic correspondences (Selkirk, 2011). Figure 20.18 shows the input and output structures of the Prosodic Constituency transduction for the running example.

The following equations generate the PWord, establish its dominance relations, and match it with the topmost MWord. The first lines of Equation 20.57 generates the PWord as output correspondent of the input's topmost MWord. The second line ensures that the input MWord is not already matched to a PWord.

$$\phi^2_{\texttt{PWord}}(x) \stackrel{\text{def}}{=} \quad \texttt{Parse:MWord}(\textsc{settings}) \wedge \texttt{MWord}(x) \tag{20.57}$$
$$\wedge\neg\exists y[\texttt{PWord}(y) \wedge \texttt{Match:word}(x,y)] \tag{20.58}$$

Equation 20.59 establishes the `Match:word` relation between the MWord and this new Pword.

$$\texttt{Match:word}^{1,2}(x,y) \qquad \phi^1_{\texttt{MWord}}(x) \wedge \phi^2_{\texttt{PWord}}(y) \tag{20.59}$$
$$\tag{20.60}$$

Finally, the PWord dominates the syllables via a type of prosodic dominance that's specialized for PWords and syllables.

$$\phi^{2,1}_{\texttt{PDom:PWord\_syll}}(x,y) \stackrel{\text{def}}{=} \quad \texttt{Parse:MWord}(\textsc{settings}) \wedge \phi^2_{\texttt{PWord}}(x) \wedge \phi^1_{\texttt{syll}}(y) \tag{20.61}$$

As before, any relation not specified here which involve a copy $c > 1$ is set to `false`. In this way, this logical transduction captures the generation of prosodic structure from morphological structure.

20.6.4 Phonology: Word-level phonology blocks reduction

After the Prosodic component, the Phonology component applies. As before, we examine the SETTINGS in order to determine what cophonology to apply. For *kʰər-ítʃ-e*, the SETTINGS has the cophonology label `Cophon:WLevel`. Thus, we should trigger the word-level cophonology of stress without reduction. Figure 20.19 shows the input structure submitted to the Phonology component and the resulting output stucture. This figure omits the morphology and operation list.

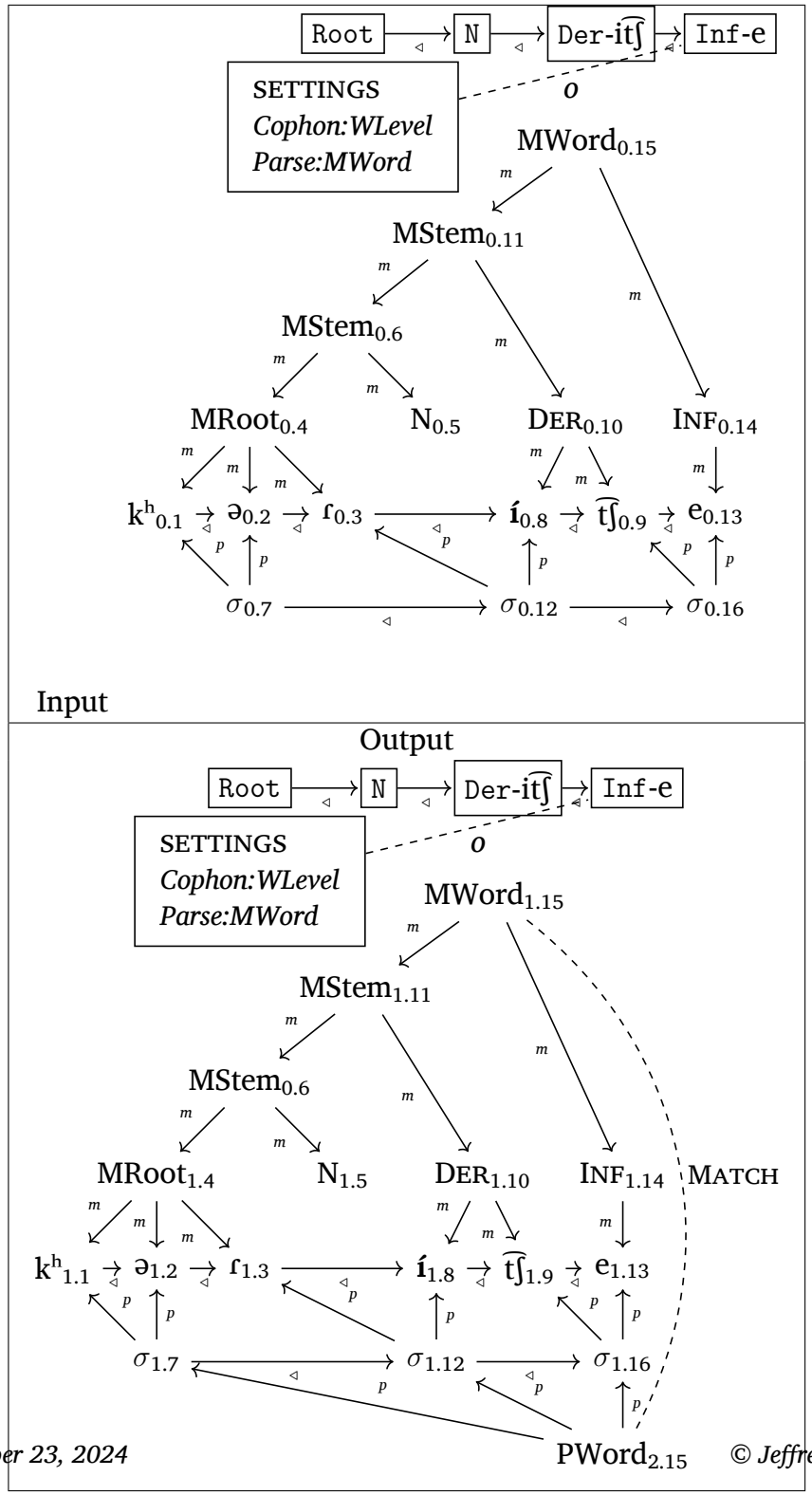


Figure 20.18: Prosody: Parsing an MWord into a PWord in *kʰər-ítʃ-e*.

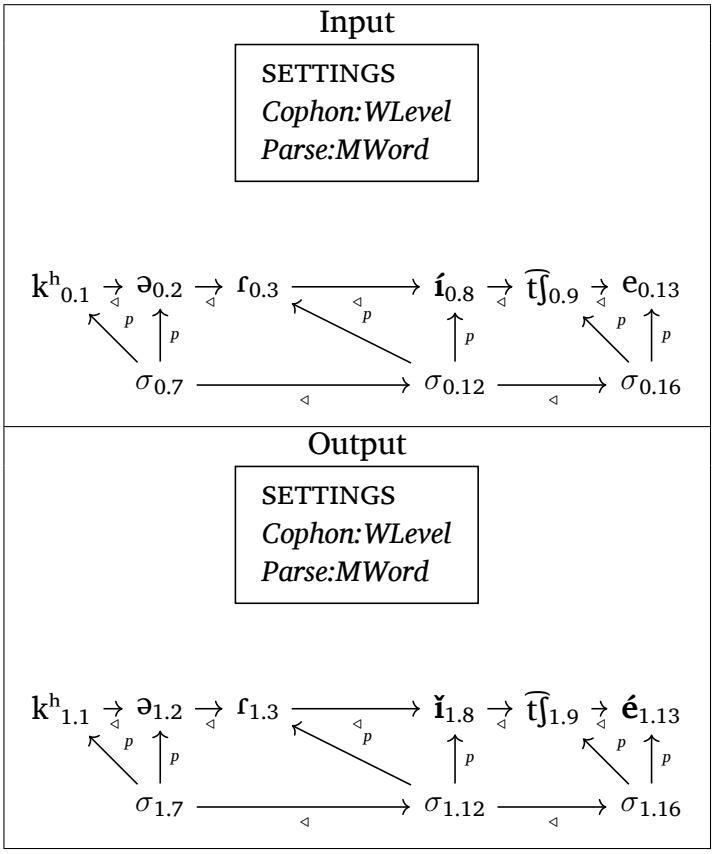


Figure 20.19: Phonology: Applying word-level phonology to an inflected word *kʰər-it͡ʃ-é*.

The same stress shift functions described in Cycle 1 (§20.4.6) shift stress onto the new suffix *-e*. However, stress reduction does not apply in this cycle because this inflectional suffix triggers the word level cophonology. These effects are obtained with the formulas previously specified. In particular, we introduced two predicates `StrDom` and `ReducDom` which must be true for stress shift and reduction to occur. Equations 20.62 and 20.63 show the definitions of these predicates. These equations repeat Equations 20.25 (page 345) and 20.51 (page 358), respectively.

$$\texttt{StrDom} \stackrel{\text{def}}{=} \texttt{Cophon:SLevel}(\textsc{settings}) \vee \texttt{Cophon:WLevel}(\textsc{settings}) \tag{20.62}$$

$$\texttt{ReducDom} \stackrel{\text{def}}{=} \texttt{Domain} : \texttt{Cophon:SLevel}(\textsc{settings}) \tag{20.63}$$

Readers can verify that functions used for stress shift (§20.4.6) all obey the cophonology condition `StrDom`. Thus they will apply for the inflected word *kʰɔr-ɪt͡ʃ-é*. In contrast, the reduction functions from Cycle 2 (§20.5.3) need to meet the condition `ReducDom`, which is not satisfied in this cycle, and this reduction is blocked.

This concludes cycle 3.

20.7 Evaluating the cyclic architecture and the computation of cyclicity

The previous sections explained how logical transductions can not only encode the input-output relationships in morphophonology, but also apply morphophonological processes in a cyclic manner. That is, the output of one round of morphophonology acts as the input to another.

Specifically, in this chapter, we presented the cycle as a series of MSO logical transductions, which apply one after another. There were five components for cyclic derivations: Operations, Morphology, Examination, Prosody, and Phonology. Within a single cycle, each component was its own logical transduction that fed the next component. To model one cycle, these five logical transductions can be combined into a single one via composition: $C = Ph \circ Pr \circ E \circ M \circ O$. Since MSO logical transductions are closed under composition, and there were a fixed number of transductions

that we employed for the cycle, this means there is a single logical transduction encompassing the entire cycle C. The application of two cycles, however, requires composing the single transduction C with itself again. And generating three cycles is obtained with $C \circ C \circ C$.

However, because there is no bound on the number of cycles in the mental grammar of Armenian (or any language), there is no bound as to how many cycles of C an arbitrary word gets. While MSO($\triangleleft$) logical transductions are closed under bounded composition, they are not closed under unbounded composition. In other words, there is not necessarily a MSO$\triangleleft$ logical transduction which captures the behavior of arbitrarily long sequences of C composed with itself. The end-result is that our formal grammar C for cyclic phonology is actually a formal grammar for computing a single cycle.

This section discusses the computational issues that arise from the absence of a bound on the number of times that a cycle may apply. In the context of the logical transducions introduced here, this amounts to no bound on the length of the operations list.

Cyclicity is a common aspect of generative phonology and morphology (Chomsky and Halle, 1968; Brame, 1974; Kiparsky, 1982). The combination of cyclicity and interactionism flourished in early work in Lexical Phonology (Kiparsky, 1982; Kaisse and Shaw, 1985), continued into Stratal OT (Trommer, 2013; Mascaró, 1976; Wiese, 1994; Kaisse and McMahon, 2011), and into contemporary phase-based approaches (Marvin, 2002; Newell, 2008; Embick, 2010; Samuels, 2011; Scheer, 2012; Guekguezian, 2017, 2021; Sande, 2017; Newell *et al.*, 2017).

But in computational linguistics, cyclicity has been a challenging problem to formalize (Sproat, 1992), with few implementations (Williams, 1993, 1994) and few learnability results (Nazarov and Pater, 2017). The same problem arguably exists in computational syntax (Levelt, 1974). There are two reasons for this difficulty: implicitness and complexity.

The first reason is that, many, if not all, cyclic formalisms utilize tools and principles that are insufficiently explicit for formal analysis. In Lexical Phonology, there have been many analyses of English phonology with a principle like the Strict Cycle Condition. In a computational formalization of English lexical phonology, Williams (1993) found various weaknesses in three contemporary theoretical models (Kiparsky, 1982; Halle and Mohanan, 1985; Booij and Rubach, 1987). Essentially, these weaknesses boiled down to various degrees of ambiguity in how certain principles or

assumptions were defined. This led to either contradictions in modelling assumptions or to errors in how the models account for aspects of the data. One major source of these difficulties lay in how models defined the Strict Cycle Condition (Kiparsky, 1993).

The second challenge for formal analysis of cyclicity is the well-known result that, in order for a phonological or morphophonological rule to be finite-state definable, the rule cannot apply to the locus of its structural change (Johnson, 1972), i.e., to the same location in the output. Otherwise, the unbounded application of a phonological rule can create non-regular languages. For example, a rule like $\varnothing \rightarrow ab / a\_b$ is not only regular, it is input strictly local if it applies simultaneously (Chandlee, 2017a; Chandlee and Heinz, 2018), and therefore as a logical transduction quantifier-free (Chapter 22). But the unbounded application of this rule results in a process which is not regular, because it effectively creates the non-regular, context-free language $\{a^n b^n \mid n \geq 1\}$.

One potential response could be to place a bound on the number of cycles (Peters and Ritchie, 1973). To our knowledge, however, this alternative has not been seriously developed because no a priori bound is clear.

To sum up, it is this unbounded application which causes a blow-up in expressivity. Unbounded rule application can simulate a Turing machine (Coleman, 1995, 1998, 77ff) or be computationally undecidable (Ristad, 1990). Kaplan and Kay (1994, 365) put it nicely as: "In the worst case, in fact, we know that the computations of an arbitrary Turing machine can be simulated by a rewriting grammar with unrestricted rule reapplication." Cyclic phonological rules are *a priori* supposed to apply to their own output, and thus we appear to reach an impasse.

This issue of unboundedness can be expressed succinctly as follows: It is not the case that there exists a number k such that for all words w, we can apply at most k cycles. In response to this problem, we sketch out three possible approaches: RUN-TIME APPROXIMATION, GIVE UP REGULARITY, and RESTRICT CYCLICITY.

The core issue of unboundedness expressed above is highlighted by the notation $(\exists k, \forall w)$, which emphasizes the order of quantification. When this order is flipped, notated $(\forall w, \exists k)$, a true statement is obtained: for every word w, there exists a number k of cycles which is *specific* to that word w, such that exactly k cycles occur. So for a given word w, there is a bound k in *run-time* or in *practice*. This approach is what we call RUN-TIME APPROXIMATION. It is essentially what the logical formalization in this

chapter is doing, because the operation list is of finite size and controls how many cycles to generate.

Given w, this approach allows us to use quite expressive logical structures such as in this chapter. One could also use this approach to define a finite-state grammar that can generate words with up to k cycles.[9] This approach prioritizes practical considerations over theoretical ones to side-step the question of the generative capacity of morphophonology. A related theoretical position one could take however would be to factor the morphophonological grammar into a regular morphophonological component and a control structure which allows the regular component to be called unboundedly many times. This is essentially the approach taken in Williams (1993) who uses a looping mechanism over the morphology and phonology. This practical approach recognizes the fact that phonological rules are regular per application (or up to k cycles) but that the generative capacity of the whole system can in principle be beyond regular.

The Run-time Approximation approach works as a grammar for an individual cycle, but not as a grammar for cyclic phonology as a whole. Given an input root r which includes its operation list o, an independent mechanism has to apply the function F as many times as needed so that we 'traverse' the operation list o from start to end and generate the desired final output word w. The end-result is that we need both a formal grammar (our logical transductions) as well as another apparatus that will actively apply the formal grammar as many times as needed.

There is another approach that we term GIVE UP REGULARITY. Some formalizations of lexical phonology abandoned cyclicity and used context-free grammars (Cole, 1990, 1995b; Coleman, 1995; Cole and Coleman, 1992). This approach effectively treats the morphophonological module of grammar as non-regular. This is a valid approach, but since we have little to say about it, we move on to the last approach.

The third approach is the most appealing to us: RESTRICT CYCLICITY. This approach aims to identify restrictions on cyclic morphophonological rules so that the transitive closure of their composition remains regular. Such restrictions include the one that Johnson and Kaplan and Kay (JKK) recognized: phonological rules cannot apply to the locus of their structural

[9]This is similar to some treatments of total reduplication (Walther, 2000; Beesley and Karttunen, 2003) which push the work of total reduplication into a run-time process. This treatment is required if we try to formalize reduplication with 1-way FSTs, but not with 2-way FSTs (cf. for a review, see Dolatian and Heinz, 2020).

change. This approach generates two questions: one empirical, and one formal.

The empirical question is whether *all* attested morphophonological rules obey the JKK restriction. In other words, are there cyclic morphophonological processes like $\varnothing \rightarrow ab / a\_b$ whose transitive closure is non-regular? Research around this question has consistently shown no. Although unbounded suffixation (a regular language) exists, there are no cases of unbounded circumfixation (a non-regular $a^n b^n$ language) (Aksënova *et al.*, 2016). Although the semantics of morphological structures is non-regular because of center-embedding in tree structures (Langendoen, 1981; Carden, 1983; Oseki, 2018; Oseki *et al.*, 2019; Oseki and Marantz, 2020), their surface morphotactics (Hammond, 1993) and cyclic phonology are regular (Bjorkman and Dunbar, 2016). The marked absence of non-regular morphotactics has been taken as evidence that, even if morphosemantics can in principle be non-regular, it is filtered through a finite-state morphophonological processing system (Hammond, 1993).

It thus seems that that restriction on regularity is empirically robust. This leads to the formal question on how we can translate the formalization of the JKK restriction from string-to-string relations to logical transductions over nonlinear representations. Answering this question appears to us to be the most appealing at present, and we encourage future work in this area.

20.8 Conclusion

This chapter showcased how to construct a formal grammar for the morphophonology of Armenian using logical transduction over a representation that included phonological, morphological, and prosodic components. In this way, we presented an alternative formalization to the cycle to previous finite-state methods, which were limited to string-based representations of the phonology, morphology, and prosody The logical formalism was capable of representing a wide array of morphophonological phenomena that are couched within cyclic phonology. By using an explicit system, we are able to appreciate the complexity of morphophonological processes and the logical formulas help express the different types of information and how they determine the morphological structure of complex words. Although we did not discuss the generative capacity of such processes, and

made liberal use of MSO($\triangleleft$) logic, Dolatian (2020a) found that the bulk of morphophonological processes are individually local functions.

A larger question is: *Is cyclicity as a larger phenomenon still computationally un-definable and too expressive?* While one of our goals was to answer this question, we essentially reached the same end-result that Williams (1993) had in her earlier formalization of cyclicity. The logical transduction here models individual cycles C, but an independent apparatus is needed to apply it as many times as needed to an input. Nonetheless, we hope that this formalization helps tease apart these moving parts, so that future work can be guided to either a) discovering the formal nature of this independent apparatus, or b) discovering restrictions on the attested typology of cyclic processes which ensures that their unbunded composition is also computationally regular. This later route again harkens back to Kaplan and Kay (1994)'s lucid discussion on why cyclicity is still a problem.

Appendix

This appendix summarizes the model signature for the relational structures used in this chapter. The list of atomic relations in Tables 20.3 and 20.4 are the ones used in this chapter.

Types of Nodes (Domain Elements)
oper(x)
morpheme(x)
segment(x)
syll(x)
Successor Relations
succ:seg(x,y)
succ:oper(x)
succ:syll(x,y)
Dominance relations
MDom(x,y)
PDom:syll_nuc(x,y)
PDom:syll_ons(x,y)
PDom:syll_coda(x,y)
PDom:PWord_syll(x,y)
Relations connecting types of structure
operate_at(x,y)
Parse:MWord(x)
Match:word(x,y)
Others
Cophon:SLevel(x)
Cophon:WLevel(x)
MRoot(x)
Op:Root(x)
stressed(x)
destressed(x)

Table 20.3: Part 1 of the $\mathfrak{R}$-signature for the structures in this chapter.

Language Specific Morphemes
Op:N(x)
…
Op:Der $-$ $\widehat{\text{it}\int}$(x)
…
Op:Inf $-$ e(x)
…
Language Specific Phonemes
i
…
Language Specific Features
consonantal
…

Table 20.4: Part 2 of the $\mathfrak{R}$-signature for the structures in this chapter.

Bibliography

Aksënova, Alëna, Thomas Graf, and Sedigheh Moradi. 2016. Morphotactics as tier-based strictly local dependencies. In *Proceedings of the 14th sigmorphon workshop on computational research in phonetics, phonology, and morphology*, 121–130.

Albro, Dan. 2005. A large-scale, LPM-OT analysis of Malagasy. Doctoral dissertation, University of California, Los Angeles.

Albro, Daniel M. 2000. Taking Primitive Optimality Theory beyond the finite state. In *Finite-state phonology: Proceedings of the 5th Workshop of SIGPHON*, edited by Jason Eisner, Lauri Karttunen, and Alain Thériault, 57–67. Luxembourg.
URL `http://aclanthology.coli.uni-saarland.de/pdf/W/W00/W00-1806.pdf`

Albro, Daniel M. 2003. A large-scale, computerized phonological analysis of malagasy. In *Talk presented at the Annual Meeting of the Linguistic Society of America*.

Allauzen, Cyril, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, vol. 4783 of *Lecture Notes in Computer Science*, 11–23. Springer.
URL `http://www.openfst.org`

Allen, Margaret Reece. 1979. Morphological investigations. Doctoral dissertation, University of Connecticut, Storrs, CT.

Anderson, Stephen. 1974. *The Organization of Phonology*. Academic Press.

Archangeli, Diana, and Douglas Pulleyblank. 2022. *Emergent phonology*, vol. 7 of *Conceptual Foundations of Language Science*. Berlin: Language Science Press.

Aronoff, Mark. 1976. *Word formation in generative grammar*. No. 1 in Linguistic Inquiry Monographs. Cambridge, MA: The MIT Press.

Baković, Eric. 2000. Harmony, dominance and control. Doctoral dissertation, Rutgers University.

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

Bale, Alan, and Charles Reiss. 2018. *Phonology: A Formal Introduction*. The MIT Press.

Basbøll, Hans. 1972. Some conditioning phonological factors for the pronunciation of short vowels in danish with special reference to syllabification. In *Annual Reports of the Institute of Phonetics*, vol. 6, 185–210. University of Copenhagen.

Basbøll, Hans. 2005. *The phonology of Danish*. Oxford University Press.

Beauquier, D., and J.E. Pin. 1991. Languages and scanners. *Theoretical Computer Science* 84:3–21.

Becker, Michael. 2022. Cairene arabic stress is local. *Radical: A Journal of Phonology* 4:211–247.

Beesley, Kenneth, and Lauri Karttunen. 2000. Finite-state non-concatenative morphotactics. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, 191–198. Hong Kong: Association for Computational Linguistics.
URL https://doi.org/10.3115/1075218.1075243

Beesley, Kenneth, and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Publications.

Benedikt, Michael, Leonid Libkin, Thomas Schwentick, and Luc Segoufin. 2001. A model-theoretic approach to regular string relations. *Logic in Computer Science, Symposium on* 0:0431.

Benua, Laura. 1997. Transderivational identity: Phonological relations between words. Doctoral dissertation, University of Massachusetts, Amherst.

Bermúdez-Otero, Ricardo. 2011. Cyclicity. In *The Blackwell companion to phonology*, edited by Marc van Oostendorp, Colin Ewen, Elizabeth Hume, and Keren Rice, vol. 4, 2019–2048. Malden, MA: Wiley-Blackwell.

Bjorkman, Bronwyn, and Ewan Dunbar. 2016. Finite-state phonology predicts a typological gap in cyclic stress assignment. *Linguistic Inquiry* 47:351–363.

Blevins, Juliette. 1995. Syllable in phonological theory. In *The Handbook of Phonological Theory*, edited by John Goldsmith, 206–244. Blackwell Publishers.

Blumenfeld, Lev. 2006. Constraints on phonological interactions. Doctoral dissertation, Stanford University.

Blust, Robert A. 2001. Thao triplication. *Oceanic Linguistics* 40:324–335.

Booij, Geert, and Rochelle Lieber. 1993. On the simultaneity of morphological and prosodic structure. In *Studies in lexical phonology*, edited by Sharon Hargus and Ellen M. Kaisse, vol. 4 of *Phonetics and Phonology*, 23–44. San Diego: Academic Press.

Booij, Geert, and Jerzy Rubach. 1987. Postcyclic versus postlexical rules in lexical phonology. *Linguistic Inquiry* 18:1–44.

Brame, Michael K. 1974. The cycle in phonology: Stress in Palestinian, Maltese, and Spanish. *Linguistic Inquiry* 5:39–60.

Brentari, Diane. 1990. Licensing in asl handshape change. *Sign language research: Theoretical issues* .

Brentari, Diane. 1998. *A prosodic model of sign language phonology*. Mit Press.

Brentari, Diane. 2019. *Sign Language Phonology*. Cambridge University Press.

Brody, Michael. 1990. Some remarks on the focus field in hungarian. *UCL Working Papers in Linguistics* 2:201–225.

Büchi, J. Richard. 1960. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly* 6:66–92.

Carden, Guy. 1983. The non-finite-state-ness of the word formation component. *Linguistic Inquiry* 14:537–541.

Caron, Pascal. 2000. Families of locally testable languages. *Theoretical Computer Science* 242:361–376.

Carrier, Jill Louise. 1979. The interaction of morphological and phonological rules in tagalog: a study in the relationship between rule components in grammar. Doctoral dissertation, Massachusetts Institute of Technology.

Chandlee, Jane. 2014. Strictly local phonological processes. Doctoral dissertation, The University of Delaware.

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

Chandlee, Jane. 2017b. Computational locality in morphological maps. *Morphology* 27:1–43.

Chandlee, Jane, Rémi Eyraud, and Jeffrey Heinz. 2014. Learning strictly local subsequential functions. *Transactions of the Association for Computational Linguistics* 2:491–503.
URL `http://aclweb.org/anthology/Q14-1038`

Chandlee, Jane, Rémi Eyraud, and Jeffrey Heinz. 2015. Output strictly local functions. In *14th Meeting on the Mathematics of Language*, 112–125.

Chandlee, Jane, Remi Eyraud, Jeffrey Heinz, Adam Jardine, and Jonathan Rawski. 2019. Learning with partially ordered representations. In *Proceedings of the 16th Meeting on the Mathematics of Language*, 91–101. Toronto, Canada: Association for Computational Linguistics.

Chandlee, Jane, and Jeffrey Heinz. 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. Montreal, Canada: Association for Computational Linguistics.

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

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

Chandlee, Jane, and Adam Jardine. 2019a. Autosegmental input-strictly local functions. *Transactions of the Association for Computational Linguistics* 7:157–168.

Chandlee, Jane, and Adam Jardine. 2019b. Quantifier-free least fixed point functions for phonology. In *Proceedings of the 16th Meeting on the Mathematics of Language*, 50–62. Toronto, Canada: Association for Computational Linguistics.

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

Chomsky, Noam. 1995. *The Minimalist Program*. The MIT Press.

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

Chomsky, Noam, and Morris Halle. 1968. *The Sound Pattern of English*. New York: Harper & Row.

Clark, Alexander. 2017. Computational learning of syntax. *Annual Review of Linguistics* 3:107–123.

Clark, Alexander, and Ryo Yoshinaka. 2012. Beyond semilinearity: Distributional learning of parallel multiple context-free grammars. In *International Conference on Grammatical Inference*, 84–96.

Clark, Alexander, and Ryo Yoshinaka. 2014. Distributional learning of parallel multiple context-free grammars. *Machine Learning* 96:5–31.

Clements, George. 1990. The role of the sonority cycle in core syllabification. In *Papers in Laboratory Phonology*, edited by John Kingston and Mary Beckmann, vol. 1, 283–333. Cambridge: Cambridge University Press.

Clements, George, and Jay Keyser. 1983. *CV phonology: a generative theory of the syllable*. Cambridge, MA: MIT Press.

Cohen-Sygal, Yael, and Shuly Wintner. 2006. Finite-state registered automata for non-concatenative morphology. *Computational Linguistics* 32:49–82.

Cohn, Abigail C. 1989. Stress in Indonesian and bracketing paradoxes. *Natural language & linguistic theory* 7:167–216.

Cole, Jennifer. 1990. Arguing for the phonological cycle: A critical review. In *Proceedings of the Formal Linguistics Society of Midamerica*, edited by Denis Meyer, Satoshi Tomioka, and Leyla Zidani-Eroglue, 51–67. Linguistics Student Association, Madison, WI: University of Wisconson.

Cole, Jennifer. 1995a. The cycle in phonology. In *The Handbook of Phonological Theory*, edited by John Goldsmith, 1 ed., 70–113. Cambridge, MA: Blackwell Publishers.

Cole, Jennifer. 1995b. Eliminating cyclicity as a source of complexity in processing phonology. In *Linguistics and Computation*, edited by Jennifer Cole, Georgia M. Green, and Jerry L. Morgan, no. 52 in CSLI Lecture Notes, 255–280. Stanford, CA: CSLI Publications.

Cole, Jennifer, and John Coleman. 1992. No need for cyclicity in generative phonology. In *Proceedings of the 28th Regional Meeting of the Chicago Linguistics Society*, edited by Costas P. Canakis, Grace P. Chan, and Jeannette Marshall Denton, vol. 2, 36–50. Chicago, IL: University of Chicago.

Coleman, Jason. 2004. Thesis shmesis: Representing reduplication with directed graphs. Bachelor's thesis, Haverford College, Haverford, PA.

Coleman, John. 1995. Declarative lexical phonology. In *Frontiers of phonology: Atoms, structures, derivations*, edited by Jacques Durand and Francsis Katamba, 333–383. London: Longman.

Coleman, John. 1998. *Phonological representations: Their names, forms and powers*. Cambridge University Press.

Coleman, John, and John Local. 1991. The "no crossing constraint" in autosegmental phonology. *Linguistics and Philosophy* 14:295–338.

Courcelle, Bruno. 1994. Monadic second-order definable graph transductions: A survey. *Theoretical Computer Science* 126:53–75.

Courcelle, Bruno. 1997. The expression of graph properties and graph transformations in monadic second-order logic. In *Handbook of Graph Grammars and Computing by Graph Transformations*, edited by Grzegorz Rozenberg, vol. 1, 313–400. World Scientific.

Courcelle, Bruno, and Joost Engelfriet. 2012. *Graph Structure and Monadic Second-Order Logic, a Language Theoretic Approach*. Cambridge University Press.

Culy, Christopher. 1985. The complexity of the vocabulary of Bambara. *Linguistics and Philosophy* 8:345–351.

Danis, Nick, and Adam Jardine. 2019. Q-theory representations are logically equivalent to autosegmental representations. In *Proceedings of the Society for Computation in Linguistics*, vol. 2, 29–38.

Davis, Stuart. 1988. *Topics in Syllable Geometry*. Outstanding Dissertations in Linguistics. New York: Garland Press.

Dolatian, Hossep. 2020a. Computational locality of cyclic phonology in armenian. Doctoral dissertation, Stony Brook University.

Dolatian, Hossep. 2020b. Cyclicity and prosodic misalignment in Armenian stems: Interaction of morphological and prosodic cophonologies. *Natural Language and Linguistic Theory* .

Dolatian, Hossep, and Jeffrey Heinz. 2018. Modeling reduplication with 2-way finite-state transducers. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 66–77. Brussels, Belgium: Association for Computational Linguistics.

Dolatian, Hossep, and Jeffrey Heinz. 2020. Computing and classifying reduplication with 2-way finite-state transducers. *Journal of Language Modeling* 8:179–250.

Dolatian, Hossep, Jonathan Rawski, and Jeffrey Heinz. 2021. Strong generative capacity of morphological processes. In *Proceedings of the Society for Computation in Linguistics*, vol. 4, 228–243.

Downing, Laura J. 1999. Prosodic stem $\neq$ prosodic word in Bantu. In *Studies on the phonological word*, edited by T Alan Hall and Ursula Kleinhenz, vol. 174, 73–98. Amsterdam/Philadelphia: John Benjamins Publishing.

Downing, Laura J. 2001. Review of Eric Raimy (2000). the phonology and morphology of reduplication. (Studies in Generative Grammar 52.) Berlin: Mouton de Gruyter. pp. viii + 200. *Phonology* 18:445.

Dresher, Elan B. 2011. The phoneme. In *The Blackwell Companion to Phonology*, edited by Elizabeth Hume Marc van Oostendorp, Colin J. Ewen and Keren Rice, vol. 1, 241–266. Malden, MA & Oxford: Wiley-Blackwell.

Droste, Manfred, and Paul Gastin. 2009. Weighted automata and weighted logics. In Droste *et al.* (2009), chap. 5.

Droste, Manfred, and Werner Kuich. 2009. Semirings and formal power series. In Droste *et al.* (2009), chap. 1.

Droste, Manfred, Werner Kuich, and Heiko Vogler, eds. 2009. *Handbook of Weighted Automata*. Monographs in Theoretical Computer Science. Springer.

Durvasula, Karthik, and Scott Nelson. 2018. Lexical retuning targets features. In *Proceedings of the Annual Meetings on Phonology*, edited by Gillian Gallagher, Maria Gouskova, and Sora Yin. Linguistic Society of America.

É. Kiss, Katalin. 1981. Structural relations in hungarian, a ”free” word order language. *Linguistic Inquiry* 12:185–213.

É. Kiss, Katalin. 2002. *The Syntax of Hungarian*. Cambridge Syntax Guides. Cambridge University Press.

Eisner, Jason. 1997. Efficient generation in primitive optimality theory. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, 313–320.

Eisner, Jason. 2000a. Directional constraint evaluation in optimality theory. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*.

Eisner, Jason. 2000b. Easy and hard constraint ranking in ot: Algorithms and complexity. In *Proceedings of the Fifth Workshop of the ACL Special Interest Group in Computational Phonology*, 22–33.

Eliasson, S. 1985. Turkish k-deletion: simplicity vs. retrieval. *Folia Linguistica* 19:289–312.

Embick, David. 2010. *Localism versus globalism in morphology and phonology*, vol. 60 of *Linguistic Inquiry Monographs*. Cambridge, MA: MIT Press.

Enderton, Herbert B. 2001. *A Mathematical Introduction to Logic*. 2nd ed. Academic Press.

Engelfriet, Joost. 2015. Two-way pebble transducers for partial functions and their composition. *Acta Informatica* 52:559–571.

Engelfriet, Joost, and Hendrik Jan Hoogeboom. 2001. MSO definable string transductions and two-way finite-state transducers. *Transactions of the Association for Computational Linguistics* 2:216–254.
URL `http://doi.acm.org/10.1145/371316.371512`

Engelfriet, Joost, and Sebastian Maneth. 2002. Two-way finite state transducers with nested pebbles. In *International Symposium on Mathematical Foundations of Computer Science*, 234–244. Springer.

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

Finley, Sara. 2008. The formal and cognitive restrictions on vowel harmony. Doctoral dissertation, Johns Hopkins University, Baltimore, MD.

Fitzpatrick, Justin. 2004. A concatenative theory of possible affix types. In *Papers from EVELIN I. MIT Working Papers in Linguistics*, edited by Andrés Salanova.

Fitzpatrick, Justin. 2006. Sources of multiple reduplication in Salish and beyond. In *MIT Working Papers on Endangered and Less Familiar Languages: Studies in Salishan 7*, edited by Shannon T. Bischoff, Lynnika Butler, Peter Norquest, and Daniel Siddiqi, 211–240.

Fitzpatrick, Justin, and Andrew Nevins. 2002. Phonological occurrences: Relations and copying. In *Proceedings of the 2nd North American Phonology Conference*. Montreal.

Fitzpatrick, Justin, and Andrew Nevins. 2004. Linearizing nested and overlapping precedence in multiple reduplication. In *University of Pennsylvania Working Papers in Linguistics*, 75–88.

Flack, Kathryn. 2009. Constraints on onsets and codas of words and phrases. *Phonology* 26:269–302.

Frampton, John. 2009. *Distributed reduplication*. Cambridge, MA: MIT Press.

Frank, Robert, and Giorgo Satta. 1998. Optimality Theory and the generative complexity of constraint violability. *Computational Linguistics* 24:307–315.

Frank, Robert, and K. Vijay-Shanker. 2001. Primitive c-command. *Syntax* 4:164–204.

Frishberg, Nancy. 1975. Arbitrariness and iconicity: historical change in american sign language. *Language* 696–719.

Gazdar, Gerald, and Geoffrey K Pullum. 1985. Computationally relevant properties of natural languages and their grammars. *New generation computing* 3:273–306.

Gerdemann, Dale, and Mans Hulden. 2012. Practical finite state optimality theory. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, 10–19.

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

Goedemans, R. W. N., Jeffrey Heinz, and Harry van der Hulst. 2015. StressTyp2.
URL `http://st2.ullet.net/`

Golan, Jonathan S. 1999. *Semirings and their Applications*. Springer.

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

Goldsmith, John. 1976. Autosegmental phonology. Doctoral dissertation, MIT, Cambridge, MA.

Goldsmith, John. 2011. The syllable. In *The Blackwell Handbook of Phonological Theory*, edited by John A. Goldsmith, Jason Riggle, and Alan C. L. Yu, 164–196. Wiley-Blackwell.

Goodman, Joshua. 1999. Semiring parsing. *Computational Linguistics* 25:573–606.

Gorman, Kyle. 2016. Pynini: A python library for weighted finite-state grammar compilation. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*, 75–80. Berlin, Germany.

Gorman, Kyle, and Richard Sproat. 2021. *Finite-State Text Processing*. Morgan & Claypool Publishers.

Graf, Thomas. 2010a. Comparing incomparable frameworks: A model theoretic approach to phonology. In *University of Pennsylvania Working Papers in Linguistics*, vol. 16.

Graf, Thomas. 2010b. Logics of phonological reasoning. Master's thesis, University of California, Los Angeles.

Graf, Thomas. 2013. Local and transderivational constraints in syntax and semantics. Doctoral dissertation, University of California, Los Angeles.

Guekguezian, Peter Ara. 2017. Prosodic recursion and syntactic cyclicity inside the word. Doctoral dissertation, University of Southern California.

Guekguezian, Peter Ara. 2021. Aspectual phase heads in Muskogee verbs. *Natural Language & Linguistic Theory* 39:1129–1172.

Guimarães, Maximiliano, and Andrew Nevins. 2012. Opaque nasalization in ludlings and the precedence relations of reduplication and infixation. *Letras & Letras* 28:129–166.

Gussmann, Edmund. 2007. *The Phonology Of Polish*. Oxford University Press.

Halle, Morris. 2008. Reduplication. *Current studies in linguistics series* 45:325.

Halle, Morris, and George N. Clements. 1983. *Problem Book in Phonology: A workbook for introductory courses in linguistics and in modern phonology*. MIT Press.

Halle, Morris, and K. P. Mohanan. 1985. Segmental phonology of Modern English. *Linguistic Inquiry* 16:57–116.

Hammond, Michael. 1988. On deriving the well-formedness condition. *Linguistic Inquiry* 19:319–325.

Hammond, Michael. 1993. On the absence of category-changing prefixes in English. *Linguistic Inquiry* 24:562–567.

Hansson, Gunnar. 2010. *Consonant Harmony: Long-Distance Interaction in Phonology*. No. 145 in University of California Publications in Linguistics. Berkeley, CA: University of California Press. Available on-line (free) at eScholarship.org.

Hao, Sophie. 2024. Universal generation for Optimality Theory is PSPACE-complete. *Computational Linguistics* 50:83–117.

Hao, Yiding. 2019. Finite-state optimality theory: non-rationality of harmonic serialism. *Journal of Language Modelling* 7:49–99.

Harris, James. 1983. *Syllable Structure And Stress in Spanish: a Nonlinear Analysis*. Cambridge, Mass.: MIT Press.

Harris, James, and Morris Halle. 2005. Unexpected plural inflections in Spanish: Reduplication and metathesis. *Linguistic Inquiry* 36:195–222.

Harrison, K David, and Eric Raimy. 2004. Reduplication in tuvan: Exponence, readjustment and phonology. In *Proceedings of Workshop in Altaic Formal Linguistics*, vol. 1. Citeseer.

Hauser, Ivy, and Coral Hughto. 2020. Analyzing opacity with contextual faithfulness constraints. *Glossa: a journal of general linguistics* 5.

Hauser, Ivy, Coral Hughto, and Megan Somerday. 2016. Faith-uo: Counterfeeding in harmonic serialism. In *Proceedings of the 2014 Annual Meeting on Phonology*, vol. 2.

Hayes, Bruce. 2009. *Introductory Phonology*. Wiley-Blackwell.

Hayes, Bruce, Robert Kirchner, and Donca Steriade, eds. 2004. *Phonetically-Based Phonology*. Cambridge University Press.

Hayes, Bruce, and Donca Steriade. 2004. Introduction: the phonetic bases of phonological markedness. In Hayes *et al.* (2004), chap. 1, 1–33.

Hayes, Bruce, Bruce Tesar, and Kie Zuraw. 2013. Otsoft 2.3.2. software package.
URL `http://www.linguistics.ucla.edu/people/hayes/otsoft`

Hayes, Bruce, and James White. 2015. Saltation and the p-map. *Phonology* 32:267–302.

Hedman, Shawn. 2004. *A First Course in Logic*. Oxford University Press.

Heinz, Jeffrey. 2007. The inductive learning of phonotactic patterns. Doctoral dissertation, University of California, Los Angeles.

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

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

Heinz, Jeffrey. 2014. Culminativity times harmony equals unbounded stress. In *Word Stress: Theoretical and Typological Issues*, edited by Harry van der Hulst, chap. 8, 255–275. Cambridge, UK: Cambridge University Press.

Heinz, Jeffrey. 2018. The computational nature of phonological generalizations. In *Phonological Typology*, edited by Larry Hyman and Frans Plank, Phonetics and Phonology, chap. 5, 126–195. De Gruyter Mouton.

Heinz, Jeffrey, and William Idsardi. 2013. What complexity differences reveal about domains in language. *Topics in Cognitive Science* 5:111–131.

Heinz, Jeffrey, Gregory M Kobele, and Jason Riggle. 2009. Evaluating the complexity of optimality theory. *Linguistic Inquiry* 40:277–288.

Heinz, Jeffrey, and Regine Lai. 2013. Vowel harmony and subsequentiality. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, edited by Andras Kornai and Marco Kuhlmann, 52–63. Sofia, Bulgaria.

Hooper, J. 1976. *An Introduction to Natural Generative Phonology*. New York: Academic Press.

Hopcroft, John, Rajeev Motwani, and Jeffrey Ullman. 2006. *Introduction to Automata Theory, Languages, and Computation*. 3rd ed. Addison-Wesley.

Hopcroft, John E, and Jeffrey D Ullman. 1969. *Formal languages and their relation to automata*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.

Horvath, Julia. 1976. Focus in hungarian and $\bar{x}$-notation. *Linguistic Analysis* 2:175–197.

Hulden, Mans. 2009a. Finite-state machine construction methods and algorithms for phonology and morphology. Doctoral dissertation, University of Arizona.

Hulden, Mans. 2009b. Foma: A finite-state compiler and library. In *Proceedings of the Demonstrations Session at EACL 2009*, 29–32. Athens, Greece: Association for Computational Linguistics.
URL `http://www.aclweb.org/anthology/E09-2008`

Hulden, Mans, and Shannon T Bischoff. 2009. A simple formalism for capturing reduplication in finite-state morphology. In *Proceedings of the 2009 conference on Finite-State Methods and Natural Language Processing: Post-proceedings of the 7th International Workshop FSMNLP 2008*, edited by Jakub Piskorski, Bruce Watson, and Anssi Yli-Jyrä, 207–214. Amsterdam: IOS Press.
URL `http://dl.acm.org/citation.cfm?id=1564035.1564059`

van der Hulst, Harry. 1993. Units in the analysis of signs. *Phonology* 10:209–241.

van der Hulst, Harry. 1994. Dependency relations in the phonological representation of signs. *Sign language research* 11–38.

van der Hulst, Harry, ed. 2014. *Word Stress: Theoretical and Typological Issues*. Cambridge University Press.

von Humboldt, Wilhelm. 1999. *On Language*. Cambridge Texts in the History of Philosophy. Cambridge University Press. Edited by Michael Losonsky. Translated by Peter Heath. Originally published 1836.

Hurch, Bernhard, ed. 2005. *Studies on reduplication*. No. 28 in Empirical Approaches to Language Typology. Berlin: Walter de Gruyter.

Hyman, Larry. 1975. *Phonology: Theory and Analysis*. Holt, Rinehart and Winston.

Hyman, Larry M. 2009. How (not) to do phonological typology: the case of pitch-accent. *Language Sciences* 31:213 – 238. Data and Theory: Papers in Phonology in Celebration of Charles W. Kisseberth.

Idsardi, William, and Eric Raimy. 2008. Reduplicative economy. In *Rules, Constraints, and Phonological Phenomena*, edited by Bert Vaux and Andrew Nevins, chap. 5, 149–184. Oxford: Oxford University Press.

Idsardi, William, and Eric Raimy. 2013. Three types of linearization and the temporal aspects of speech. In *Challenges to linearization*, edited by Eric Raimy and Charles Cairns, 31–56. Mouton de Gruyter, Berlin.

Idsardi, William J. 2006. A simple proof that optimality theory is computationally intractable. *Linguistic Inquiry* 37:271–275.

Idsardi, William J, and Eric Raimy. 2005. Remarks on language play. Unpublished manuscript, University of Delaware, Newark DE.

Idsardi, William J, and Rachel Shorey. 2007. Unwinding morphology. Presented at CUNY Phonology Forum Workshop on Precedence Relations.

Inkelas, Sharon. 2014. *The interplay of morphology and phonology*. Oxford: Oxford University Press.

Inkelas, Sharon, and Laura J Downing. 2015a. What is reduplication? Typology and analysis part 1/2: The typology of reduplication. *Language and Linguistics Compass* 9:502–515.

Inkelas, Sharon, and Laura J Downing. 2015b. What is reduplication? Typology and analysis part 2/2: The analysis of reduplication. *Language and Linguistics Compass* 9:516–528.

Inkelas, Sharon, and Cheryl Zoll. 2005. *Reduplication: Doubling in Morphology*. Cambridge: Cambridge University Press.

Ito, Junko. 1986. Syllable Theory in Prosodic Phonology. Doctoral dissertation, University of Massachusetts, Amherst. Published 1988. Outstanding Dissertations in Linguistics series. New York: Garland.

Itô, Junko. 1989. A prosodic theory of epenthesis. *Natural Language & Linguistic Theory* 7:217–259.

Jakobson, Roman. 1962. *Selected Writings 1: Phonological Studies*. The Hague: Mouton & Co. Second expanded edition.

Jardine, Adam. 2016. Locality and non-linear representations in tonal phonology. Doctoral dissertation, University of Delaware.

Jardine, Adam. 2017. The local nature of tone-association patterns. *Phonology* 34:363–384.

Jardine, Adam, Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. 2014. Very efficient learning of structured classes of subsequential functions from positive data. In *International Conference on Grammatical Inference*, 94–108.

Jardine, Adam, Nick Danis, and Luca Iacoponi. 2021. A formal investigation of q-theory in comparison to autosegmental representations. *Linguistic Inquiry* 52:333–358.
URL `https://doi.org/10.1162/ling_a_00376`

Jassem, Wiktor. 2003. Polish. *Journal of the International Phonetic Association* 33.

Johnson, C Douglas. 1972. *Formal aspects of phonological description*. The Hague: Mouton.

Kager, René. 1999. *Optimality Theory*. Cambridge University Press.

Kaisse, Ellen M, and April McMahon. 2011. Lexical phonology and the lexical syndrome. In *The Blackwell companion to phonology*, edited by Marc van Oostendorp, Colin Ewen, Elizabeth Hume, and Keren Rice, vol. 4, 2236–2257. Malden, MA: Wiley-Blackwell.

Kaisse, Ellen M, and Patricia A Shaw. 1985. On the theory of lexical phonology. *Phonology* 2:1–30.

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

Karttunen, Lauri. 1998. The proper treatment of optimality in computational phonology. In *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing*, 1–12. International Workshop on Finite-State Methods in Natural Language Processing, Bilkent University, Ankara, Turkey.

Karttunen, Lauri. 2003. Computing with realizational morphology. In *International Conference on Intelligent Text Processing and Computational Linguistics*, 203–214. Springer.

Karttunen, Lauri. 2006. The insufficiency of paper-and-pencil linguistics: the case of Finnish prosody. Rutgers Optimality Archive #818-0406.

Keisler, H. Jerome, and Joel Robbin. 1996. *Mathematical Logic and Computability*. McGraw-Hill.

Kenstowicz, Michael, and Charles Kisseberth. 1977. *Topics in Phonological Theory*. New York: Academic Press.

Kenstowicz, Michael, and Charles Kisseberth. 1979. *Generative Phonology*. Academic Press, Inc.

Kenstowicz, Michael, and Charles Kisseberth. 1990. Chizigula tonology: the word and beyond. In *The Phonology–Syntax Connection*, edited by Sharon Inkelas and Draga Zec, 163–194. Chicago: the University of Chicago Press.

Kiparsky, Paul. 1973. Abstractness, opacity, and global rules. In *Three dimensions of linguistic theory*, edited by Osamu Fujimura. Tokyo: Taikusha.

Kiparsky, Paul. 1982. Lexical morphology and phonology. In *Linguistics in the morning calm: Selected papers from SICOL-1981*, edited by I.-S. Yang, 3–91. Seoul: Hansin.

Kiparsky, Paul. 1983. Word-formation and the lexicon. In *Proceedings of the 1982 Mid-America Linguistics Conference*, edited by Frances Ingemann, 3–29. Mid-America Linguistics Conference, Lawrence: University of Kansas.

Kiparsky, Paul. 1993. Blocking in non-derived environments. In *Studies in lexical phonology*, edited by Sharon Hargus and Ellen M. Kaisse, vol. 4 of *Phonetics and Phonology*, 277–313. San Diego: Academic Press.

Kiparsky, Paul. 2010. Reduplication in stratal OT. *Reality exploration and discovery: Pattern interaction in language & life* 125–142.

Kobele, Gregory Michael. 2006. Generating copies: An investigation into structural identity in language and grammar. Doctoral dissertation, University of California, Los Angeles.

Kornai, András. 1995. *Formal phonology*. Garland Publishing Inc.

Kornai, András. 2009. The complexity of phonology. *Linguistic Inquiry* 40:701–712.

Koskenniemi, Kimmo. 1983. Two-level morphology: A general computational model for word-form recognition and production. Doctoral dissertation, University of Helsinki.

Kozen, Dexter. 1997. *Automata and Computability*. Springer.

Krämer, Martin. 2012. *Underlying Representations*. Cambridge University Press.

de Lacy, Paul. 2011. Markedness and faithfulness constraints. In *The Blackwell Companion to Phonology*, edited by M. V. Oostendorp, C. J. Ewen, E. Hume, and K. Rice. Blackwell.

Lai, Regine. 2015. Learnable vs. unlearnable harmony patterns. *Linguistic Inquiry* 46:425–451.

Lambert, Dakotah. 2022. Unifying classification schemes for languages and processes with attention to locality and relativizations thereof. Doctoral dissertation, Stony Brook University.
URL `https://vvulpes0.github.io/PDF/dissertation.pdf/`

Lambert, Dakotah. 2023. Relativized adjacency. *Journal of Logic Language and Information* .

Lambert, Dakotah. 2024. System description: A theorem-prover for subregular systems: The language toolkit and its interpreter, plebby. In *Functional and Logic Programming*, edited by Jeremy Gibbons and Dale Miller, 311–328. Singapore: Springer Nature Singapore.

Lambert, Dakotah, and Jeffrey Heinz. 2023. An algebraic characterization of total input strictly local functions. In *Proceedings of the Society for Computation in Linguistics*, vol. 6.

Lambert, Dakotah, Jonathan Rawski, and Jeffrey Heinz. 2021. Typology emerges from simplicity in representations and learning. *Journal of Language Modelling* 9:151–194.

Lambert, Dakotah, and James Rogers. 2020. Tier-based strictly local stringsets: Perspectives from model and automata theory. In *Proceedings of the Society for Computation in Linguistics*, vol. 3, 330–337. New Orleans, Louisiana.

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

Lamont, Andrew. 2022a. Directional harmonic serialism. Doctoral dissertation, University of Massachusetts Amherst.

Lamont, Andrew. 2022b. Optimality theory implements complex functions with simple constraints. *Phonology* 38:729–740.

Langendoen, D Terence. 1981. The generative capacity of word-formation components. *Linguistic Inquiry* 12:320–322.

Lapoliwa, Hans. 1981. *A generative approach to the phonology of Bahasa Indonesia*. Dept. of Linguistics, Research School of Pacific Studies, The Australian.

Law, Howard. 1958. Morphological structure of Isthmus Nahuat. *International Journal of American Linguistics* 24:108–129.

Lepic, Ryan. 2015. Motivation in morphology: Lexical patterns in asl and english. Doctoral dissertation, UC San Diego.

Levelt, Willem. 1974. *Formal grammars in linguistics and psycholinguistics*. The Hague: Mouton.

Liddell, Scott K. 1984. Think and believe: sequentiality in american sign language. *Language* 372–399.

Liddell, Scott K, and Robert E Johnson. 1986. American sign language compound formation processes, lexicalization, and phonological remnants. *Natural Language & Linguistic Theory* 4:445–513.

Liddell, Scott K, and Robert E Johnson. 1989. American sign language: The phonological base. *Sign language studies* 64:195–277.

Lieber, Rochelle. 1980. On the organization of the lexicon. Doctoral dissertation, Massachusetts Institute of Technology.

Łubowicz, Anna. 2002. Derived environment effects in Optimality Theory. *Lingua* 112:243–280.

Luo, Huan. 2017. Long-distance consonant agreement and subsequentiality. *Glossa* 2:1–25.

Manaster-Ramer, Alexis. 1986. Copying in natural languages, context-freeness, and queue grammars. In *Proceedings of the 24th annual meeting on Association for Computational Linguistics*, 85–89. Association for Computational Linguistics.

Marantz, Alec. 1982. Re reduplication. *Linguistic Inquiry* 13:435–482.

Marantz, Alec. 2007. Phases and words. In *Phases in the theory of grammar*, edited by Sook-Hee Choe, Dong-Wee Yang, Yang-Soon Kim, Sung-Hun Kim, and Alec Marantz, 191–222. Seoul: Dong-In Publishing Co.

Marvin, Tatjana. 2002. Topics in the stress and syntax of words. Doctoral dissertation, Massachusetts Institute of Technology.

Mascaró, Joan. 1976. Catalan phonology and the phonological cycle. Doctoral dissertation, Massachusetts Institute of Technology.

McCarthy, John. 1979. Formal problems in semitic phonology and morphology. Doctoral dissertation, MIT, Cambridge, MA.

McCarthy, John. 2003. OT constraints are categorical. *Phonology* 20:75–138.

McCarthy, John. 2008a. *Doing Optimality Theory*. Malden, MA: Blackwell.

McCarthy, John J. 2008b. The gradual path to cluster simplification. *Phonology* 25:271–319.

McCarthy, John J. 2010. An introduction to harmonic serialism. *Language and Linguistics Compass* 4:1001–1018.

McCarthy, John J, and Alan Prince. 1995. Faithfulness and reduplicative identity. In *Papers in Optimality Theory*, edited by Jill N. Beckman, Laura Walsh Dickey, and Suzanne Urbanczyk. Amherst, MA: Graduate Linguistic Student Association, University of Massachusetts.

McClory, Daniel, and Eric Raimy. 2007. Enhanced edges: morphological influence on linearization.

McNaughton, Robert, and Seymour Papert. 1971. *Counter-Free Automata*. MIT Press.

Miller, Philip H. 1999. *Strong generative capacity: The semantics of linguistic formalism*. Stanford: CSLI publications.

Mohri, Mehryar, and Richard Sproat. 1996. An efficient compiler for weighted rewrite rules. In *Proceedings of the 34th Meeting of the Association for Computational Linguistics (ACL '96)*.

Moravcsik, Edith. 1978. Reduplicative constructions. In *Universals of Human Language*, edited by Joseph Greenberg, vol. 1, 297–334. Stanford, California: Stanford University Press.

Nazarov, Aleksei, and Joe Pater. 2017. Learning opacity in stratal maximum entropy grammar. *Phonology* 34:299–324.

Nelson, Scott. 2022. A model theoretic perspective on phonological feature systems. In *Proceedings of the Society for Computation in Linguistics 2022*, edited by Allyson Ettinger, Tim Hunter, and Brandon Prickett, 1–10. online: Association for Computational Linguistics.
URL `https://aclanthology.org/2022.scil-1.1`

Nespor, Marina, and Irene Vogel. 1986. *Prosodic phonology*. Dordrecht: Foris.

Nevins, Andrew. 2004. What ug can and can't do to help the reduplication learner. In *MIT Working Papers in Linguistics 48*, edited by Aniko Cslrmaz, Andrea Gualmini, and Andrew Nevins, 113–126. Cambridge,MA: MIT Department of Linguistics and Philosophy.

Newell, Heather. 2008. Aspects of the morphology and phonology of phases. Doctoral dissertation, McGill University, Montreal, QC.

Newell, Heather. 2021. Deriving level 1/level 2 affix classes in English: Floating vowels, cyclic syntax. *Acta Linguistica Academica* 68:31–76.

Newell, Heather, Máire Noonan, and Glyne Piggott, eds. 2017. *The structure of words at the interfaces*, vol. 68. Oxford: Oxford University Press.

Newkirk, Don E. 1998. On the temporal segmentation of movement in american sign language. *Sign language & linguistics* 1:173–211.

Oakden, Chris. 2020. Notational equivalence in tonal geometry. *Phonology* 37:257–296.

Odden, David. 1982. Tonal phenomena in Kishambaa. *Studies in African Linguistics* 13:177–208.

Odden, David. 1994. Adjacency parameters in phonology. *Language* 70:289–330.

Odden, David. 2014. *Introducing Phonology*. 2nd ed. Cambridge University Press.

Oncina, Jose, and Pedro Garcia. 1991. Inductive learning of subsequential functions. Tech. Rep. DSIC II-34, University Politécnia de Valencia.

van Oostendorp, Marc, Colin Ewen, Elizabeth Hume, and Keren Rice, eds. 2011. *The Blackwell companion to phonology*. Malden, MA: Wiley-Blackwell.

Oseki, Yohei. 2018. Syntactic structures in morphological processing. Doctoral dissertation, New York University.

Oseki, Yohei, and Alec Marantz. 2020. Modeling human morphological competence. *Frontiers in Psychology* 11.

Oseki, Yohei, Charles Yang, and Alec Marantz. 2019. Modeling hierarchical syntactic structures in morphological processing. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, 43–52. Minneapolis, Minnesota: Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W19-2905

Papillon, Maxime. 2020. Precedence and the lack thereof: Precedence-relation-oriented phonology. Doctoral dissertation, University of Maryland.

Pater, Joe. 1999. Austronesian nasal substitution and other *NC̥ effects. In *The Prosody–Morphology Interface*, edited by René Kager, Harry van der Hulst, and Wim Zonneveld. Cambridge: Cambridge University Press. ROA 160-1196.

Pater, Joe. 2004. Austronesian nasal substitution and other NC effects. In *Optimality Theory in Phonology: A Reader*, edited by John McCarthy, 271–289. Oxford and Malden, MA: Blackwell.

Payne, Amanda. 2017. All dissimilation is computationally subsequential. *Language* 4:e353–e371.

Payne, Amanda, Mai Ha Vu, and Jeffrey Heinz. 2017. A formal analysis of correspondence theory. In *Proceedings of the Annual Meetings on Phonology*, vol. 4.

Payne, Sarah. 2024. A generalized algorithm for learning positive and negative grammars with unconventional string models. In *Proceedings of the Society for Computation in Linguistics*, vol. 7, 75–85.

Perlmutter, David M. 1993. Sonority and syllable structure in american sign language. In *Current issues in ASL phonology*, 227–261. Elsevier.

Peters, P Stanley, and Robert W Ritchie. 1973. On the generative power of transformational grammars. *Information sciences* 6:49–83.

Postal, Paul M. 1968. *Aspects of Phonological Theory*. Harper & Row.

Potts, Christopher, and Geoffrey K Pullum. 2002. Model theory and the content of OT constraints. *Phonology* 19:361–393.

Prince, Alan. 2002. Arguing optimality. In *Papers in Optimality Theory II*, edited by Angela Carpenter, Andries Coetzee, and Paul De Lacy, no. 26 in University of Massachussetts Occasional Papers in Linguistics, 269–304. Amherst, MA: GLSA Publications. Available on Rutgers Optimality Archive, ROA-562.

Prince, Alan, and Paul Smolensky. 1993. Optimality Theory: Constraint interaction in generative grammar. Tech. Rep. 2, Rutgers University Center for Cognitive Science.

Prince, Alan, and Paul Smolensky. 2004. *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell Publishing.

Prince, Alan, Bruce Tesar, and Nazarré Merchant. 2016. Otworkplace. software package. Additions by Luca Iacoponi and Natalie DelBusso. URL `https://sites.google.com/site/otworkplace/home`

Puskás, Genoveva. 2000. Negation. In *Word Order in Hungarian: The Syntax of A'-positions*, vol. 33 of *Linguistik Aktuell*, 295–376. Amsterdam: John Benjamins Publishing Company.

Rabin, Michael Oser, and Dana Scott. 1959. Finite automata and their decision problems. *IBM Journal of Research and Development* 3:114–125.

Raimy, Eric. 1999a. Representing reduplication. Doctoral dissertation, University of Delaware, Newark, DE.

Raimy, Eric. 1999b. Strong syllable reduplication in Mokilese. In *Proceedings from ESCOL'99*, 191–202. Ithaca, NY: CLC Publications.

Raimy, Eric. 2000a. *The Phonology and Morphology of Reduplication*. Berlin: Mouton de Gruyter.

Raimy, Eric. 2000b. Remarks on backcopying. *Linguistic Inquiry* 31:541–552.

Raimy, Eric. 2003. Asymmetry and linearization in phonology. In *Asymmetry in grammar*, edited by Anna Maria Di Sciullo, vol. 2, 129–146. John Benjamins Publishing.

Raimy, Eric. 2007. Precedence theory, root and template morphology, priming effects and the structure of the lexicon.

Raimy, Eric. 2009a. A case of appendicitis. In Raimy and Cairns (2009), 177–188.

Raimy, Eric. 2009b. Deriving reduplicative templates in a modular fashion. In Raimy and Cairns (2009), 383–404.

Raimy, Eric. 2011. Reduplication. In van Oostendorp *et al.* (2011), 2383–2413.

Raimy, Eric, and Charles Cairns. 2011. Precedence relations in phonology. In van Oostendorp *et al.* (2011), 799–823.

Raimy, Eric, and Charles E. Cairns, eds. 2009. *Contemporary views on architecture and representations in phonology*. No. 48 in Current Studies in Linguistics. Cambridge, MA: MIT Press.

Rawski, Jonathan. 2017. Phonological complexity is subregular: Evidence from sign language. In *Proceedings of the 53rd Chicago Linguistics Society Annual Meeting*.

Rawski, Jonathan. 2021. Structure and learning in natural language. Doctoral dissertation, Stony Brook University.

Rawski, Jonathan, Hossep Dolatian, Jeffrey Heinz, and Eric Raimy. 2023. Regular and polyregular theories of reduplication. *Glossa: a journal of general linguistics* 8:1–38.

Reiss, Charles, and Marc Simpson. 2009. Reduplication as projection. Unpublished manuscript, Concordia University, Montréal.

Riggle, Jason. 2004. Generation, recognition, and learning in finite state Optimality Theory. Doctoral dissertation, University of California, Los Angeles.

Ristad, Eric Sven. 1990. Computational structure of human language. Doctoral dissertation, Massachusetts Institute of Technology.

Ritchie, Graeme. 1989. On the generative power of two-level morphological rules. In *Proceedings of the fourth conference on European chapter of the Association for Computational Linguistics*, 51–57. Association for Computational Linguistics.

Ritchie, Graeme. 1992. Languages generated by two-level morphological rules. *Computational Linguistics* 18:41–59.

Roark, Brian, and Richard Sproat. 2007. *Computational Approaches to Morphology and Syntax*. Oxford: Oxford University Press.

Robinson, Andrew. 2018. Einstein said that – didn't he? *Nature* 557:30.

Rogers, James. 2003. wMSO theories as grammar formalisms. *Theoretical Computer Science* 293:291–320.

Rogers, James, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. 2010. On languages piecewise testable in the strict sense. In *The Mathematics of Language*, edited by Christian Ebert, Gerhard Jäger, and Jens Michaelis, vol. 6149 of *Lecture Notes in Artifical Intelligence*, 255–265. Springer.

Rogers, James, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. 2013. Cognitive and sub-regular complexity. In

Formal Grammar, edited by Glyn Morrill and Mark-Jan Nederhof, vol. 8036 of *Lecture Notes in Computer Science*, 90–108. Springer.

Rogers, James, and Dakotah Lambert. 2019a. Extracting Subregular constraints from Regular stringsets. *Journal of Language Modelling* 7:143–176.

Rogers, James, and Dakotah Lambert. 2019b. Some classes of sets of structures definable without quantifiers. In *Proceedings of the 16th Meeting on the Mathematics of Language*, 63–77. Toronto, Canada: Association for Computational Linguistics.

Rogers, James, and Geoffrey Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information* 20:329–342.

Rose, Sharon, and Rachel Walker. 2004. A typology of consonant agreement as correspondence. *Language* 80:475–531.

Rubach, Jerzy. 1984. *Cyclic and Lexical Phonology: The Structure of Polish*. Dordrecht, The Netherlands: Foris Publications.

Sakarovitch, Jaques. 2009. *Elements of Automata Theory*. Cambridge University Press. Translated by Reuben Thomas from the 2003 edition published by Vuibert, Paris.

Samuels, Bridget. 2010. The topology of infixation and reduplication. *The Linguistic Review* 27:131–176.

Samuels, Bridget D. 2011. *Phonological architecture: A biolinguistic perspective*. Oxford Studies in Biolinguistics. Oxford University Press.

Sande, Hannah L. 2017. Distributing morphologically conditioned phonology: Three case studies from guébie. Doctoral dissertation, University of California, Berkeley, Berkeley, CA.

Sandler, Wendy. 1986. The spreading hand autosegment of american sign language. *Sign Language Studies* 50:1–28.

Sandler, Wendy. 1989. *Phonological representation of the sign: Linearity and nonlinearity in American Sign Language*, vol. 32. Walter de Gruyter.

Sandler, Wendy. 1993. Sign language and modularity. *Lingua* 89:315–351.

Sandler, Wendy, and Diane Lillo-Martin. 2006. *Sign language and linguistic universals*. Cambridge University Press.

Saussure, Ferdinand de. 1916. *Cours de linguistique générale*. Paris: Payot.

Savitch, Walter J. 1982. *Abstract machines and grammars*. Boston: Little Brown and Company.

Savitch, Walter J. 1989. A formal model for context-free languages augmented with reduplication. *Computational Linguistics* 15:250–261.

Savitch, Walter J. 1993. Why it may pay to assume that languages are infinite. *Annals of Mathematics and Artificial Intelligence* 8:17–25.

Schane, S. A. 1984. The fundamentals of particle phonology. In *Phonology Yearbook*, 129–155.

Scheer, Tobias. 2011. *A guide to morphosyntax-phonology interface theories: How extra-phonological information is treated in phonology since Trubetzkoy's Grenzsignale*. Berlin: Mouton de Gruyter.

Scheer, Tobias. 2012. Chunk definition in phonology: Prosodic constituency vs. phase structure. In *Modules and interfaces*, edited by M. Bloch-Trojnar and A. Bloch-Rozmej, 221–253. Lublin: Wydawnictwo KUL.

Schmidt, Hans. 2003. Temathesis in Rotuman. In *Issues in Austronesian historical phonology*, edited by John Lynch, 175–207.

Scobbie, James M., John S. Coleman, and Steven Bird. 1996. Key aspects of declarative phonology. In *Current Trends in Phonology: Models and Methods*, edited by Jacques Durand and Bernard Laks, vol. 2, 685–709. Manchester, UK: European Studies Research Institute. University of Salford.

Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88:191–229.

Seki, Hiroyuki, Ryuichi Nakanishi, Yuichi Kaji, Sachiko Ando, and Tadao Kasami. 1993. Parallel multiple context-free grammars, finite-state translation systems, and polynomial-time recognizable subclasses of lexical-

functional grammars. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, 130–139. Association for Computational Linguistics.

Selkirk, Elisabeth. 1984. *Phonology and syntax: The relation between sound and structure*. Cambridge, MA: MIT Press.

Selkirk, Elisabeth. 2011. The syntax-phonology interface. In *The Handbook of Phonological Theory*, edited by John Goldsmith, Jason Riggle, and Alan C. L. Yu, 2 ed., 435–483. Oxford: Blackwell.

Selkirk, Elisabeth O. 1982. *The syntax of words*. No. 7 in Linguistic Inquiry Monographs. Cambridge, Mass: MIT Press.

Shen, David Ta-Chun. 2016. Precedence and search: Primitive concepts in morpho-phonology. Doctoral dissertation, National Taiwan Normal University, Taipei, Taiwan.

Shukla, Shaligram. 2000. *Hindi Phonology*. Muenchen: Lincom Europa.

Siegel, Dorothy Carla. 1974. Topics in English morphology. Doctoral dissertation, Massachusetts Institute of Technology.

Sievers, Eduard. 1881. *Grundz uge der Phonetik*. Leipzig: Breitkopf and Hartel.

Simon, Imre. 1975. Piecewise testable events. In *Automata Theory and Formal Languages*, 214–222.

Sipser, Michael. 2012. *Introduction to the Theory of Computation*. 3rd ed. Cengage Learning.

Siptar, Peter. 2005. The phonology of Hungarian vowel clusters. *Magyar Nyelv (Hungarian Language)* 3:282–304.

Sproat, Richard William. 1992. *Morphology and computation*. Cambridge,MA: MIT press.

Stabler, Edward P. 1997. Derivational minimalism. In *Logical aspects of computational linguistics*, edited by Christian Retoré, vol. 1328 of *Lecture Notes in Computer Science*, 68–195. Berlin: Springer.

Stabler, Edward P. 2004. Varieties of crossing dependencies: structure dependence and mild context sensitivity. *Cognitive Science* 28:699–720.

Staubs, Robert, Michael Becker, Christopher Potts, Patrick Pratt, John J. McCarthy, and Joe Pater. 2010. Ot-help 2.0. software package. URL `http://people.umass.edu/othelp/`

Steriade, Donca. 1982. Greek prosodies and the nature of syllabification. Doctoral dissertation, MIT, Cambridge, Mass.

Steriade, Donca. 1988. Reduplication and syllable transfer in Sanskrit and elsewhere. *Phonology* 5:73–155.

Stowell, Timothy Angus. 1981. Origins of phrase structure. Doctoral dissertation, Massachusetts Institute of Technology.

Strother-Garcia, Kristina. 2018a. Imdlawn Tashlhiyt Berber syllabification is quantifier-free. In *Proceedings of the Society for Computation in Linguistics*, vol. 1. Article 16.

Strother-Garcia, Kristina. 2018b. Imdlawn Tashlhiyt Berber syllabification is quantifier-free. In *Proceedings of the Society for Computation in Linguistics*, vol. 1, 145–153.

Strother-Garcia, Kristina. 2019. Using model theory in phonology: A novel characterization of syllable structure and syllabification. Doctoral dissertation, University of Delaware.

Strother-Garcia, Kristina, Jeffrey Heinz, and Hyun Jin Hwangbo. 2017. Using model theory for grammatical inference: A case study from phonology. In *Proceedings of The 13th International Conference on Grammatical Inference*, JMLR: Workshop and Conference Proceedings, 66–78.

Struijke, Carolina Maria. 2000. Reduplication, feature displacement, and existential faithfulness. Doctoral dissertation, University of Maryland, College Park.

Tesar, Bruce. 2014. *Output-driven Phonology*. Cambridge University Press.

Thomas, Wolfgang. 1982. Classifying regular events in symbolic logic. *Journal of Computer and Systems Sciences* 25:370–376.

Thomas, Wolfgang. 1997. Languages, automata, and logic. In *Handbook of Formal Languages*, edited by Grzegorz Rozenberg and Arto Salomaa, vol. 3, 389–455. New York, NY, USA: Springer-Verlag New York, Inc. URL http://dl.acm.org/citation.cfm?id=267871.267878

Trommer, Jochen. 2013. Stress uniformity in Albanian: Morphological arguments for cyclicity. *Linguistic Inquiry* 44:109–143.

Trubetzkoy, Nikolai S. 1969. *Principles of phonology*. Berkeley & Los Angeles: University of California Press. Originally published 1939 as Grundzüge der Phonologie. Göttingen: van der Hoeck & Ruprecht.

Urbanczyk, Suzanne. 2007. Themes in phonology. In *The Cambridge Handbook of Phonology*, edited by Paul de Lacy, 473–493.

Urbanczyk, Suzanne. 2011. Reduplication. In *Oxford Bibliography*, edited by Mark Aronoff.
URL http://oxfordindex.oup.com/view/10.1093/obo/9780199772810-0036

Vaux, Bert. 1998. *The phonology of Armenian*. Oxford: Clarendon Press.

Vu, Mai Ha, Ashkan Zehfroosh, Kristina Strother-Garcia, Michael Sebok, Jeffrey Heinz, and Herbert G. Tanner. 2018. Statistical relational learning with unconventional string models. *Frontiers in Robotics and AI* 5:1–26.

Walther, Markus. 2000. Finite-state reduplication in one-level prosodic morphology. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, NAACL 2000, 296–302. Seattle, Washington: Association for Computational Linguistics.
URL http://dl.acm.org/citation.cfm?id=974305.974344

Wang, Yang. 2021a. Recognizing reduplicated forms: Finite-state buffered machines. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 177–187. Online: Association for Computational Linguistics.

Wang, Yang. 2021b. Regular languages extended with reduplication: Formal models, proofs and illustrations. Master's thesis, University of California, Los Angeles.

Watters, James. 1988. Topics in Tepehua grammar. Doctoral dissertation, University of California, Berkeley.

White, James. 2017. Accounting for the learnability of saltation in phonological theory: A maximum entropy model with a p-map bias. *Language* 93:1–36.

Wiese, Richard, ed. 1994. *Recent developments in lexical phonology*. Düsseldorf: Heinrich-Heine-Universität.

Wilbur, RB. 1982. A multi-tiered theory of syllable structure for american sign language. In *Annual Meeting of the Linguistic Society of America, San Diego*.

Wilbur, Ronnie. 2011. Sign syllables. *The Blackwell companion to phonology* 1:1309–1334.

Wilbur, Ronnie B. 2005. A reanalysis of reduplication in American Sign Language. In Hurch (2005), 595–623.

Wilbur, Ronnie Bring. 1973. The phonology of reduplication. Doctoral dissertation, University of Indiana, Bloomington, Indiana.

Williams, Sheila M. 1994. Lexical phonology and speech style: Using a model to test a theory. In *Computational Phonology*.

Williams, Sheila Margaret. 1993. Lexphon: A computational implementation of aspects of lexical phonology. Doctoral dissertation, University of Reading.

Wilson, Colin, and Gillian Gallagher. 2018. Accidental gaps and surface-based phonotactic learning: a case study of South Bolivian Quechua. *Linguistic Inquiry* 49:610–623.

Woollams, Geoff. 1996. *A Grammar of Karo Batak, Sumatra*. Canberra: Pacific Linguistics.

Yanti, and Eric Raimy. 2010. Reduplication in Tanjung Raden Malay. In *Linguistik Aktuell/Linguistics Today*, edited by Raphael Mercado, Eric Potsdam, and Lisa deMena Travis, vol. 167, 25–44. Amsterdam: John Benjamins Publishing Company.

Yip, Moira. 2002. *Tone*. Cambridge University Press.

Yu, Alan CL. 2007. *A Natural History of Infixation*. No. 15 in Oxford Studies in Theoretical Linguistics. Oxford: Oxford University Press.

Zetzsche, Georg. 2018. Separability by piecewise testable languages and downward closures beyond subwords. In *LICS '18: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, 929–938. New York, NY: Association for Computing Machinery.

Zhang, Jie. 2004. The role of contrast-specific and language-specific phonetics in contour tone distribution. In Hayes *et al.* (2004).

Index